VRIJE
UNIVERSITEIT
BRUSSEL

Graduation thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Applied Sciences and Engineering: Computer Science

# STORYTELLING AS A MEANS TO IMPROVE SOCIAL INCLUSION IN COMPUTER SCIENCE EDUCATION

Inas Ghazouani Ghailani

June 2024

Promotor: Prof. Dr. Beat Signer
Advisor: Yoshi Malaise

**Sciences and Bioengineering Sciences**

VRIJE
UNIVERSITEIT
BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

# HET GEBRUIK VAN STORYTELLING OM SOCIALE INCLUSIE TE VERBETEREN IN COMPUTERWETENSCHAPPEN EDUCATIE

Inas Ghazouani Ghailani

Juni 2024

Promotor: Prof. Dr. Beat Signer
Advisor: Yoshi Malaise

**Wetenschappen en Bio-ingenieurswetenschappen**

# Abstract

One of the biggest challenges faced by non-profit organisations that provide computer science education for underrepresented people is high dropout rates. These rates are the result of a number of factors that impact both students and teachers. In this thesis, we aimed to address this problem. In order to do so we started by visiting HYFBE, a Belgian non-profit organisation teaching web development to refugees and migrants, and we performed a literature review to expand our understanding of the domain. Based on the insights gained during this process, we developed the JsStories tool. The tool helps students learn JavaScript through storytelling. To minimise any barriers to entry and to maximise the feeling of connection to the story, we incorporated stories from HYFBE's alumni. Additionally, educational best practises such as the PRIMM approach and suggesting level-appropriate content based on knowledge graphs were followed to build the tool. Throughout this thesis, we have used the Design Science Research Methodology. We conducted interviews and a survey to evaluate a first prototype of the tool. Based on the positive results of the evaluation, we believe the tool could be beneficial for organisations such as HYFBE to use both during their programs and in the student selection procedure, providing insights to students about what to expect from such programs.

# Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor Yoshi Malaise, without whom I could not have undertaken this journey. His weekly advice, guidance and support have been invaluable in shaping this thesis. I could truly not have imagined having a better advisor.

Furthermore, I am extremely grateful to my supervisor, professor Beat Signer, for giving me this opportunity and for his valuable advice and feedback.

I would also like to extend my sincere thanks to the HYFBE alumni who contributed to this thesis by sharing their stories and without whom the conducted research would not have been possible. I wish them all the best in their lives in Belgium.

Special thanks also to Evan Cole for his helpful feedback and given insights on how to enhance the JsStories tool. Many thanks also to Laura van der Lubbe, whose expertise in knowledge graphs has significantly influenced the creation of the one used in the JsStories tool.

I would also like to thank the students and the education coordinator I talked to during my visit at HYFBE, who helped me acquire a better understanding of the difficulties students face in such programmes.

In addition, I am thankful for the representatives of HYF Denmark, Borders None, Open Cultural Center and Social Hackers Academy, as well as the survey participants, that contributed to the evaluation of the tool.

Last but not least, I would like to thank my family and friends for their continuous emotional support, encouragement and for believing in me.

# Contents

# Chapter 1

# Introduction

## 1.1 Problem Statement

As computer scientist Wasseem Latif once said: *"Give a man a program, frustrate him for a day. Teach a man to program, frustrate him for a lifetime."* Learning and teaching programming is up until this day well-known to be difficult [19, 39, 7] as evidenced by the failure rates in programming courses that can undoubtedly be improved [48]. The primary reasons for this challenge are, among others, students' poor problem-solving skills as well as shortcomings in the way programming is taught, including the learning materials utilised [7]. Several crucial aspects that could be improved in the current approach to teaching programming have been identified [19]. The suggestions for improvement include making the teaching of programming more flexible, enabling students to learn at their own pace and through various methods. Students should also be provided with enough and appropriate assistance. Additionally, it is important that the teachers are not only skilled programmers but also have strong pedagogical skills to effectively teach programming.

Furthermore, with the increasing digitalisation, STEM fields such as computer science are growing in importance and demand. However, statistical evidence shows that people of different races, ethnicities and socially vulnerable groups, such as refugees, are underrepresented in such fields [36]. Additionally, the same applies to people of diverse sexual orientations and gender identities [13]. There is also a significant gender disparity, as men outnumber women in STEM professions and education [21, 36]. Hence, numerous (non-profit) organisations worldwide aim to fight the inequality and high dropout rates from underrepresented groups. Migracode Europe[1], co-funded by the Erasmus+ program of the European union, fosters open-access technology education for refugees and migrants. Their network is made up of nine coding schools across several European countries. Among these, is HackYourFuture Belgium[2] (HYFBE), a non-profit organisation that offers a free web development programme to refugees and other disadvantaged groups, preparing them for their first job or internship in the IT industry. As such, HYFBE supports these people in their integration process regardless of their background and prior pro-

[1]`https://migracode.eu/about-migracode/`
[2]`https://hackyourfuture.be`

gramming knowledge while also addressing the high demand in the IT industry. The team at HYFBE comprises volunteers, including mentors and coaches, who are willing to share their knowledge and expertise with the students and provide support for them. The program consists of weekly classes that take place every Sunday. Additionally, during the week the students are expected to complete some homework assignments about the topics covered in class. However, these organisations are facing challenges in maintaining low dropout rates. We believe this is due to several factors. In addition to the complexity of programming, volunteer teachers may lack the necessary pedagogical skills for effective teaching as they are often not given a formal training. Furthermore, it is important to take into account the background and needs of these refugees and migrants to foster their sense of belonging. Therefore, we believe that a solution to tackle the high dropout rates in organisations aiming to teach programming to socially vulnerable groups should address each of these aspects while taking into account the mentioned suggestions to improve programming teaching.

## 1.2 Method

Throughout this thesis, we have made use of the *Design Science Research Method (DSRM)* in information systems research [38], which consists of six different phases. These are the following: (1) problem identification and motivation, (2) defining the requirements and objectives for the solution, (3) designing the solution and then developing it, (4) demonstrating the solution, (5) conducting an evaluation, (6) communicating the artefact along with the problem it aims to solve and the results of the evaluation.

The problem identification and motivation is described in the problem statement (Section 1.1). To gather the requirements and objectives of the solution, we studied several papers related to social inclusion in computer science education, the challenges of learning programming and computer science concepts, as well as studies on potential pedagogical approaches to be used in computer science education. In addition, we attended a Sunday session at HackYourFuture to interact directly with the target group and obtain more information on the requirements and objectives. There we had the opportunity to talk with the educational coordinator at that time and socially vulnerable students including refugees and asylum seekers. We discussed the curriculum of the 9-month web development programme, challenges faced by students, motivations for joining the program and any fears or assumptions the students initially had about computer science. These insights shaped the design and development of the JsStories tool. The artefact is demonstrated using screenshots presenting the tool's functionality. Additionally, the appendix contains a complete example of one of the books currently available in the JsStories tool. The artefact was then evaluated by means of interviews and a survey with both students and teachers. Finally, the findings and conclusions are communicated through this thesis.

## 1.3 Contributions

The main contribution of this thesis is the introduction of the JsStories tool, designed to help learn programming in JavaScript through storytelling. The tool provides an interface containing books consisting of stories and exercises based on the PRIMM model [43, 44]. In order to track students' progress, the tool is also built using the concept of a knowledge graph. By using storytelling, the PRIMM approach and a knowledge graph, the tool strives to enhance the social inclusion in organisations aimed at teaching socially vulnerable groups and potentially reduce the dropout rates. The stories used, which are real stories of HYFBE's alumni, focus on building a sense of belonging among socially vulnerable students. Additionally, the PRIMM approach and the knowledge graph provide a learning environment that allows students to learn at their own pace while offering support along the way. The stories are managed through a content management system allowing for story flexibility and thus support for various target groups. Although JsStories was initially developed to teach JavaScript in a social education setting, it may for instance also be adapted to teach programming to children by using appropriate stories such as fairytales. Another contribution involves the exploration of relevant models for effective programming pedagogy, the concepts of knowledge graphs and gamification, more specifically storytelling, which have been discussed in Chapter 2. Furthermore, we investigated the challenges students face in organisations like HYFBE through a visit there, talking to students and the educational coordinator. Additionally, insights on the challenges and internal workings of a number of Migracode schools, such as their student selection procedure have been acquired through interviews and documented in this thesis.

## 1.4 Thesis Outline

The remainder of this thesis is organised as follows: Chapter 2 contains background information on educational psychology focusing on models for programming pedagogy as well as some general education theories. Furthermore, gamification including storytelling is discussed. Finally, this chapter discusses knowledge graphs. The following chapter covers some relevant work in the research domain. This includes work that is related to the application of certain concepts described in Chapter 2, as well as work that is pertinent due to its utilisation during the implementation of the tool. Additionally, Chapter 4 provides a detailed, illustrated description of the tool that was developed as a potential solution for the problem statement. Using the background information described in Chapter 2, the motivation behind every aspect of the tool is explained. Furthermore, in Chapter 5 information is provided on the collection and writing processes for the stories included in the tool. Moreover, it offers further details on the implementation of the tool and the motivation behind every implementation choice. The evaluation of the tool was conducted using interviews and a survey. Chapter 6 includes a thorough description of every interview that was carried out as well as an overview of the outcomes of the survey. Furthermore, Chapter 7 discusses potential future work to further improve the tool. Finally, a conclusion is provided in Chapter 8. The appendix contains an example of a book currently available in the JsStories tool, figures depicting the survey results and larger images of the screenshots used in Chapter 4.

# Chapter 2

# Background

## 2.1 Bloom's Taxonomy

Bloom's taxonomy, named after Benjamin S. Bloom and introduced in [5], is a popular model that is widely adopted in education. The taxonomy is a six-level hierarchical classification of learning objectives. It consists of the levels *knowledge*, *comprehension*, *application*, *analysis*, *synthesis* and *evaluation*, ranging from lower to higher levels, with the highest levels requiring more cognitive load than the lower ones [12, 1]. Teachers can use Bloom's taxonomy to structure their lessons such that their students go through each level progressively and only move on to the higher levels once they have mastered the lower ones [12].

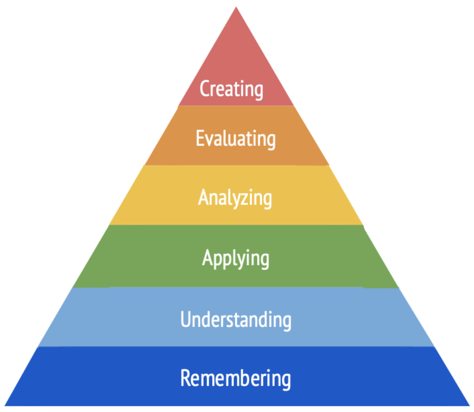


Figure 2.1: Revised Bloom's taxonomy [12]

The taxonomy was later on revised in [2] which included structural modifications and a renaming of the learning objectives, as illustrated in the pyramid shown in Figure 2.1. The learning objectives were renamed to *remembering*, *understanding*, *applying*, *analysing*, *evaluating* and *creating*, using verbs instead of nouns. In this version, creating is the highest level, as opposed to the original version where evaluation was the highest level. The objectives from lower to higher levels are listed in Table 2.1, along with their subcategories, a description, and an example which are given by the authors [2].

| | **Subcategories** | **Description** | **Examples** |
|---|---|---|---|
| **Remembering** | Recognising and recalling | Retrieve relevant knowledge from long-term memory | Recall the dates of important events in U.S. history |
| **Understanding** | Interpreting, exemplifying, classifying, summarising, inferring, comparing and explaining | Construct meaning from instructional messages, including oral, written, and graphic communication | Compare historical events to contemporary situations |
| **Applying** | Executing and implementing | Carry out or use a procedure in a given situation | Divide one whole number by another whole number, both with multiple digits |
| **Analysing** | Differentiating, organising and attributing | Break material into its constituent parts and determine how the parts relate to one another and to an overall structure or purpose | Structure evidence in a historical description into evidence for and against a particular historical explanation |
| **Evaluating** | Checking and critiquing | Make judgements based on criteria and standards | Determine if a scientist's conclusions follow from observed data |
| **Creating** | Generating, planning and producing | Put elements together to form a coherent or functional whole; reorganise elements into a new pattern or structure | Generate hypotheses to account for an observed phenomenon |

Table 2.1: Revised Bloom's taxonomy [2]

## 2.2 Code Reading and Tracing

In 2004, Lister et al. [28] looked into the reasons why first-year CS students still have difficulties writing code after completing their introductory programming classes. Their study consisted of providing students from multiple countries with twelve multiple choice questions to answer, which tested their skills in predicting what a code snippet will do as well as their skills in completing some given code. Based on their study, they found that in most cases the students have difficulties writing code not only because they lack problem-solving skills, which was the main explanation given in [33], but more importantly because the students have difficulties reading code and understanding it. The study was conducted again a couple of years later [27] to make sure that no biases due to other factors, such as the programming language covered by the questions, were included. Additionally, they made sure that the students who participated had not participated to the original study. The results of this second study confirmed those of the previous one, as they found that there was a clear correlation between the abilities to trace, explain and write code.

## 2.3 Models for Programming Pedagogy

### 2.3.1 The Block Model

The Block model, a model that focuses on the understanding of programs, was introduced by Schulte [42]. The model is depicted in Table 2.2 as a 4x3 table representing three dimensions over four different levels. Each cell or block (hence the name of the model) in the table emphasises a particular part of the comprehension process. The first two dimensions *Text surface* and *Program execution (data flow and control flow)*, both concern the structure of the program. *Text surface* refers to the actual text of the program, whereas *Program execution (data flow and control flow)* deals with the order in which the program is executed as well as the manipulation of data. The third dimension is *Goals of the Program*, which is about what the program is intended to do. Furthermore, the four different levels are *Atoms* (the language elements such as keywords), *Blocks* (the code blocks/units made up of atoms), *Relations* (the relations between the different blocks) and *Macro Structure* (the whole program). The first step of program comprehension involves reading the program text and retrieving information, which is added to the reader's mental model. It is typically a bottom-up process in the table where first the atoms, then the code blocks, then the relationships between those, and eventually the whole program are considered, involving one or multiple dimensions.

The model can be used to plan lessons in programming courses, focusing on code comprehension. Moreover, the author states that the blocks/cells of the model should be viewed as *movable*, meaning that when planning a lesson, not every block has to be considered and also that the order in which the blocks are arranged may vary in accordance with different learning trajectories. In a 2019 paper [18], Izu et al. proposed some examples of exercises based on the Block model that may be used when teaching programming to improve program comprehension. In Table 2.3 one of these examples is given for each block.

Furthermore, the Block model may be seen as a taxonomy, similar to Bloom's taxonomy, because it enables the classification of various code comprehension activities and questions into the different blocks. In addition, the hierarchy is somewhat also present in the Block Model since, for instance, the students first have to comprehend atoms before they can understand the macro structure. Both models were compared in [49] and although there are some similarities, according to the authors, the Block model "*provided a better way of describing novice programming code comprehension tasks because of the increased granularity that it provides*".

| | | | |
|---|---|---|---|
| **Macro Structure** | Understanding the overall structure of the program | Understanding the 'algorithm' of the program | Understanding the goal/purpose of the program |
| **Relations** | References between blocks, eg. method calls, object creation, accessing data... | Sequence of method calls - 'object sequence diagrams' | Understanding how subgoals relate to goals, how function is achieved by subfunctions |
| **Blocks** | 'Regions of Interests' (ROI) that syntactically or semantically build a unit | Operation of a block of code, a method, or a ROI (as a sequence of statements) | Purpose of a block of code, possibly seen as a subgoal |
| **Atoms** | Language elements | Operation of a statement | Purpose of a statement |
| | **Text Surface** | **Program Execution (data flow and control flow)** | **Goals of the Program** |
| **Duality** | **Structure** | | **Function** |

Table 2.2: The Block model, taken from [42]

| | | | |
|---|---|---|---|
| **Macro Structure** | Represent the overall program structure by drawing a tree of function/procedure dependencies (relative to invocations) | Verify if a program statement or block is ever reachable during program execution | Select the sentence, from a few options, which most accurately summarises the program's purpose |
| **Relations** | Link each occurrence of a variable with its declaration | Identify the scope of a variable. | Summarise in a short sentence the purpose of a simple block invoking one or more procedures/functions |
| **Blocks** | Draw a box around the code of each loop | Change a for loop into a while loop | Write comments explaining the purpose of a block and of the statements it is built from |
| **Atoms** | Identify the keywords in a piece of code | Determine the value of an expression for given values of the involved variables | Rename a constant with an appropriate name from the problem |
| | **Text Surface** | **Program Execution (data flow and control flow)** | **Goals of the Program** |
| **Duality** | **Structure** | | **Function** |

Table 2.3: Examples of activities/questions for each cell in the Block model, based on [18]

### 2.3.2 Use-Modify-Create

In [24], Lee et al. looked into the development of Computational Thinking (CT) in middle and high school students. CT is considered to be an essential aspect of computer science as it denotes the thinking abilities needed to be able to define, comprehend and analyse problems as well as solving these with the use of a computer, automation and abstraction [24]. Based on their research, they introduced a learning progression consisting of three phases, called *Use-Modify-Create* (UMC), to help students progress in CT. Figure 2.2 illustrates the UMC learning progression. In the *Use* phase, the students make use and explore other people's work. The students may be given some program, for instance, that they can analyse and run to test its outcome. Then in the *Modify* phase, they can make small modifications to the work provided to them, like modifying the program's use of a certain visual aspect such as a colour. Lastly, in the *Create* phase, after having gained more understanding of the original program and experience with the small modifications, the students can make more important modifications, such as adding some extra functionalities. This requires the students to add their own written code to the program. Following an iterative process of testing, analysing, and refining their modifications, the students ultimately get ownership of the final altered program. After having been through the UMC phases for different programs, the students get more confident to write entirely new programs based on the knowledge they have gained through these different phases. Furthermore, the model can be used for different levels of complexity. For instance, after going through the UMC phases introducing basic programming principles, the students might proceed to going through the three phases for somewhat more advanced programming concepts, and so on. The model thus makes sure that the students get more comfortable to write programs by progressively going through all these phases for different levels of complexity. Also, the model ensures (in the context of programming) that students have a thorough understanding of programs before they are required to write them from scratch.

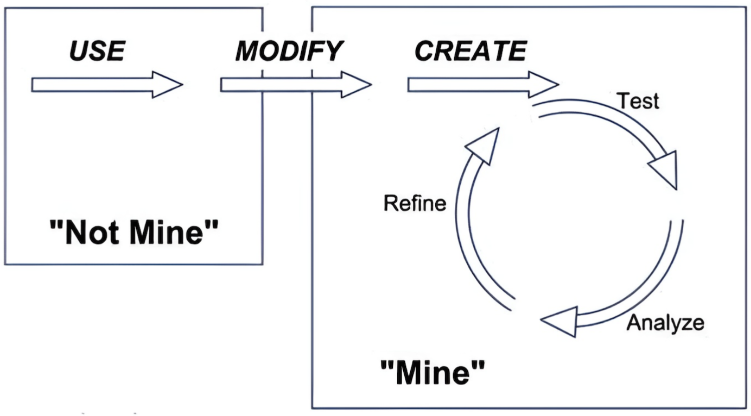


Figure 2.2: The Use-Modify-Create progression, taken from [24]

### 2.3.3 PRIMM

PRIMM is a pedagogical model for both learning and teaching programming that was introduced by Sentance, Waite and Kallia [43, 44]. It represents the following five distinct activities:

- **Predict** stands for predicting what the provided code will do. Examples of predict activities are predicting the output of the given code and predicting state changes.
- **Run** stands for testing the predictions of the predict step by running the provided code.
- **Investigate** stands for inspecting the structure of the provided code. An example of an investigate activity would be answering code comprehension questions about the structure of the code such as for example: What happens if in line 4 the variable `x` is replaced by `y`?
- **Modify** stands for making code modifications. The addition of functionality to the provided code and the removal of bugs in the code are two examples of modify actions.
- **Make** stands for utilising the same or modified structures from the previous four activities to make a new program from scratch.

The model is based on the Use-Modify-Create framework, described in the previous section. PRIMM only focuses on programming, as opposed to the UMC framework, which can be used for any type of computational thinking, including modelling, designing, programming, and more. In order to better reflect the concept of programming, the authors expanded the *Use* phase of the UMC framework to include the three different activities: *Predict*, *Run*, and *Investigate*. Furthermore, the *Modify* and *Make* activities of PRIMM correspond to the *Modify* and *Create* phases of the UMC framework. PRIMM also makes use of different levels of abstraction, similar to the ones used in the Block model (described in Section 2.3.1), as the five different activities each focus on a particular abstraction level. *Predict* and *Run* emphasises the execution of the program whereas *Investigate* focuses on the program itself, mainly its structure and *Make* concentrates on the purpose and function of the program. Moreover, as the goal of the predict phase is to first read the code in order to understand it and be able to predict its outcome, it also relies on the work related to the importance of the ability to trace and read code, described in Section 2.2.

PRIMM can thus be used in programming courses by dividing the lessons into the five different activities. In [43], the authors examined the experiences of programming teachers with PRIMM. The teachers were first introduced to it and were provided with some materials containing exercices that they could use during their lessons. They were also free to adapt the exercices if they wanted to do so. Following the introduction to the model, the teachers implemented it in their programming courses for a couple of months. Afterwards, they were interviewed about how the lessons went, whether they felt confident using the PRIMM method and on the impact it had on the student's performances. Based on the feedback from the teachers, the authors concluded that using PRIMM helped improve the understanding of programming for students but it also had a positive impact on the structure and executions of the lessons. Furthermore, the favourable effect PRIMM had on student's performances was also described in greater depth in [44]. The model was evaluated in several high schools with the students being divided in a control group and an experimental group. To assess their prior knowledge and basic comprehension of programming, all of the students were required to first take a baseline test in the form of a multiple-choice quiz. Afterwards both groups followed programming lessons for several months, but the lessons given to the experimental group were based on the PRIMM method. Subsequently, both groups completed a post-test. The findings show that the experimental group significantly outperformed the control group on the post-test, despite the fact that both groups had similar scores on the baseline test taken prior to the programming lessons.

## 2.4 Expertise Reversal Effect

The term *expertise reversal effect* is well-known within the field of education psychology. This effect refers to situations in which learning materials that are advantageous for beginners turn out to be unfavourable for more experienced learners [41, 47]. In other words, the effect arises when the effectiveness of the learning materials decreases as learners become more experienced. The expertise reversal effect is assumed to be a result of redundancy and the limitations of the working memory in terms of capacity and duration [41]. For instance, as more experienced learners do not require as much additional information as beginners, the information becomes redundant and the working memory is needlessly overloaded. According to Wolfgang Schnotz [41], in order for learning to be effective, instructional support and individual learning must be rightly balanced in order to match the learning materials to the learners level of expertise.

## 2.5 Formative Assessment

Formative assessment can have a significant impact on students' learning performance, according to a thorough literature review on the subject by Black and William [4]. They defined formative assessment as follows: "*Assessment refers to all those activities undertaken by teachers and — by their students in assessing themselves — that provide information to be used as feedback to modify teaching and learning activities. Such assessment becomes formative assessment when the evidence is actually used to adapt the teaching to meet student needs*". Whereas summative assessment consists of occasional, larger and graded activities, formative assessment is about more regular, smaller tasks that are generally ungraded [14]. A typical summative assessment could be a graded exam at the end of the academic year covering all taught material. On the other hand, an example of formative assessment could be an ungraded homework covering a specific topic taught in class that would be corrected by the teacher to provide feedback. As Graham [14] states, formative assessment is not only helpful for teachers to adapt their teaching to the student needs but it also helps students themselves to recognise their limitations in their understanding of certain topics allowing them to work on it.

GoFormative[1] (or also called Formative) is a web-based platform for digital formative assessments. It is free for both teachers and students. Using the platform teachers can create assessments based on various pre-defined types of questions, assign the assessments to selected students, view their responses and provide individual feedback that can immediately be viewed by the students [14].

[1] `https://goformative.com`

## 2.6 Parsons Problem

In a Parsons problem or puzzle, first presented in 2006 by Parson and Haden in [37], a code program is fragmented into several code blocks or lines appearing in a wrong order, and the purpose is to rearrange the code fragments to form the correct code [37]. A 2008 study on Parsons problems [8] showed that a significant amount of students that scored well on Parsons problems also performed well on code writing exercises, indicating that the two are positively correlated and that similar skills are needed for both. Furthermore, the authors state that grading Parsons problems is simpler than grading traditional code writing exercises. For instance, a Parsons problem may be graded by assigning 1 point for each correctly ordered code line. Also, several variants of the problem exists such as, the Two-dimensional Parsons problem and the Parsons problem including distractors. When reordering the code fragments in the former, indentation should also be taken into account. Furthermore, in the latter, distractors can be extra erroneous code fragments or code fragments that do not belong to the code program and should thus not be used when forming the solution [11]. The authors of [11] analysed Two-dimensional Parsons problems with distractors in contrast to modifying faulty code and writing code. They conducted a study and found that in terms of learning outcomes and knowledge retention, based on pretests and posttests, there was no significant difference between the three. However, completing Two-dimensional Parsons problems required substantially less time than the other two, making it more efficient.

## 2.7 Gamification

According to the definition given by Deterding et al., *gamification* refers to "*the use of game design elements in non-game contexts*" [9]. In game studies, a distinction is made between the terms *game* and *play* as the former denotes the type of playing following rules and aiming to achieve some goals, whereas the latter does not involve any rules or goals. Gamification refers only to the former [9]. As per the definition, a gamified application is in and of itself not considered to be a game but rather an application that contains certain game elements. The authors also proposed the following definition for what should be considered as game elements: "*elements that are characteristic to games — elements that are found in most (but not necessarily all) games, readily associated with games, and found to play a significant role in gameplay*". Moreover, as mentioned in the definition of gamification, the game elements used are more specifically *game design elements*. Those can be divided into five distinct levels of abstractions that can all be used in gamification: *Game interface design patterns*, *Game design patterns and mechanics*, *Game design principles and heuristics*, *Game models*, *Game design methods* [9]. The five levels along with a short description and some examples are depicted in Table 2.4. Furthermore, gamification as well as toys, playfull design and serious games are situated along two axes in Figure 2.3 based on the gaming/playing and parts/whole aspects. Serious games are considered to be a type of game that is intentionally created with a serious purpose and thus not merely for amusement [3, 9]. Similarly, gamification can be used in any non-game context such as education or employee training for some specific purpose other than entertainment [9].

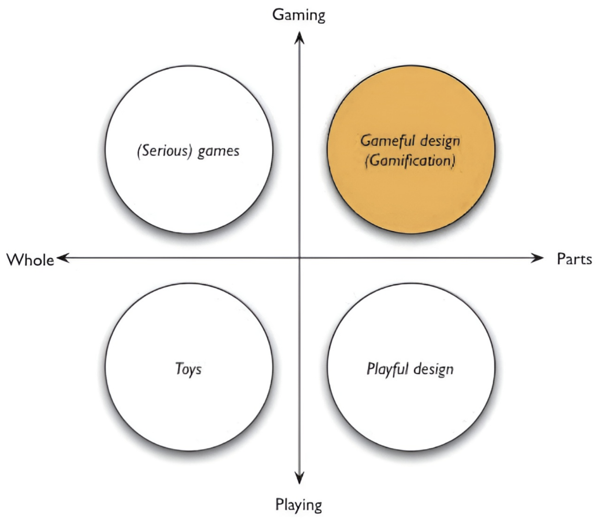


Figure 2.3: Situating gamification, taken from [9]

Gamification gained popularity in 2010 and has since remained a major topic in academia [30, 15], particularly with regard to its application in the context of education [30]. According to a literature review by Nah et al. [35], eight particular game interface design elements are most frequently used in education. These are the following: Points, Levels/Stages, Badges, Leaderboards, Prizes/Rewards, Progress bars, Storyline/Narrative and Feedback. The authors also provided a description for each of these game elements, which is summarised in the following lines. There are several types of points but they all have the same general purpose, which is to indicate accomplishments. Points that are gained when specific activities are accomplished are called Experience Points. Furthermore, earned points can in some cases not only serve as an indication of achievement but also as virtual currency. These are known as Steam Points. For instance, in some games, these points can be used to acquire hints. Additionally, the authors state the usefulness of rewards and prices as it helps keeping the learners motivated. In order to maximise the impact that rewards and prices have on the learners' motivation it is best to give several small rewards at various points during their learning journey. An example of a reward is a character upgrade in a gamified application that uses characters. Another way to retain the students' motivation to continue their learning process by completing upcoming learning task is earning badges, which can be obtained after achieving a given milestone. Further, to keep track of the learners' progression, levels/stages as well as progress bars can be used. Levels are often used in games with the first few levels being less challenging than the higher ones in terms of efforts needed to complete the level. The learners may have a feeling of satisfaction after completing a level, which can drive them to continue with the next levels. Furthermore, progress bars can be used by the learners to monitor their progression regarding the main learning goal or a particular sub-goal. The progress bars mainly have a boosting effect on the learners to achieve the particular goal, especially when they are close to doing so. Another game element that can certainly help with keeping the learners motivated, is a leaderboard. On a leaderboard, the learners are ranked based on their obtained score, which could result from the accumulation of their experience points. Because they can compare their score and ranking with those of other learners, it might encourage them to achieve a higher rank. To avoid discouraging learners with low scores, only those with the highest scores should be displayed on the leaderboard, such as the top 5 or top 10. In terms of performance, feedback is a key game element as it improves the learning experience and efficacy while also keeping the learners interested. The usefulness of the feedback element depends on how often the feedback is given, when it is provided and how explicit it is. The more explicit the feedback and the sooner it is provided, the better for engaging the students. Consider, for instance, textual feedback that immediately appears after completing a task, to indicate what was done incorrectly and why it is incorrect. Finally, the storytelling/narrative element relates to the usage of a story in the gamified application. Having a story include the aspects to be learned provides a context and is a way to improve the learners' engagement.

| Level | Description | Example |
|---|---|---|
| Game interface design patterns | Common, successful interaction design components and design solutions for a known problem in a context, including prototypical implementations | Badge, leaderboard, level |
| Game design patterns and mechanics | Commonly reoccurring parts of the design of a game that concern gameplay | Time constraint, limited resources, turns |
| Game design principles and heuristics | Evaluative guidelines to approach a design problem or analyze a given design solution | Enduring play, clear goals, variety of game styles |
| Game models | Conceptual models of the components of games or game experience | MDA; challenge, fantasy, curiosity; game design atoms |
| Game design methods | Game design-specific practices and processes | Playtesting, playcentric design, value conscious game design |

Table 2.4: Game design elements, the five levels of abstractions [9]

## 2.8 Storytelling and Adult Learning

Enzo Caminotti and Jeremy Gray conducted a thorough literature study on the impact of storytelling on adult learning [6]. The authors state that in contrast to child education, interest and experience play a major role in adult education. Adults have a wealth of experiences that shape who they are and tend to learn based on their own interests. Hence, the authors discuss experience, role-play and case studies as three types of effective storytelling for adult education. In the first, students share their story composed of their experiences. This allows teachers to link newly acquired knowledge and information to what the students already knew from prior experiences. Furthermore, role-play involves having the students take on a role using their own knowledge and experience. Lastly, case studies covering a real-world scenario can be used for practice. Moreover, according to Marsha Rossiter [40], stories make new information easier to remember and has an impact on the involvement of students based on how much they can relate to the story and their emotional connection to the narration. The author provides an example of how reading about a success story of a character they can relate to might inspire and motivate adult learners who are having difficulty.

## 2.9 Knowledge Graphs

Over the last decade, knowledge graphs have been widely used in various application domains such as medical care, cyber security, journaling and education [50]. According to Hogan et al. [17], the first occurrence in literature of the term *knowledge graph* dates back to as early as 1972. However, it was the introduction of Google's knowledge graph on their blog in 2012 that considerably contributed to the term's widespread use [46]. Google's knowledge graph, as described in the video on the blog post, is essentially a graph-structured database containing worldly facts and the relationships between these. It was launched to improve users' experiences with the search engine using semantics. Three improvements were made using the knowledge graph: more accurate search results, the ability to provide a summary containing information relevant to the search query and, the capability to provide users with further information that might be interesting based on others past searches [46]. However, their blog post merely described the improvements the knowledge graph will make to the search engine without providing further detail about the knowledge graph itself or its implementation. Following Google's announcement, the term *knowledge graph* began to be widely used without an explicit definition. Therefore, Ehrlinger and Wöß proposed the following definition in their 2016 publication [10]: "*A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge*". In Figure 2.4 the architecture of the knowledge graph is depicted. From the definition and the architecture it is clear that although the terms knowledge base and knowledge graph are often interchanged in some publications, they are not the same. A knowledge base contains a set of information while a knowledge graph consists of a knowledge base supplemented with a reasoning engine to extract additional new information.

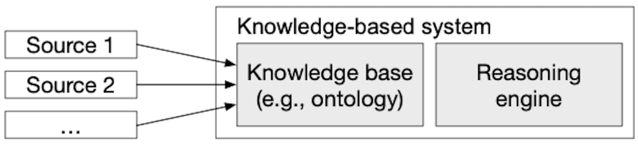


Figure 2.4: Knowledge graph architecture, taken from [10]

To illustrate with a simple example, consider Figure 2.5 depicting a knowledge base and the corresponding graph-structured version. In this example, the knowledge base consists of factual triples such as *(Albert Einstein, sonOf, Hermann Einstein)* indicating that Hermann Einstein was the father of Albert Einstein and *(Hans Albert Einstein, sonOf, Albert Einstein)* indicating that Albert Einstein was the father of Hans Albert Einstein. We can infer using reasoning that Hermann Einstein was the grandfather of Hans Albert Einstein even though this was not information that was provided in the knowledge base, i.e. the knowledge base did not include the triple (Herman Einstein, isGrandfatherof, Hans Albert Einstein).

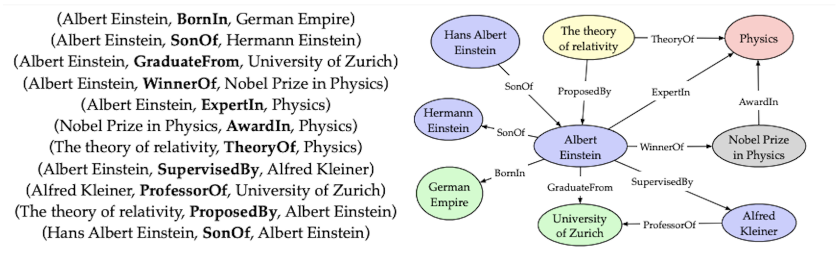

(Albert Einstein, **BornIn**, German Empire)
(Albert Einstein, **SonOf**, Hermann Einstein)
(Albert Einstein, **GraduateFrom**, University of Zurich)
(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)
(Albert Einstein, **ExpertIn**, Physics)
(Nobel Prize in Physics, **AwardIn**, Physics)
(The theory of relativity, **TheoryOf**, Physics)
(Albert Einstein, **SupervisedBy**, Alfred Kleiner)
(Alfred Kleiner, **ProfessorOf**, University of Zurich)
(The theory of relativity, **ProposedBy**, Albert Einstein)
(Hans Albert Einstein, **SonOf**, Albert Einstein)

Figure 2.5: Knowledge base comprising factual triples and its equivalent graph-structured version, taken from [20]

# Chapter 3

# Related Work

## 3.1 Storytelling Alice

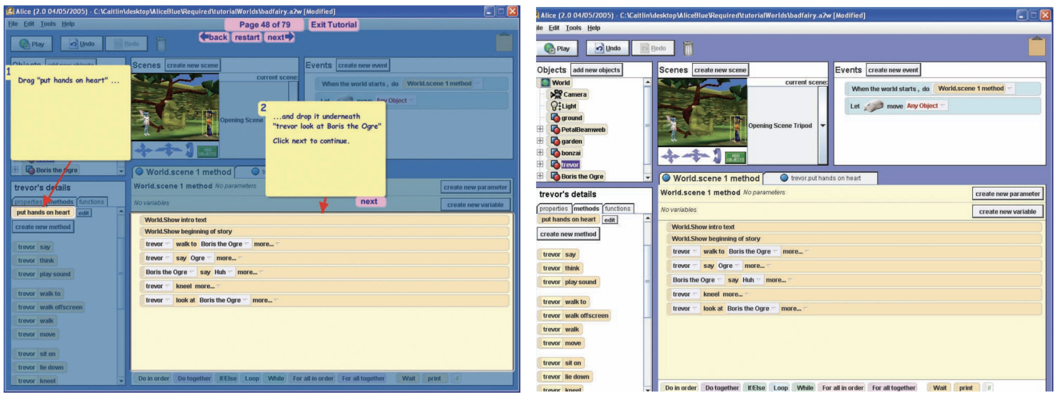

Drag "put hands on heart" ...
...and drop it underneath "trevor look at Boris the Ogre" Click next to continue.
Page 48 of 79
Exit Tutorial
back
restart
next
Objects
Scenes
Events
World.scene 1 method
trevor's details
World.Show intro text
World.Show beginning of story
trevor walk to Boris the Ogre more...
trevor say Ogre more...
Boris the Ogre say Huh more...
trevor kneel more...
trevor look at Boris the Ogre more...
Do in order
Do together
If/Else
Loop
While
For all in order
For all together
Wait
print

Figure 3.1: Interface of Storytelling Alice, taken from [23]

Caitlin Kelleher and Randy Pausch introduced *Storytelling Alice* in 2007, which is a tool to produce 3D animated movies while learning programming [23]. The idea was to make learning how to program more engaging by adding the storytelling aspect. *Storytelling Alice* builds further on *Alice 2.0*[1], a tool that enables the creation of 3D virtual environments through programming, by dragging and dropping tiles containing code parts to create the final program. It may be used to practice programming with concepts such as variables, conditionals, looping and any other similar concept that is likely to be taught in first-year programming classes [22]. *Alice 2.0* helps reduce the frustration that novice programmers might experience during their learning journey, as the drag and drop construct prevents syntax errors and as bugs can be more easily detected in the animations, which represents the constructed program [23, 22]. It was shown in [34] that *Alice 2.0* not only helped students perform better but it also increased the retention rates of

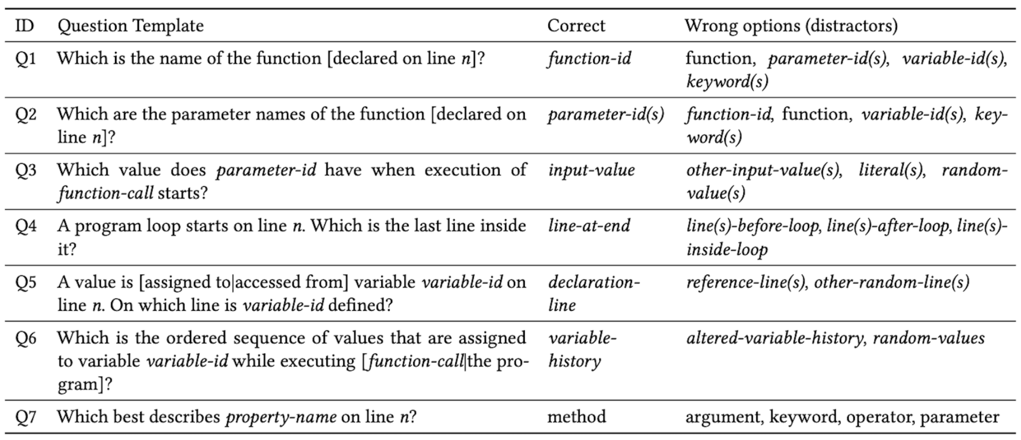
[1]http://www.alice.org

students intending to pursue a computer science education but having limited background in both computer science and mathematics, also referred to as high risk students. Kelleher [22] states that, even though *Alice 2.0* is effective in improving the retention rates, it does not necessarily help with drawing new students into computer science, which is why *Storytelling Alice* was introduced. *Storytelling Alice* mainly targeted middle school girls, as according to the authors, girls are less likely to engage in computer science education. The authors carried out a study in which they examined the effect of *Storytelling Alice* on the girls' drive to learn programming and their performance. They also looked at the effect of *Alice 2.0* and compared it to the effect of *Storytelling Alice* in order to focus on the impact of the storytelling aspect and eliminate the possibility of any other factor being the reason behind the obtained results. Overall, they found that *Storytelling Alice* had a positive impact on the girls' motivation to learn programming.

## 3.2 Questions About Learner's Code

Lehtinen, Santos and Sorva presented the concept of automatically generated *Questions about Learner's Code (QLCs)* in their 2021 paper [26]. According to the authors, it is often the case that students do not fully understand the code they have written, even though the code is working correctly. This may, for instance, be because they have used code snippets authored by others (plagiarism), or because they obtained the code via trial and error [26]. Therefore, the authors propose to prompt students with automatically generated questions about their own written code to enhance their code comprehension and learning. More recently, in a subsequent publication on the QLCs [25], the authors introduced a tool implementing the concept of automatically generated QLCs. The provided open-source library[2] generates multiple choice questions based on some specified javascript code. The QLCs include the different questions with multiple answer alternatives. Moreover, for each answer option, a brief explanation is provided on why it is the correct answer or not. In addition to the javascript code, a configuration about the QLCs can be provided to, among other things, indicate the number of QLCs that should be returned. The questions are generated based on the abstract syntax tree (AST) and the different variable values of the provided javascript code. The library can generate up to seven different kind of questions, which are listed in Figure 3.2. Besides the library, the authors also provided an example of how to incorporate QLCs in courses. They developed a tool that makes use of a web editor where students can exercise by writing code according to some given description. Once they submit the code and that it passes some tests, indicating that it is correct, they are prompted with a number of multiple choice questions regarding their code, to make them reflect about it.

[2]`https://github.com/teemulehtinen/qlcjs`



| ID | Question Template | Correct | Wrong options (distractors) |
|---|---|---|---|
| Q1 | Which is the name of the function [declared on line *n*]? | *function-id* | function, *parameter-id(s)*, *variable-id(s)*, *keyword(s)* |
| Q2 | Which are the parameter names of the function [declared on line *n*]? | *parameter-id(s)* | *function-id*, function, *variable-id(s)*, *keyword(s)* |
| Q3 | Which value does *parameter-id* have when execution of *function-call* starts? | *input-value* | *other-input-value(s)*, *literal(s)*, *random-value(s)* |
| Q4 | A program loop starts on line *n*. Which is the last line inside it? | *line-at-end* | *line(s)-before-loop*, *line(s)-after-loop*, *line(s)-inside-loop* |
| Q5 | A value is [assigned to\|accessed from] variable *variable-id* on line *n*. On which line is *variable-id* defined? | *declaration-line* | *reference-line(s)*, *other-random-line(s)* |
| Q6 | Which is the ordered sequence of values that are assigned to variable *variable-id* while executing [*function-call*\|the program]? | *variable-history* | *altered-variable-history*, *random-values* |
| Q7 | Which best describes *property-name* on line *n*? | method | argument, keyword, operator, parameter |

Figure 3.2: The different types of QLCs, taken from [25]

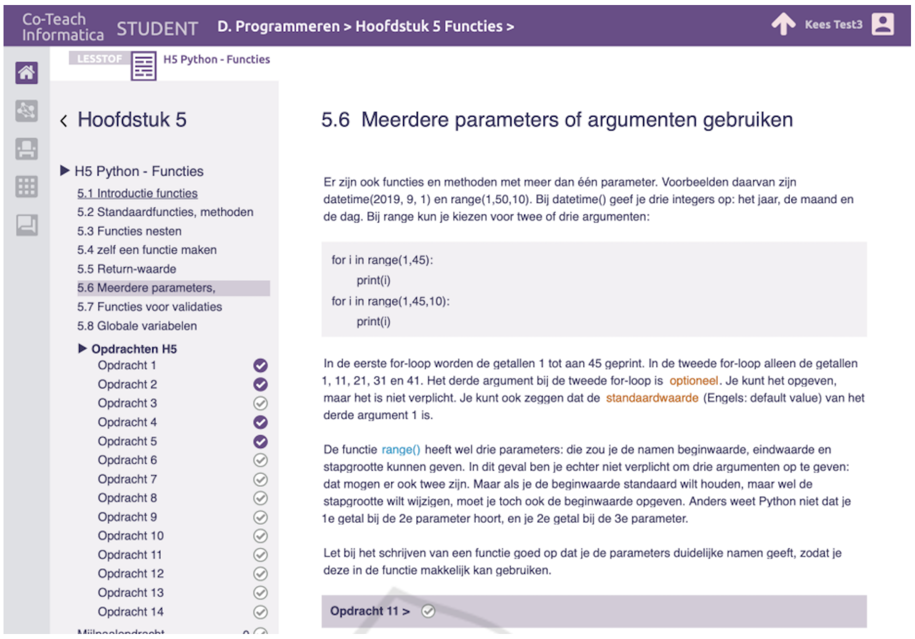
## 3.3 Knowledge Graph Use Cases

### 3.3.1 Co-teach Informatica's Digital Learning Platform

The importance of computer science is growing worldwide, however there are not enough computer science teachers, making it impossible for certain schools to offer a Computer Science course [29]. Co-Teach Informatica[3] provides a solution to tackle this problem in the Netherlands. They offer a program that may be added to the curriculum of schools wanting to provide a Computer Science course, but facing difficulty finding a qualified teacher. The program consists of an online learning platform on which students can follow modules autonomously in class while being supervised by a teacher. The students can get support for anything related to the content of modules via the remote support desk the program offers. Hence, the supervising teacher does not necessarily need any Computer Science background as their role is to administer the course and help the students with things like learning how to use the platform. The modules also include exercises and tests, which are graded by the teaching assistants of the remote support desk. In addition, the program includes guest lectures given by IT experts in which the students get to work on projects in order to gain hands-on experience. In their 2023 paper [29], Van der Lubbe et al. presented an online learning platform for the Co-Teach Informatica program. As the students work independently in that program, being able to track their progress is important for both the supervising teachers as well as the students themselves. That is why their design makes use of a *student model*, representing the current student's knowledge. The student model is built on top of a what they call, a *domain model*, which basically is a knowledge graph consisting of all the learning goals organised by chapter, as well as the dependencies between the different learning goals. The program modules are related to certain learning goals based on the activities they contain, so that as the student progresses through them, the student model is updated. Figure 3.3 depicts the learning platform from the viewpoint of a student. Furthermore, in Figure 3.4 an example of the learning goals and its dependencies for a particular chapter are illustrated. Unfortunately, as their work is very recent no evaluations have been done yet. However, they are

[3]`https://co-teach.nl/`

planning on conducting a pilot study in which their online learning platform will be evaluated within the context of a specific programming course and several student classes.



5.6 Meerdere parameters of argumenten gebruiken

Er zijn ook functies en methoden met meer dan één parameter. Voorbeelden daarvan zijn datetime(2019, 9, 1) en range(1,50,10). Bij datetime() geef je drie integers op: het jaar, de maand en de dag. Bij range kun je kiezen voor twee of drie argumenten:

```
for i in range(1,45):
      print(i)
for i in range(1,45,10):
      print(i)
```

In de eerste for-loop worden de getallen 1 tot aan 45 geprint. In de tweede for-loop alleen de getallen 1, 11, 21, 31 en 41. Het derde argument bij de tweede for-loop is optioneel. Je kunt het opgeven, maar het is niet verplicht. Je kunt ook zeggen dat de standaardwaarde (Engels: default value) van het derde argument 1 is.

De functie range() heeft wel drie parameters: die zou je de namen beginwaarde, eindwaarde en stapgrootte kunnen geven. In dit geval ben je echter niet verplicht om drie argumenten op te geven: dat mogen er ook twee zijn. Maar als je de beginwaarde standaard wilt houden, maar wel de stapgrootte wilt wijzigen, moet je toch ook de beginwaarde opgeven. Anders weet Python niet dat je 1e getal bij de 2e parameter hoort, en je 2e getal bij de 3e parameter.

Let bij het schrijven van een functie goed op dat je de parameters duidelijke namen geeft, zodat je deze in de functie makkelijk kan gebruiken.

Opdracht 11 >

Figure 3.3: The learning platform, taken from [29]

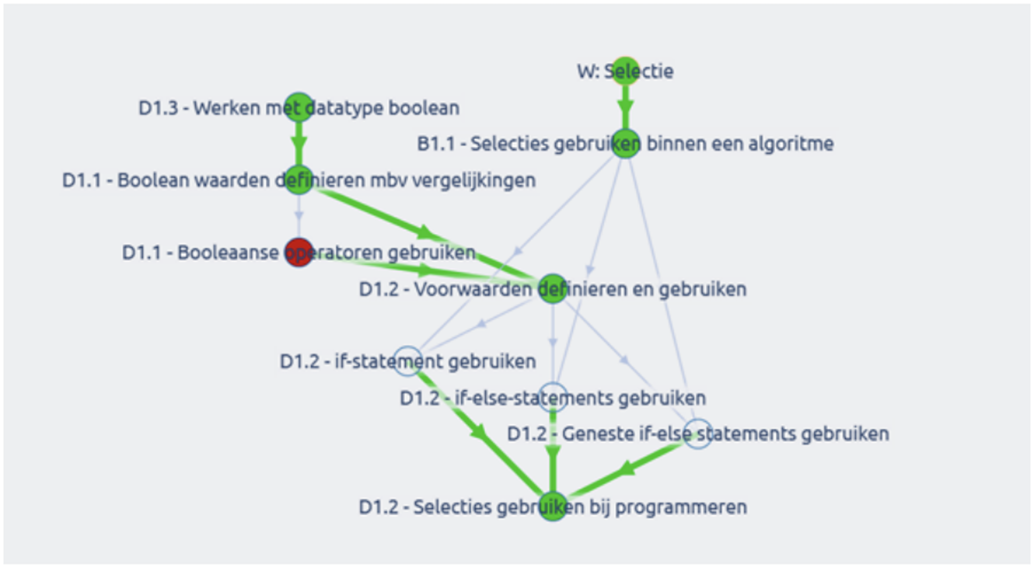

W: Selectie
D1.3 - Werken met datatype boolean
B1.1 - Selecties gebruiken binnen een algoritme
D1.1 - Boolean waarden definieren mbv vergelijkingen
D1.1 - Booleaanse operatoren gebruiken
D1.2 - Voorwaarden definieren en gebruiken
D1.2 - if-statement gebruiken
D1.2 - if-else-statements gebruiken
D1.2 - Geneste if-else statements gebruiken
D1.2 - Selecties gebruiken bij programmeren

Figure 3.4: Example of the learning goals for a specific chapter, taken from [29]

### 3.3.2 iSport

As part of his Masters thesis [31] under the supervision of Beat Signer, Yoshi Malaise presented the iSport framework. After having surveyed multiple table tennis trainers about their way of managing their training sessions, he found that the methods used by most of the trainers were not optimal and could be improved. For instance, the majority of trainers write down their session preparation on paper and do not use any form of ICT tools for it, which makes it difficult to first of all keep track of the trainees progress but also to share the session details with other coaches, as it is often the case that trainees are trained by different coaches. Therefore, in his research, he looked at how well the concepts of knowledge graphs and personalised learning could be applied in the context of table tennis training. In Figure 3.5 the iSport framework, which is based on the Resource-Selector-Link metamodel by Signer and Norrie [45], is illustrated. The framework is a knowledge graph representing the field-specific knowledge of table tennis education that may help the trainers mainly in managing the training sessions and in monitoring the trainees' progresses. For example, based on the graph it can be deduced that an exercise may be used to teach certain techniques but also that an exercise can have some prerequisites in terms of previously acquired techniques. In addition, Malaise developed a tool built on top of the iSport framework to be used by the trainers. The tool consists of different modules where among other things sessions can be scheduled and planned based on saved exercises, assessments can be created and user profiles of the trainees can be visualised including the knowledge graph that keeps track of their progress. As the prototype was built during the Covid-19 pandemic, the evaluation had to be done using a video demonstrating how the tool may be used and a survey. However, the majority of trainers were enthusiastic about using the tool in the future.

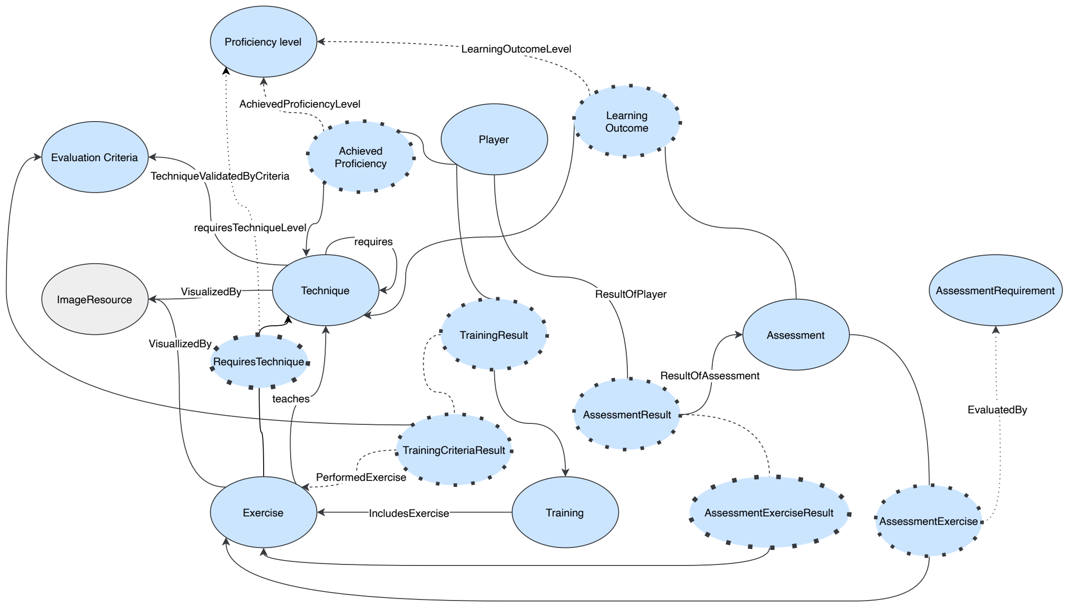


Figure 3.5: The iSport framework, taken from [31]

# Chapter 4

# JsStories

We introduce the JsStories tool as a solution to foster social inclusion in computer science education, particularly within organisations like HYFBE, which aim to teach computer science to socially vulnerable groups. The purpose of the tool is to help students learn the JavaScript programming language. Storytelling, the PRIMM approach and the concept of knowledge graphs form the core foundations of the tool.

## 4.1 JsStories

As previously mentioned in Section 1.2, we had the opportunity to visit HYFBE during a Sunday session and engage with the educational coordinator, as well as the students following the web development programme. When we asked the students what they found the most challenging in the curriculum, the first *JavaScript* module was the most popular answer, followed by the module on *separation of concerns*. In the program, the students begin with learning markup languages like markdown and HTML, as well as styling with CSS. Following this introduction, in the first *JavaScript* module, the students move on to a real programming language and encounter many constructs of programming such as variables, conditionals and loops for the first time. Furthermore, during our discussion with the educational coordinator, he confirmed that these were the two modules students struggle the most with. However, he explained that for *separation of concerns* this is mostly related to the curriculum itself, in terms of when the subject is introduced, how long it is taught and the teaching methods utilised. This was already something they were planning and working on adapting in their curriculum. As a result, we decided to focus the tool on the JavaScript programming language.

### 4.1.1 Storytelling

As discussed in Section 2.7 and 2.8, storytelling can enhance adult learning depending on the learner's interest, background and experience, as well as the kind of stories used. In addition to improving knowledge retention, it boosts students' engagement and involvement depending on how they can relate to the story and its characters. Therefore, JsStories was designed with storytelling at its core. The goal of the tool is to help students learn programming in JavaScript by working through digital books that contain stories and exercises. The stories included are real pseudonymised stories from HYFBE alumni, detailing their journeys to Belgium, the challenges they faced, their HackYourFuture experience and any insights they have learned along the way. As such, the selected stories aim to foster the sense of belonging among learners who can relate to the main characters. Moreover, reading such achievement stories can inspire and motivate the students to complete the program, potentially reducing the drop out rates in the long run. Figure 4.1 depicts the homepage of the JsStories tool, which currently includes four different stories. As one can see from the illustration, progress bars are also used for each book as they allow students to monitor their progress and serve as additional incentive to complete the book, as outlined in Section 2.7. Additionally, when students have completed all the JsStories, they can generate and download a personalised certificate including their name. This is illustrated in Figures 4.8g and 4.8h. Therefore, before being able to use the tool, students have to sign up with their first and last name as shown in Figure 4.8a.

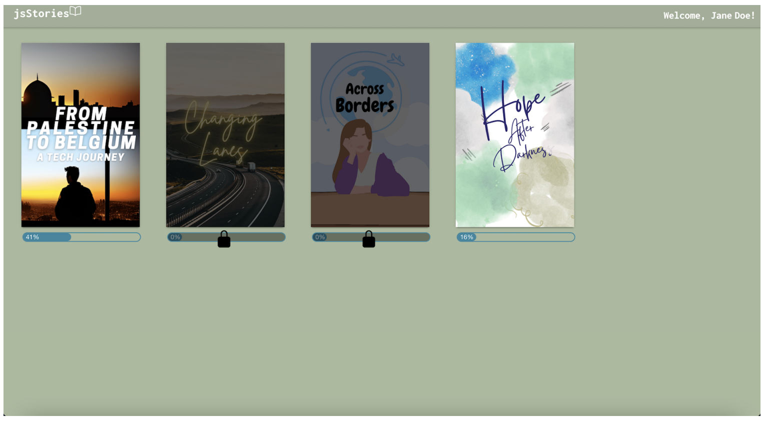
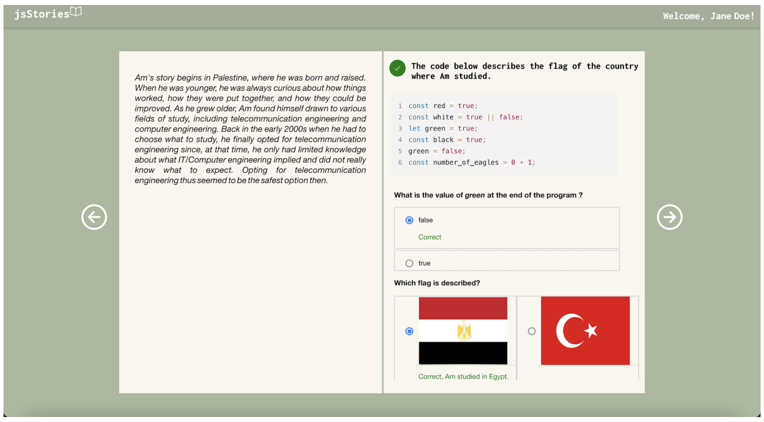


Figure 4.1: Homepage of the JsStories tool

### 4.1.2 PRIMM

The incorporated exercises are based on the PRIMM model, which consists of dividing the programming lessons and learning materials into five phases: (1) Predict, (2) Run, (3) Investigate, (4) Modify and (5) Make. As discussed in Section 2.3.3, this model enhances the understanding of students as they progress through each of these phases and thus ensures that students are not required to write code before being able to read it. Hence, the JsStories tool contains exercises for each of these phases. For the predict phase, multiple choice questions requiring the learner to predict the outcome of code have been included. These questions feature options that may contain images as well as text. Figure 4.2 illustrates an example of such a predict exercise. In this particular exercise, the given code describes a flag and the student has to predict which flag it represents out of a set of given options.



*Am's story begins in Palestine, where he was born and raised. When he was younger, he was always curious about how things worked, how they were put together, and how they could be improved. As he grew older, Am found himself drawn to various fields of study, including telecommunication engineering and computer engineering. Back in the early 2000s when he had to choose what to study, he finally opted for telecommunication engineering since, at that time, he only had limited knowledge about what IT/Computer engineering implied and did not really know what to expect. Opting for telecommunication engineering thus seemed to be the safest option then.*

**The code below describes the flag of the country where Am studied.**

```
const red = true;
const white = true || false;
let green = true;
const black = true;
green = false;
const number_of_eagles = 0 + 1;
```

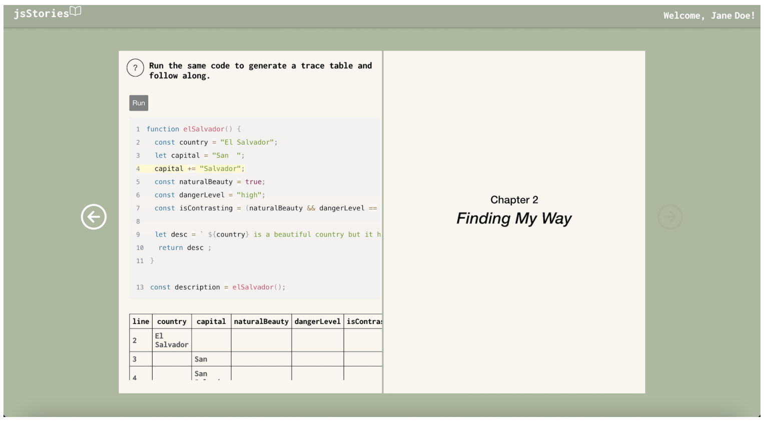
**What is the value of *green* at the end of the program ?**

- [✓ false] Correct
- true

**Which flag is described?**

Correct, Am studied in Egypt.

Figure 4.2: Example of a predict exercise



**Run the same code to generate a trace table and follow along.**

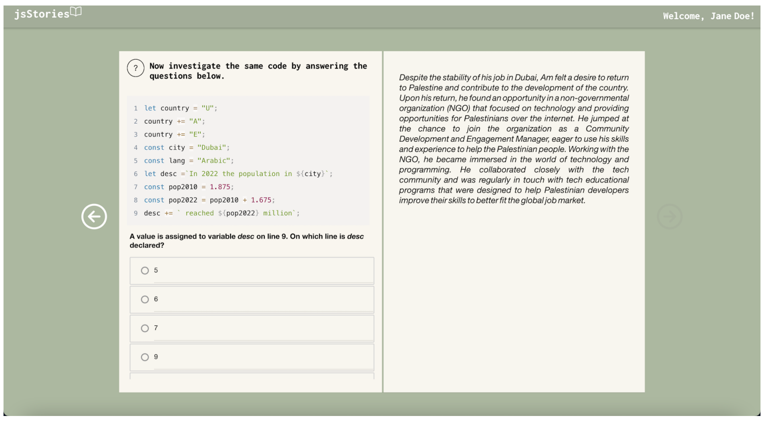
```
1 function elSalvador() {
2   const country = "El Salvador";
3   let capital = "San  ";
4   capital += "Salvador";
5   const naturalBeauty = true;
6   const dangerLevel = "high";
7   const isContrasting = (naturalBeauty && dangerLevel ==
8
9   let desc = ` ${country} is a beautiful country but it h
10   return desc ;
11 }

13 const description = elSalvador();
```

| line | country | capital | naturalBeauty | dangerLevel | isContras |
|---|---|---|---|---|---|
| 2 | El Salvador | | | | |
| 3 | | San | | | |
| 4 | | San | | | |

Chapter 2

*Finding My Way*

Figure 4.3: Example of a run exercise



**Now investigate the same code by answering the questions below.**

```
1 let country = "U";
2 country += "A";
3 country += "E";
4 const city = "Dubai";
5 const lang = "Arabic";
6 let desc =`In 2022 the population in ${city}`;
7 const pop2010 = 1.875;
8 const pop2022 = pop2010 + 1.675;
9 desc += ` reached ${pop2022} million`;
```

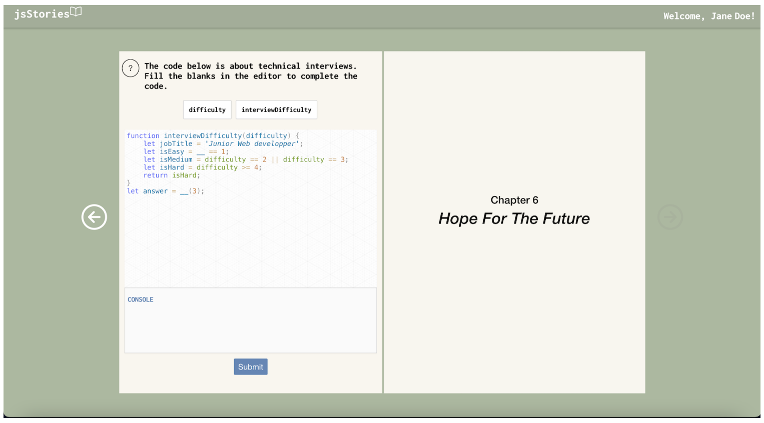
**A value is assigned to variable *desc* on line 9. On which line is *desc* declared?**

- 5
- 6
- 7
- 9

*Despite the stability of his job in Dubai, Am felt a desire to return to Palestine and contribute to the development of the country. Upon his return, he found an opportunity in a non-governmental organization (NGO) that focused on technology and providing opportunities for Palestinians over the internet. He jumped at the chance to join the organization as a Community Development and Engagement Manager, eager to use his skills and experience to help the Palestinian people. Working with the NGO, he became immersed in the world of technology and programming. He collaborated closely with the tech community and was regularly in touch with tech educational programs that were designed to help Palestinian developers improve their skills to better fit the global job market.*

Figure 4.4: Example of an investigate exercise

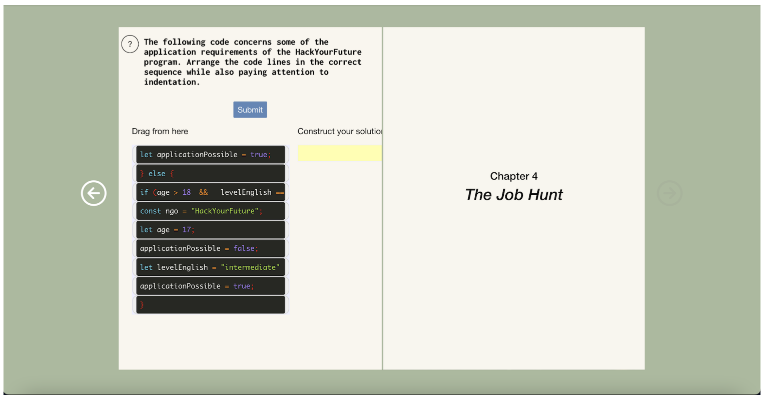
Next, in the run phase, the students are required to run the provided code and follow along an automatically filled-in trace table. Each line in the execution gets highlighted and the student can observe the changes in the trace table for that specific line. This type of exercise was included because, as discussed in Section 2.2, the abilities to trace, explain and write code are closely related. Tracing code, i.e. simulating the execution of code, improves the understanding of it. An example of this exercise is shown in Figure 4.3. As one can see, the fourth line of the code, currently being executed, is highlighted and in the trace table the student can observe the value assigned to the variable `capital`. In the investigate phase of PRIMM, the students' focus should be on understanding code structure. Therefore, exercises involving multiple choice questions requiring the learner to investigate the structure of code have been included. These questions cover, among others, topics such as variable declaration and initialisation. Figure 4.4 depicts an example.



The code below is about technical interviews. Fill the blanks in the editor to complete the code.

```
function interviewDifficulty(difficulty) {
    let jobTitle = 'Junior Web developper';
    let isEasy = __ == 1;
    let isMedium = difficulty == 2 || difficulty == 3;
    let isHard = difficulty >= 4;
    return isHard;
}
let answer = __(3);
```

Figure 4.5: Example of a modify exercise: blanks



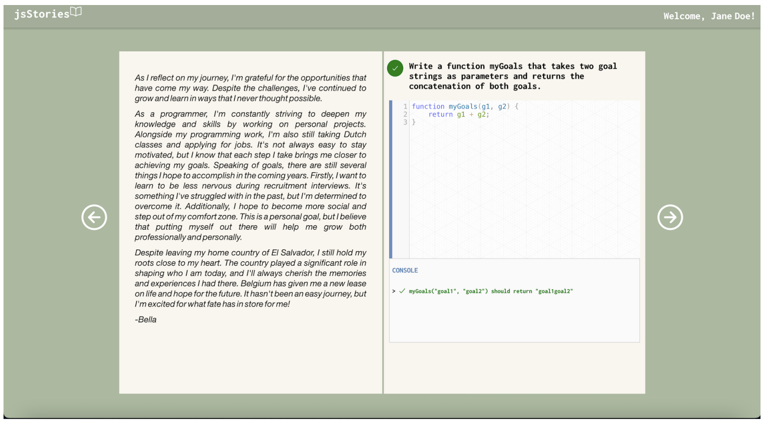
The following code concerns some of the application requirements of the HackYourFuture program. Arrange the code lines in the correct sequence while also paying attention to indentation.

```
let applicationPossible = true;
} else {
if (age > 18  &&   levelEnglish ==
const ngo = "HackYourFuture";
let age = 17;
applicationPossible = false;
let levelEnglish = "intermediate"
applicationPossible = true;
}
```

Figure 4.6: Example of a modify exercise: parsons

Furthermore, for the modify phase, where students finally get to modify code, two types of exercises were incorporated. The first type is a blanks exercise, in which students are given some code containing blanks and options to fill them in. These options can for instance include keywords, variable names and values. An example of this exercise is shown in Figure 4.5. The second type of exercise is a parsons puzzle, which has been shown to be an effective exercise for learning programming, as described in Section 2.6. More precisely, these exercises include two-dimensional parsons puzzles, which require students to put code fragments in the right order while taking into consideration indentation. Figure 4.6 illustrates such a puzzle.

Finally, in the exercises for the make phase, the students are required to write code, which is then checked using predefined unit tests. They are given a description of the required functionality and have to write code accordingly. For instance, in the exercise depicted in Figure 4.7, the students are tasked to write a function with a specific name, number of arguments and return value.



*As I reflect on my journey, I'm grateful for the opportunities that have come my way. Despite the challenges, I've continued to grow and learn in ways that I never thought possible.*

*As a programmer, I'm constantly striving to deepen my knowledge and skills by working on personal projects. Alongside my programming work, I'm also still taking Dutch classes and applying for jobs. It's not always easy to stay motivated, but I know that each step I take brings me closer to achieving my goals. Speaking of goals, there are still several things I hope to accomplish in the coming years. Firstly, I want to learn to be less nervous during recruitment interviews. It's something I've struggled with in the past, but I'm determined to overcome it. Additionally, I hope to become more social and step out of my comfort zone. This is a personal goal, but I believe that putting myself out there will help me grow both professionally and personally.*

*Despite leaving my home country of El Salvador, I still hold my roots close to my heart. The country played a significant role in shaping who I am today, and I'll always cherish the memories and experiences I had there. Belgium has given me a new lease on life and hope for the future. It hasn't been an easy journey, but I'm excited for what fate has in store for me!*

*-Bella*

**Write a function myGoals that takes two goal strings as parameters and returns the concatenation of both goals.**

```
function myGoals(g1, g2) {
    return g1 + g2;
}
```

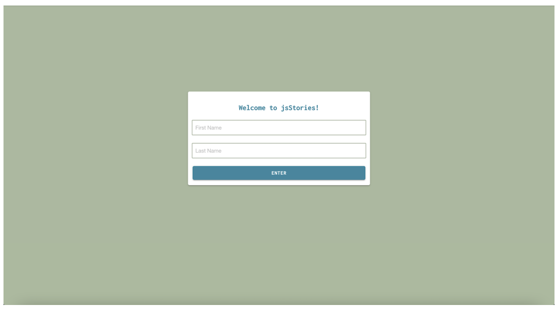
```
CONSOLE

> ✓ myGoals("goal1", "goal2") should return "goal1goal2"
```

Figure 4.7: Example of a make exercise

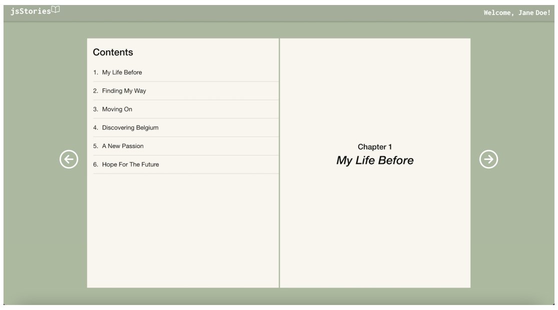
The exercises in the JsStories tool serve as a form of formative assessment (see Section 2.5). The small ungraded tasks include feedback which helps students identify gaps in their understanding. The requirements for effective feedback, discussed in Section 2.7, have been taking into consideration. As such, the feedback is provided immediately after the students submit their answers and it clearly indicates any mistake depending on the kind of exercise. For the predict and investigate exercises, textual feedback is provided for both correct and incorrect answers. The feedback is indicated in green and red respectively and explains why it is correct or incorrect. Additionally, for the exercises of the modify phase, the code lines which have been incorrectly filled in or placed in the puzzle are highlighted in red, helping students identify their mistakes. Furthermore, in the parsons puzzles, textual feedback is also provided indicating whether code lines have been incorrectly placed or incorrectly indented. Lastly, in the make exercises, students can view which unit tests their code has successfully passed and which it hasn't, providing an indication of what might be incorrect in their code.

### 4.1.3 Content of a JsStory

Every JsStory is displayed on the home page using a cover image and title, as depicted in Figure 4.1. A JsStory can consist of several chapters and an overview of all the chapters can be found on the book's contents page, as shown in Figure 4.8b. Moreover, every chapter is composed of multiple scenes including text and images to convey parts of the story as well as one or more of the aforementioned exercises. These exercises are integrated into the story itself, offering additional information or addressing topics covered in the story. Finally, each JsStory concludes with a motivating quote related to the story, as shown in Figure 4.8c. The appendix contains a complete example of a JsStory, where the entire story can be explored along with the included exercises. Moreover, it contains a larger version of the screenshots in Figure 4.8.



(a) The sign up page

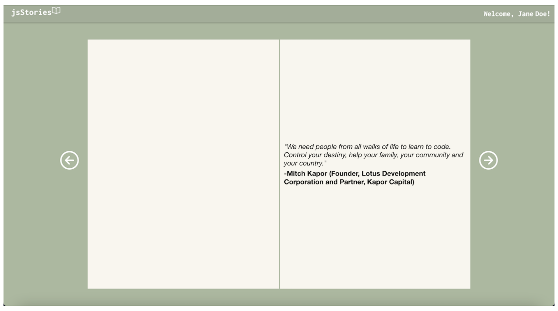


Contents

1. My Life Before
2. Finding My Way
3. Moving On
4. Discovering Belgium
5. A New Passion
6. Hope For The Future

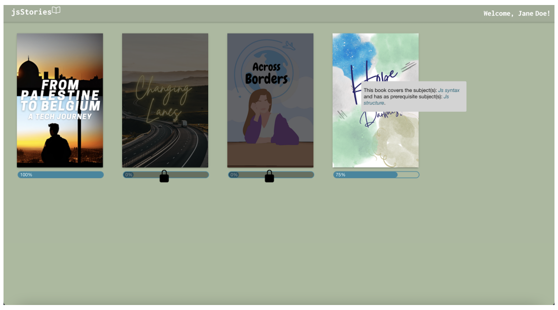
Chapter 1
*My Life Before*

(b) The contents page in a book



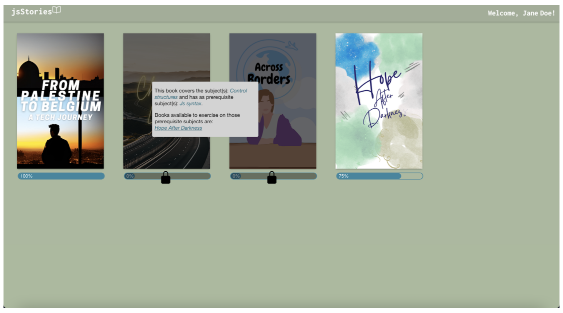
*"We need people from all walks of life to learn to code. Control your destiny, help your family, your community and your country."*
**-Mitch Kapor (Founder, Lotus Development Corporation and Partner, Kapor Capital)**

(c) The quote at the end of a book



This book covers the subject(s): *Js syntax* and has as prerequisite subject(s): *Js structure*.

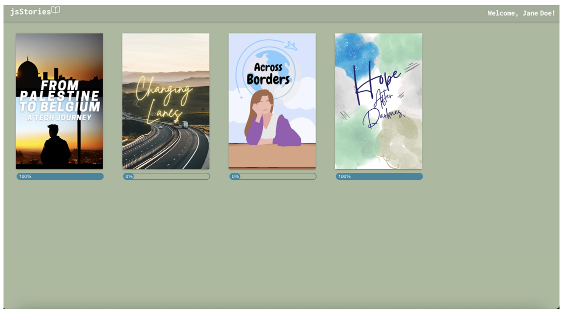
(d) Tooltip info of an unlocked book



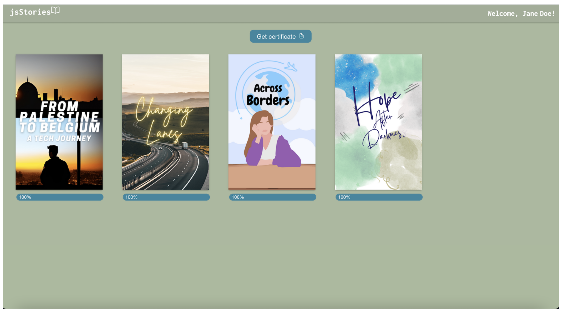
This book covers the subject(s): *Control structures* and has as prerequisite subject(s): *Js syntax*.

Books available to exercise on those prerequisite subjects are:
*Hope After Darkness*

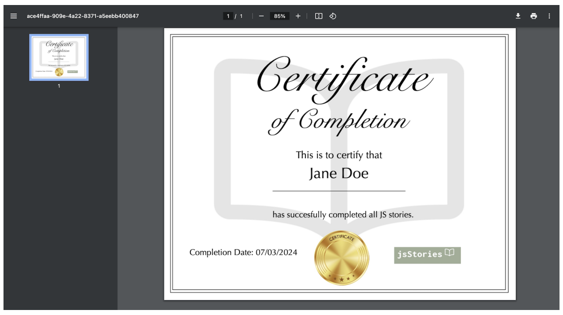
(e) Tooltip info of a locked book



(f) All books being unlocked



(g) Button to generate a certificate



Certificate of Completion

This is to certify that
Jane Doe

has succesfully completed all JS stories.

Completion Date: 07/03/2024

(h) The generated certificate

Figure 4.8: Screen captures of the JsStories tool

### 4.1.4 The Knowledge Graph

One of the suggestions for enhancing programming teaching, as discussed in the problem statement (see Section 1.1), is to provide a flexible learning environment that enables students to learn at their own pace. In the related works Section 3.3.1 and 3.3.2, a personalised learning environment was created making use of the concept of a knowledge graph, described in Section 2.9. Hence, a JavaScript knowledge graph was integrated as the third core component of the JsStories tool.

The knowledge graph used in the JsStories tool is illustrated in Figure 4.9. Initially, the goal was to build a general knowledge graph based on the JavaScript programming language. However, doing so was much more challenging than anticipated. After discussing it with Laura Van der Lubbe, who contributed to the Python knowledge graph described in Section 3.3.1, the decision was made to follow her advice on limiting the scope and instead building the knowledge graph based on a particular curriculum. She noted that the prior knowledge differs in a lot of handbooks, curricula, tutorials and other resources. As a result, creating a general knowledge graph would be difficult as the prior knowledge defined in the graph would never be in line with the prior knowledge stated in all these materials. The resulting knowledge graph is therefore built with the curriculum of HYFBE's web development programme as its foundation. The curriculum, which is open source and available online on GitHub[1], was thoroughly analysed in order to build this knowledge graph.

From the curriculum's GitHub repository, six different modules were deduced. These were coloured differently in the knowledge graph: Agile (blue), User experience and User interface (red), Welcome to JavaScript (green), Debugging (yellow), Behaviour, strategy and implementation (purple), and Separation of concerns (grey). Every subject covered in the curriculum is represented by an oval figure in the knowledge graph. As is illustrated in the knowledge graph, a module may cover multiple subjects. Furthermore, the knowledge graph is a directed acyclic graph in which the edges determine the prior knowledge required for a certain subject, indicating which other subjects must be completed first before moving on to this subject. On the GitHub page, emojis illustrating the growth phases of a chicken are used for each module, helping students to keep track of their progress but most importantly, indicating what material or exercises they should prioritise. For instance, material marked with an egg cover the module's fundamental skills, whereas a chicken emoji denotes material covering more advanced concepts and skills. Students should thus make sure to understand the material marked with an egg before moving on to materials marked with the next emoji in the growth phase. This approach was analysed and used to determine the prior knowledge in the graph. Additionally, a subject may be split into multiple sub-subjects according to the curriculum. As the purpose of the JsStories tool is to learn how to program in JavaScript, we decided to only focus on the subjects covered in the module 'Welcome to JS' and related modules in terms of prior knowledge. These subjects with their sub-subjects are listed below in Table 4.1.

[1] `https://github.com/HackYourFutureBelgium`

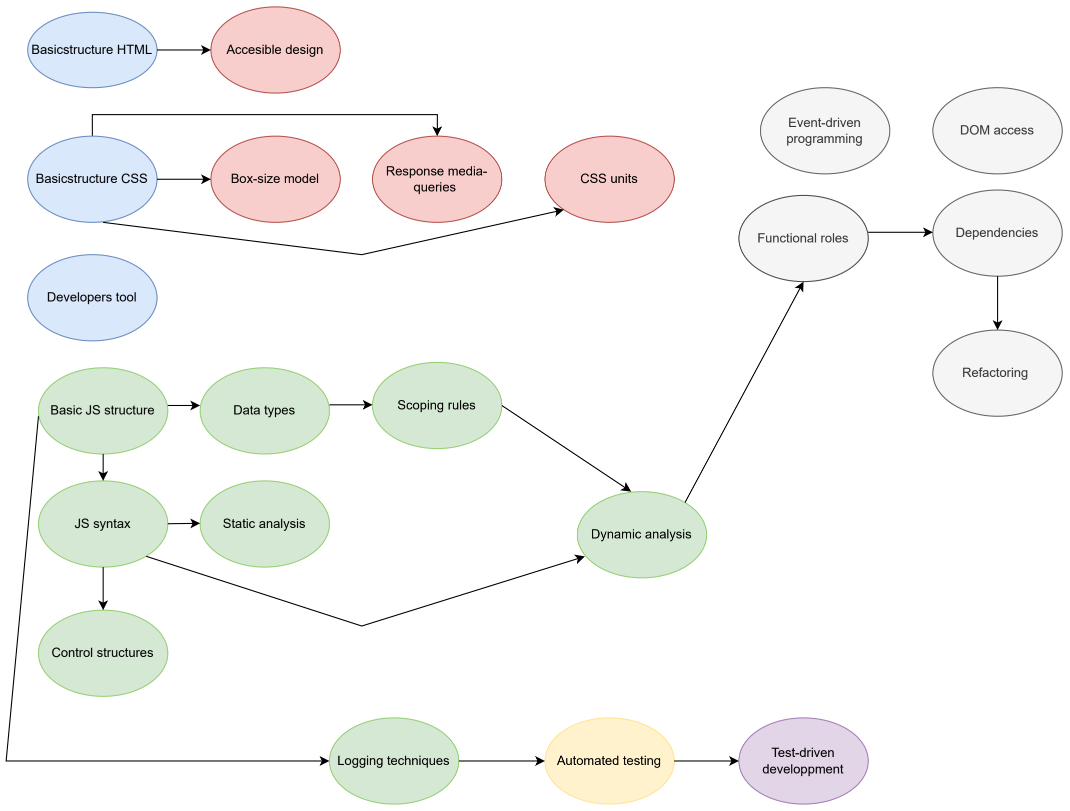


Figure 4.9: The knowledge graph based on the curriculum of the web development program of HYFBE

| **Subject** | **Module** | **Sub-subjects** |
|---|---|---|
| Basic JavaScript structure | Welcome to JavaScript | Understanding the program structure, Using semicolon at the end of a code line |
| JavaScript syntax | Welcome to JavaScript | Primitives, Operators, Identifiers, Keywords, Checks, Blocks, Function calls |
| Control Structures | Welcome to JavaScript | If-else, While, Do-while, Switch, For-loop |
| Data types | Welcome to JavaScript | string, number, bigint, boolean, undefined, null, symbol, object |
| Static analysis | Welcome to JavaScript | Flowchart, Analysing variables, Annotating code |
| Scoping rules | Welcome to JavaScript | var vs. let, Dynamic vs. static scoping, Lifetime |
| Logging techniques | Welcome to JavaScript | console.log, console.error, console.info, alert, prompt |

| | | |
|---|---|---|
| Dynamic analysis | Welcome to JavaScript | Tracetables, Variable access, Variable types |
| Automated testing | Debugging | console.assert, Debugger statement |
| Test-driven development | Behaviour, strategy and implementation | Documenting behaviour, Writing tests, Implement functions based on tests, Fuzz testing |
| Functional roles | Separation of concerns | Listeners, Handlers, Utils, Components, Custom events |
| Dependencies | Separation of concerns | Entry points, import, NPM packages, Scope hierarchy |
| Refactoring | Separation of concerns | Code splitting, Isolating components, Component unit tests, Single responsibility pattern |
| Event-driven programming | Separation of concerns | Event listeners, Event handlers, Event, Program flow, Event loop |
| DOM access | Separation of concerns | find element by id, find element by classname, find element by type, element.innerHTML, element.attribute, document.createElement, document.appenChild, document.removeChild |

Table 4.1: Covered subjects split into sub-subjects.

**Knowledge Graph Update**

Every exercise within a story covers specific aspects of the JavaScript programming language, corresponding to one or more sub-subjects in the knowledge graph. Upon completion of an exercise, the relevant sub-subjects are updated in the knowledge graph and the learner can move on with the story. A sub-subject is considered complete once the learner has successfully finished at least one exercise covering that sub-subject for all the different PRIMM phases (see Section 4.1.2). For instance, the sub-subject *Primitives* is considered complete in the knowledge graph once the learner has completed at least once all of the following exercises related to *Primitives*: a predict exercise, a run exercise, an investigate exercise, either a blanks exercise or parsons puzzle and a make exercise. Furthermore, a subject in the knowledge graph is considered complete when all of its sub-subjects have been completed. For instance, the subject *JS syntax* is marked complete once all its sub-subjects (*Primitives*, *Operators*, *Identifiers*, *Keywords*, *Checks*, *Blocks* and *Function calls*) are marked complete. A challenging design decision was whether to mark a sub-subject as complete once the learner has completed at least one exercise addressing that sub-subject for all the different PRIMM phases or once all available exercises covering that sub-subject have been completed. We finally opted for the first option as this allows learners to choose whether to further exercise on that sub-subject by reading other stories covering that sub-subject or to move on and continue with stories covering other same-level sub-subjects or more advanced subjects.

#### Story Locking

As depicted in Figure 4.8d, stories can be locked or unlocked based on the subjects they cover and the prior knowledge required. When hovering on unlocked books, the tooltip indicates which subjects it covers and the prior knowledge it requires in terms of mastered subjects. For locked books, the tooltip provides additional information about the books the learner must read first to acquire the prior knowledge needed to unlock that story, as demonstrated in Figure 4.8e. To illustrate with an example, consider Figure 4.8d where one can see that the books *From Palestine to Belgium: A Tech Journey* and *Hope After Darkness* are unlocked while the books *Changing Lanes* and *Across Borders* are locked. Moreover, Figure 4.8e shows that the book *Changing Lanes* covers the subject *Control structures* and that it has as prerequisite subject *JS Syntax*, which can be acquired by first reading the book *Hope After Darkness.* As soon as the learner completes the book *Hope after darkness*, the knowledge graph is updated and consequently the book *Changing Lanes* is unlocked since the necessary prior knowledge has been acquired. As such, the knowledge graph is used to create a personalised learning environment allowing the learner to learn at its own pace. Furthermore, with the locking of stories, the learning way is paved for the learners allowing them to complete exercises that are in line with their current knowledge level, minimising the frustration caused by exercises that are too difficult based on their current level.

# Chapter 5

# Implementation

This chapter provides further details on the implementation of the JsStories tool which was described in the previous chapter. Furthermore, details about the collection and writing processes of the stories used in the tool are also provided in this chapter.

## 5.1 Story Collection and Processing

The used stories were collected from HYFBE alumni who volunteered to share their story. These participants were contacted using the programme's Slack platform. A total of five people volunteered but after contacting them individually, only four of them replied. The interviews were scheduled using the Vectera[1] platform. This allowed the interviews to be easily scheduled at a time that worked best for the participants. Each interview started with an introduction to the research and a clear explanation on how their stories would be used. The interviews were not structured with pre-planned questions, allowing the participants to freely tell their stories. This was done to avoid monotonic stories and encourage participants to share what was most relevant to them. To ensure accurate understanding and avoid any misinterpretations, the LSD method [16] was used during the interviews. This involved listening to the participants, summarising what they were saying, and asking relevant questions based on what was being shared. Every interview was recorded after obtaining permission from the participants and the recordings were used to transcribe the stories later on. The transcribed stories were then summarised in bullet points and converted to narrative text using ChatGPT[2]. Furthermore, all stories were pseudonymised to maintain the privacy of the participants.

[1]`https://go.vectera.com`
[2]`https://chat.openai.com/`

## 5.2 The Application

The high-level architecture of the JsStories tool is depicted in Figure 5.1. The JsStories tool was developed using the TypeScript[3] programming language, which is essentially a strongly typed language that is built on top of JavaScript. Hence, TypeScript was used because of its compile-time type checking, as well as its compatibility with the Ionic framework[4]. Ionic is an open-source framework that is especially popular for the development of mobile apps, as it allows cross-platform app development. It is known to be easy to learn because it mainly requires basic web development skills. Furthermore, Ionic provides built-in UI components compatible with various frontend frameworks. Initially, we wanted to keep the possibility to create a mobile app in the future open, should it be deemed useful and necessary. Therefore, Ionic, integrated with the component-based frontend framework React JS[5], was utilised. The JsStories tool is currently available as a desktop application. It was packaged and distributed using the Electron framework[6].

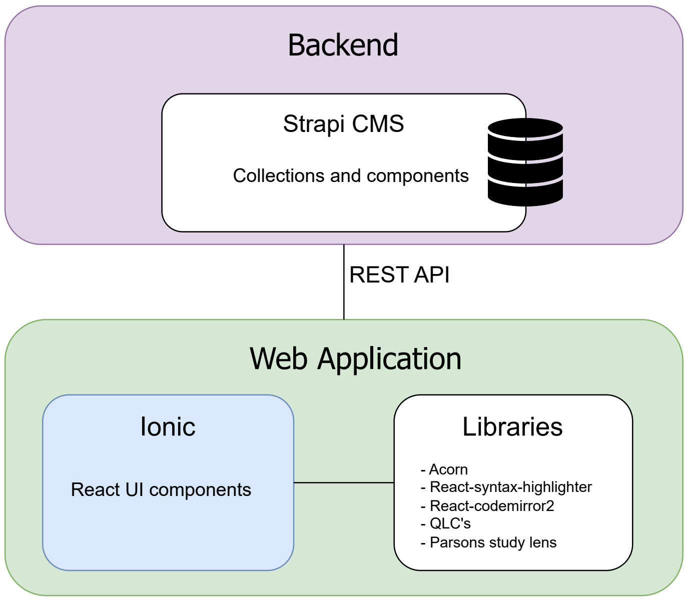
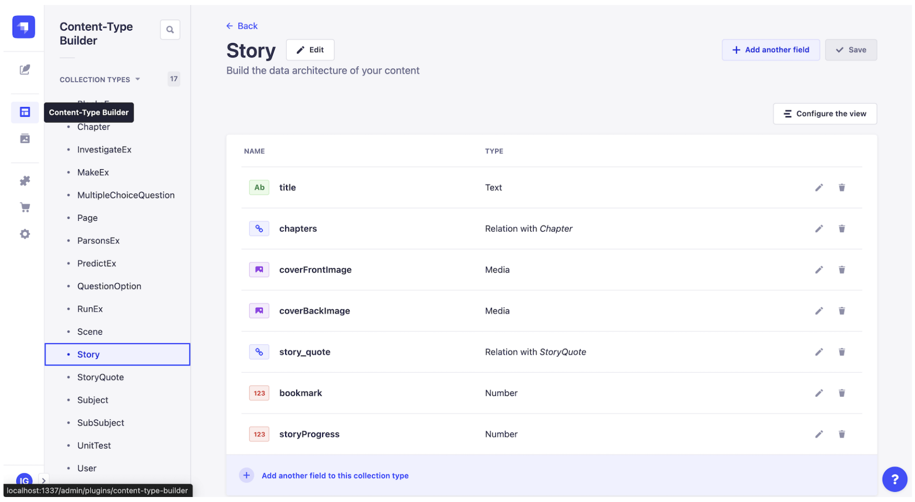


Figure 5.1: High level architecture of the application

More importantly, since the tool is to be used in an educational setting, including teachers and organisational personnel, who may not have a technical background, it should be easy to manage and adapt stories and exercises. Hence, it was decided to separate the content, specifically the stories and the exercises, from the more technical aspects of the tool. Therefore, Strapi[7], an open source headless Content Management System (CMS), was used. The content can be managed through both a REST API and an admin panel, which is depicted in Figures 5.2 and 5.3. Strapi supports three types of content: collections, single types and components. A single type can only

[3]https://www.typescriptlang.org/
[4]https://ionicframework.com/
[5]https://react.dev/
[6]https://www.electronjs.org/
[7]https://strapi.io/

have one instance, whereas multiple instances of a collection can be created. Finally, components are data structures that can be used as attributes within the other two content types.



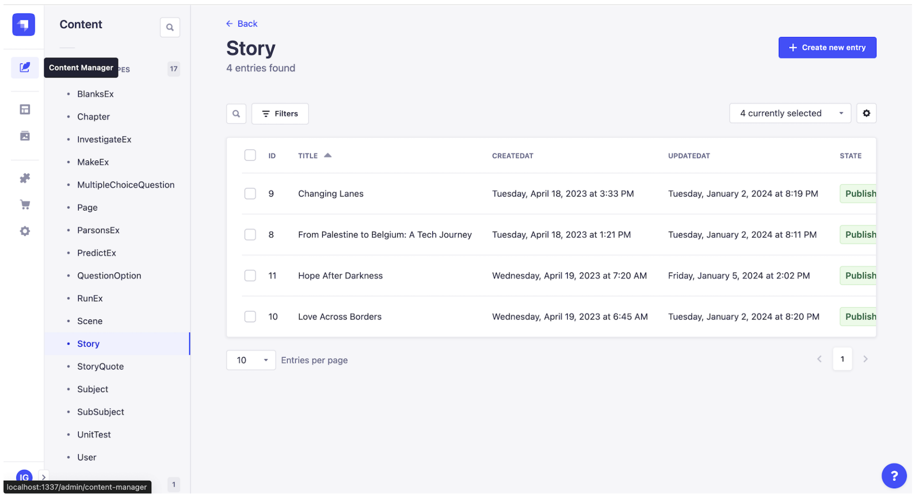
**Story**

Build the data architecture of your content

| NAME | TYPE |
|---|---|
| title | Text |
| chapters | Relation with *Chapter* |
| coverFrontImage | Media |
| coverBackImage | Media |
| story_quote | Relation with *StoryQuote* |
| bookmark | Number |
| storyProgress | Number |

Figure 5.2: Strapi's Content-Type Builder

Content types can be created using the Content-Type Builder shown in Figure 5.2. Specifically, the *Story* collection in the Content-Type Builder, is illustrated. This collection is built based on seven attribute fields of various types, such as media fields for the cover images, text for the title and relation fields linking to other content types. For instance, as discussed in the previous chapter, a story consists of multiple chapters, including scenes with text, images and exercises. Hence, the *Story* collection has a relation field *chapters*, representing the one-to-many relationship with the *Chapter* collection. Furthermore, instances of the different content types may be created and managed in the Content Manager, depicted in Figure 5.3. As one can see, four instances of the *Story* collection have been created.



**Story**

4 entries found

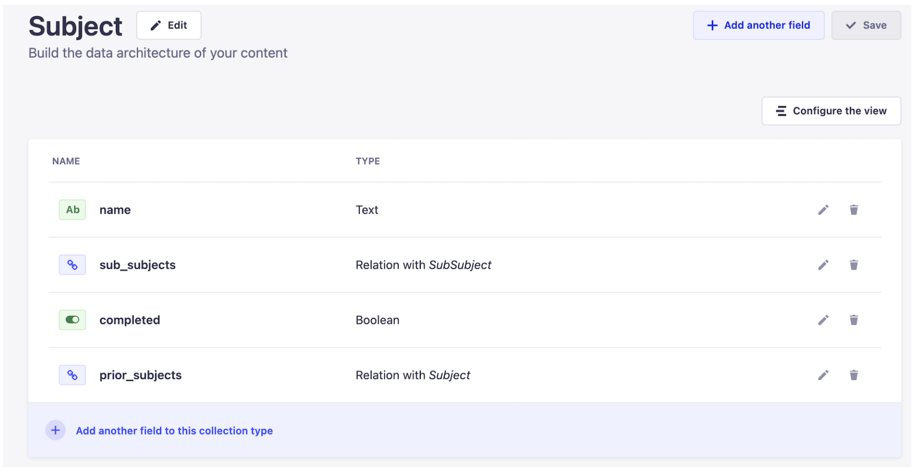
| ID | TITLE | CREATEDAT | UPDATEDAT | STATE |
|---|---|---|---|---|
| 9 | Changing Lanes | Tuesday, April 18, 2023 at 3:33 PM | Tuesday, January 2, 2024 at 8:19 PM | Publish |
| 8 | From Palestine to Belgium: A Tech Journey | Tuesday, April 18, 2023 at 1:21 PM | Tuesday, January 2, 2024 at 8:11 PM | Publish |
| 11 | Hope After Darkness | Wednesday, April 19, 2023 at 7:20 AM | Friday, January 5, 2024 at 2:02 PM | Publish |
| 10 | Love Across Borders | Wednesday, April 19, 2023 at 6:45 AM | Tuesday, January 2, 2024 at 8:20 PM | Publish |

Figure 5.3: Strapi's Content Manager

The following sections provide further detail on the backend and frontend implementation of the exercises and the knowledge graph.

### 5.2.1 The Knowledge Graph

The knowledge graph has also been implemented using Strapi. As such, adaptations to the curriculum can be easily reflected using Strapi's Content Manager.

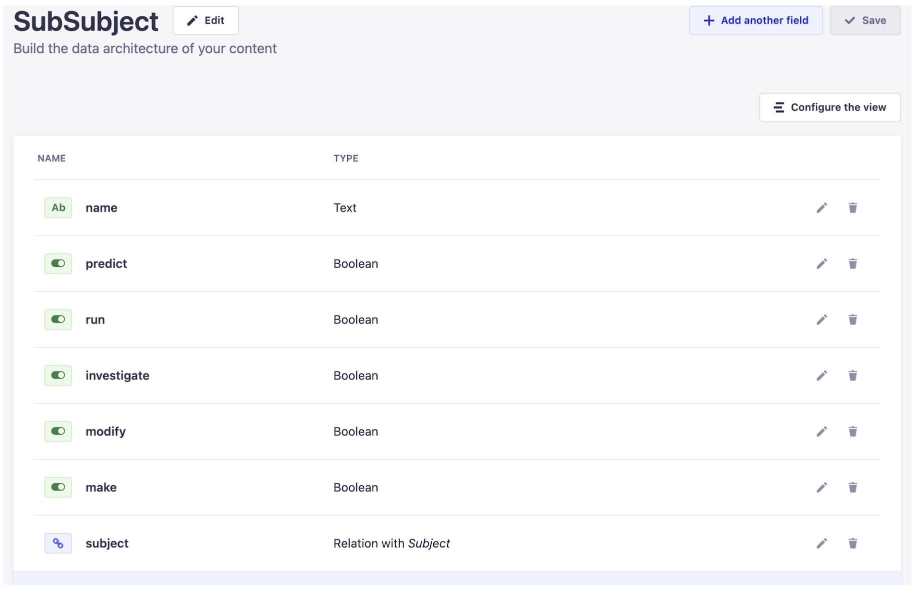


Subject

Build the data architecture of your content

| NAME | TYPE |
|---|---|
| name | Text |
| sub_subjects | Relation with *SubSubject* |
| completed | Boolean |
| prior_subjects | Relation with *Subject* |

Figure 5.4: The Subject collection



SubSubject

Build the data architecture of your content

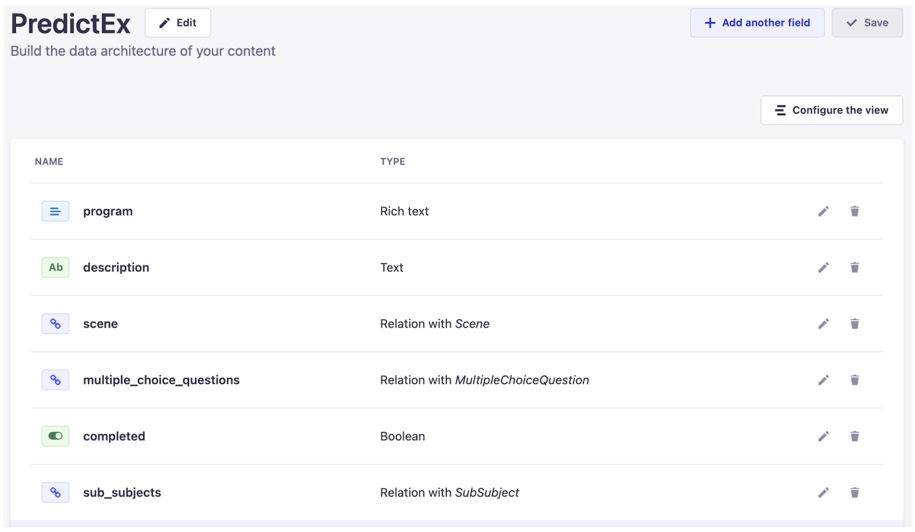
| NAME | TYPE |
|---|---|
| name | Text |
| predict | Boolean |
| run | Boolean |
| investigate | Boolean |
| modify | Boolean |
| make | Boolean |
| subject | Relation with *Subject* |

Figure 5.5: The Subsubject collection

A collection has been created for both subjects and sub-subjects, as described in Section 4.1.4. The *Subject* collection, depicted in Figure 5.4, contains four attribute fields: a name field, a boolean field indicating whether the subject has been completed, a relation field representing the one-to-many relationship with its subsubjects and a many-to-many relation field indicating required prior knowledge in terms of other subjects. Moreover, Figure 5.5 depicts the *Subsubject* collection. Alongside a name attribute field and a relation field to its parent subject, the *Subsubject* collection also contains a boolean field for all the PRIMM phases. Whenever an exercise is completed, the covered subsubjects are updated using the relevant boolean field. As such, a *Subsubject* is considered completed once all of its boolean fields are set to true, indicating that the learner has practiced that subsubject in the five PRIMM phases. Furthermore, the *completed* boolean field of a *Subject* is set to true once all of its subsubjects are considered completed.

### 5.2.2 Predict Exercise

The collection representing a predict exercise, illustrated in Figure 5.6, comprises seven attribute fields. A description of the exercise is stored in the *description* field. Moreover, every predict exercise includes a code fragment, contained in the *program* field, along with some multiple-choice questions, represented by the one-to-many relation field, *multiple_choice_questions*. The multiple-choice questions can contain text options as well as images. As previously mentioned, exercises are contained within scenes, which explains the one-to-one relation with the *Scene* collection. Upon completion of a predict exercise, its *completed* boolean attribute is updated along with the subsubjects it covers, represented by the *sub_subjects* relation field. In terms of User Interface, the React Syntax Highlighter[8] has been used to emphasise code syntax.



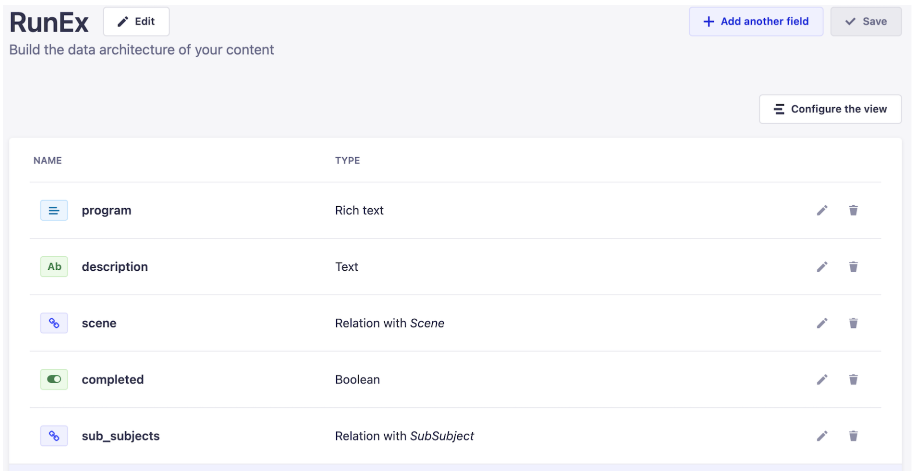
| NAME | TYPE |
|---|---|
| program | Rich text |
| description | Text |
| scene | Relation with *Scene* |
| multiple_choice_questions | Relation with *MultipleChoiceQuestion* |
| completed | Boolean |
| sub_subjects | Relation with *SubSubject* |

Figure 5.6: The predictEx collection

[8]`https://www.npmjs.com/package/react-syntax-highlighter`

### 5.2.3 Run Exercise

Figure 5.7 depicts the *runEx* collection, which consists of the same attribute fields as the *predictEx* collection, without the *multiple_choice_questions* attribute. Among all the exercises, the run exercise was the most challenging to implement. Initially, we attempted to implement it using the JavaScript parser Acorn[9] to generate an abstract syntax tree (AST). Additionally, Acorn-walk[10] was used to traverse the tree and extract all the necessary information to build the trace table. However, we soon realised that this was not the most effective approach, as this requires a lot of manual work with a bigger risk of overlooking details. Therefore, we explored the possibility of using Stopify[11], a JavaScript-to-JavaScript compiler that supports single-stepping and adding breakpoints. Unfortunately, due to a lack of documentation, this was quite challenging. Finally, we adapted code from the Explorotron IDE extension [32], which is based on the Aran[12] library. Aran basically allows to weave advice in the AST of the code, depending on whether the actions are READ or WRITE. Thereafter, the code along with the advice can be converted back to code, enabling evaluation of both together.



| NAME | TYPE |
|---|---|
| program | Rich text |
| description | Text |
| scene | Relation with *Scene* |
| completed | Boolean |
| sub_subjects | Relation with *SubSubject* |

Figure 5.7: The runEx collection

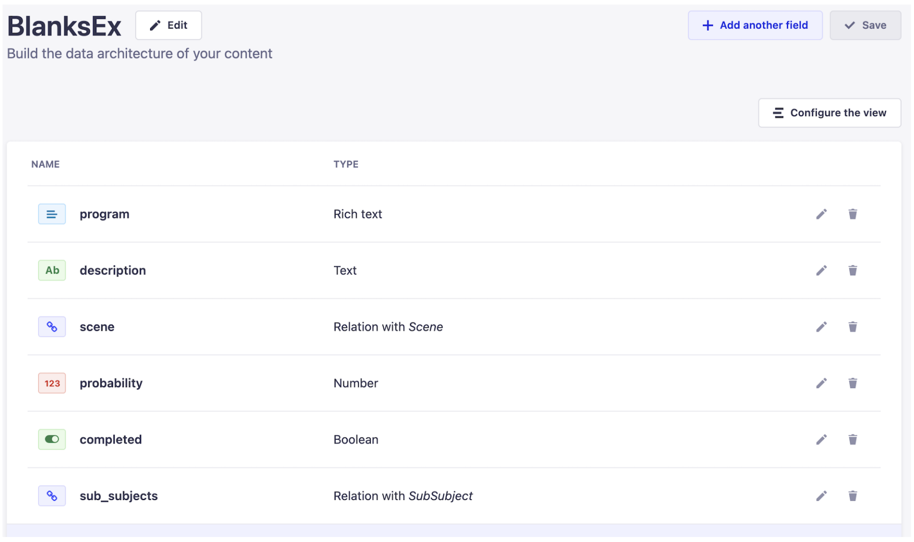
### 5.2.4 Investigate Exercise

The Strapi collection for the investigate exercise contains the exact same set of attribute fields as the *runEx* collection. Furthermore, the open-source library described in Section 3.2, was used to automatically generate questions regarding the structure of the code fragment, stored in the *program* field. The library also provides multiple options per question. Configurations have been set such that an investigate exercise includes between one to three automatically generated multiple-choice questions, depending on the possibility to generate questions based on the given code.

[9] https://www.npmjs.com/package/acorn
[10] https://www.npmjs.com/package/acorn-walk
[11] https://www.stopify.org/
[12] https://github.com/lachrist/aran

### 5.2.5 Modify Exercise

As described in Section 4.1.2, two types of exercises have been provided for the modify phase of PRIMM: blanks exercises and parsons exercises. The collection for the parsons exercise is also similar to the *runEx* collection. Moreover, the implementation of the parsons exercise is mostly based on the implementation of the Parsons lens in the Study Lenses tool[13], which allows students to explore code through a parsons puzzle.



| NAME | TYPE |
|---|---|
| program | Rich text |
| description | Text |
| scene | Relation with *Scene* |
| probability | Number |
| completed | Boolean |
| sub_subjects | Relation with *SubSubject* |

Figure 5.8: The blanksEx collection

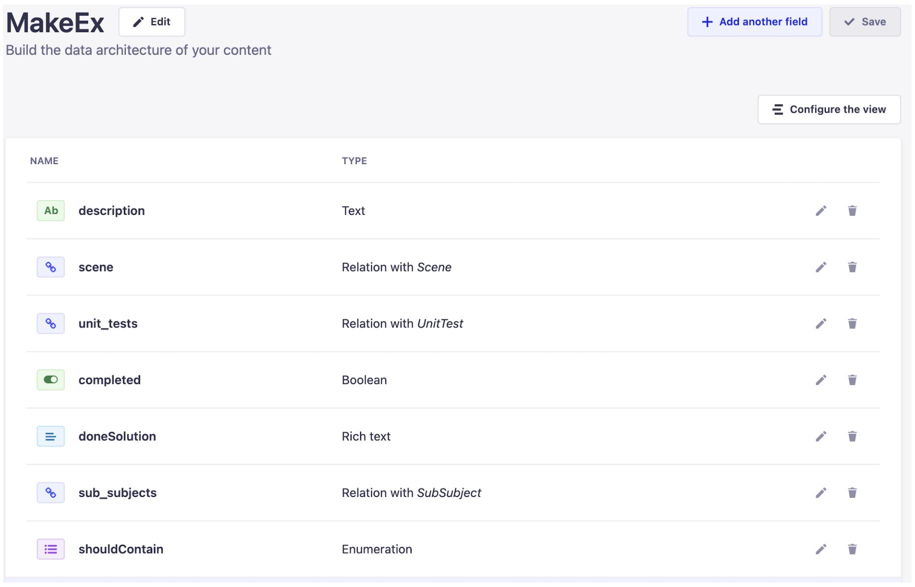
Furthermore, Figure 5.8 depicts the *blanksEx* collection type. To incorporate the blanks, the Acorn parser was employed. The code fragment in the *program* field is parsed using Acorn, to generate an AST. Acorn-walk is then used to traverse the tree. When variables and identifiers are encountered, they are potentially replaced by a blank (__), based on a randomly generated float and the collection's *probability* parameter. As such, based on the required difficulty level for the exercise, the *probability* parameter can be set accordingly. For the code editor in the User Interface, the library react-codemirror2[14] was employed.

### 5.2.6 Make Exercise

The *makeEx* collection is illustrated in Figure 5.9. In a make exercise, students are required to write code in a code editor, based on a provided description. The written code is then evaluated using predefined unit tests. Therefore, the collection contains a one-to-many relation field, *unit_tests*. The *UnitTest* collection comprises fields for a description and an assertion. Hence, the written code is evaluated together with the unit tests' assertions. If the test passes, then the description is displayed in green in the editor's console. Otherwise, it is displayed in red. Upon completion of the exercise, the student's solution is stored in the *doneSolution* field,

[13] `https://www.npmjs.com/package/react-codemirror2`
[14] `https://www.npmjs.com/package/react-codemirror2`

used to continuously display the solution in the editor thereafter. Finally, the *shouldContain* field is an enumeration containing all the keywords and variables the student should use in their solution. This is verified by traversing through the AST of the written code, similar to the blanks exercise.



| NAME | TYPE |
|---|---|
| description | Text |
| scene | Relation with *Scene* |
| unit_tests | Relation with *UnitTest* |
| completed | Boolean |
| doneSolution | Rich text |
| sub_subjects | Relation with *SubSubject* |
| shouldContain | Enumeration |

Figure 5.9: The makeEx collection

# Chapter 6

# Evaluation

Given that the JsStories application is intended to be used in an educational setting, it is important to consider feedback on the tool's perception from both parties involved. This includes students, as well as educators and organisational personnel. Consequently, the JsStories application was evaluated using a combination of interviews and a survey involving both parties. A detailed discussion of every interview, along with a thorough explanation of the survey's content and results is provided in the following sections. The main focus of the evaluation was on the impact of the tool on social inclusion, its effectiveness as a learning tool, as well as its applicability in programmes such as the one offered by HYFBE.

## 6.1 Interviews

A total of six interviews were conducted. All four alumni of the HYFBE program who had previously shared their stories for this research have been contacted for the evaluation. Unfortunately, only one of them agreed to provide some feedback on the prototype. Furthermore, we interviewed a former education coordinator at HYFBE. In addition to HYFBE, we reached out to the eight other coding schools affiliated with Migracode. Among these, we received a response from HYF Denmark, Borders None (Kroatia), Open Cultural Center (Spain) and Social Hackers Academy (Greece). We conducted semi-structured interviews with representatives of each of these organisations using the Microsoft Teams platform[1], during which the tool was demonstrated and feedback was provided. The interviews concluded with a discussion regarding the possibility of involving their students and teachers in a survey.

[1]`https://www.microsoft.com/nl-be/microsoft-teams/log-in`

### 6.1.1 HYF Belgium

#### Education Coordinator

The interview with the former education coordinator at HYFBE took place as an online meeting using the Microsoft Teams platform. As an education coordinator, his work there mainly consisted of developing and implementing the curriculum. Prior to that role, he also worked as an educational officer, assisting the students and closely monitoring their learning journeys. At the start of the interview, the motivation behind the JsStories application was explained. This was followed by a demonstration, showing and briefly describing the functionality of the application. The interview was conducted in an unstructured manner to allow for a genuine open discussion in which any remark and feedback, whether positive or constructive, may be shared. The interviewee was overall rather enthusiastic about the application and appreciated the storytelling aspect. More specifically, he liked the fact that every story is designed in a book form and that the integrated exercises fit the storyline. He believes that it has a lot to offer for students who are still doubting if programming, or computer science in general, is anything for them or students who are seeking something more than just exercises. The applicability of the tool within the HackYourFuture programme, or a similar programme, was then discussed. In terms of exercises, he noted that the current version of the application contains a limited number of exercises. To illustrate his point, he mentioned the problem with the quizzes of the JavaScript curriculum of W3schools. Despite the quality of the questions, only two or three questions per language feature are included in the quizzes, which is not enough practice for effective learning. Therefore, he suggested including optional exercises after each exercise, such that the learner can choose to complete them or not. The optional exercises would be of the same type and cover the same learning feature(s). For instance, once a learner has completed a parsons exercise covering the `if` control structure, they will be given the possibility to practice more on parsons exercises covering the `if` control structure. Additionally, he proposed adding a sandbox at the end of the story with a summarising table of content as interface. In the table of content, all the learning features covered in the story are summed up with some external links and references. The learner would also have the possibility to complete extra optional exercises for each language feature. This allows the learner to choose on what they want to practice further at the end of the story. He also noted that making it available at the end of the story would avoid making the book more daunting.

Furthermore, when asked about his opinion on using the tool as part of the selection procedure of a programme such as HackYourFuture, the interviewee liked the idea for two main reasons. He started by explaining that the majority of forms and websites simply show pictures of people who successfully learned programming, giving the impression that everyone can learn it. However, these websites often merely direct users to platforms such as freeCodeCamp[2], which he is not a big fan of. Therefore, he appreciated the fact that in JsStories, the story of representation is the content itself. Also, he claimed that testing someone on their ability to write code as they enter a programming course is not helpful, which is why he appreciated the fact that the tool uses the PRIMM pedagogy instead of simply asking to write code. Nonetheless, he stated that if the tool is used for the entrance examination, extra support needs to be provided such that students do not feel disoriented. For instance, a help button showing a recording of how to complete the exercise or providing some extra written explanation. Also, since trace tables are assumed to be new for students starting their programming journey, it should not be expected from them to

[2] `https://www.freecodecamp.org/`

just understand that they have to read the code, follow along the automatically filled in trace table and make the connection between these two. To support his claim, he referenced the expertise reversal effect, stating that the way novices and experts learn is different in the sense that novices require much more step by step approaches (see Section 2.4). Therefore, although automatically filled in trace tables are acceptable for more experienced students, it may not be suitable for beginners. Hence, he recommended adapting the run exercises to include blanks that the students must fill in instead of having the trace table filled in completely automatically. This way, focus can be put on certain code lines. Alternatively, multiple choice questions can be included after the trace table.

Moreover, we discussed other topics and directions the tool may cover rather than purely JavaScript if it were to be used for the entrance exam. He suggested to use the tool for typing exercises, depending on the level of student targeted. He went on to explain that in the past, typing was something that slowed down the students quite a lot. When provided with links to typing games, the students seemed to find it very helpful. Furthermore, he noted that the stories could also be used to explain the Internet in a high level way, covering topics such as the client, the server and protocols without going into more advanced topics like API calls and promises. He further recommended using the stories to explain markdown or the file system structure. Additionally, he noted that the tool could also be adapted to learn another programming language such as Python. In the last part of the interview, collaboration was also briefly discussed. The former education coordinator suggested two possible approaches to collaboration: at the exercise level and at the story level. The first option would be to provide a kind of chat box where students may discuss the exercises and move forward in the story together. The latter would involve a more asynchronous collaboration with branching stories. Each student would be working on a branch of the story and, once completed, they would be able to read the other branch of the story.

### Alumni

The second interview with one of the HackYourFuture graduates was conducted using the Vectera[3] platform. The interviewee was already familiar with the research and the motivation behind it since they had previously contributed by sharing their story. Therefore, the interview started with just a brief explanation to summarise the main points. The goal of the semi-structured interview was to let the interviewee test the prototype and then proceed with some prepared questions. A link to the JsStories tool was send to the interviewee. However, some components of the user interface were unreadable on their screen. At the time, we were unsure of the cause as the interviewee indicated that they did not have the dark mode enabled. Therefore, we proceeded by sharing our screen to present the tool and its functionalities, making sure to show an example of an exercise for every PRIMM phase. The interviewee was first asked about the potential impact the tool's storytelling aspect could have on the learners' motivation and willingness to engage with the content and exercises. They believe it can have a positive effect, but they noted that it mostly depends on the story itself and how much the learner can relate to it. Additionally, they stated that it also depends on the quality of the story in terms of how well it is written. Furthermore, when asked whether such stories could enhance social inclusion, they affirmed that reading happy ending stories of people having experienced similar situations could bring a sense of hope to the reader. For instance, reading about someone who

[3]`https://go.vectera.com/`

successfully completed their programming journey despite facing similar challenges may motivate a reader who is uncertain about continuing their programming journey and discourage them from dropping out.

Moreover, they mentioned that they would have appreciated having access to such a tool when they were following the HYF programme and believe it may be useful for the JavaScript module. Additionally, they commented that the tool may be used as means of preparation for the programme, giving novices an idea of what to expect before starting their programming journey. Furthermore, they valued the tool's use of the PRIMM pedagogy providing a step-by-step support for the students. When asked if they had any remarks about the design of the tool, they mentioned liking the usage of the book-style design including text, images and exercises. However, they remarked that the design of the parsons exercise could be improved. They mentioned that the purpose of the exercise was not clear at first because of the exercise's compression on one page of the book, requiring horizontal scrolling. The interviewee was then also asked if they believe a mobile version of the application would be useful. They stated that it would be more convenient for students who would like to use the tool when commuting for instance. Nonetheless, the interviewee remarked that it would only be beneficial in early stages of the learning process implying short coding exercises, as it would otherwise be difficult to code on a smartphone. Lastly, when asked if they had any other general remark or feedback, they began by expressing gratitude for the efforts put into this work to help socially vulnerable people. They found the tool promising and envisioned it being used not only in HYF but also other similar programs. Finally, we discovered that the unreadable components of the tool on his screen were due to the dark mode they had enabled. Therefore, they proposed to go through the stories after our interview to review them. They informed us via mail afterwards that they appreciated the stories and found them to be an interesting and promising way to learn programming.

### 6.1.2 HYF Denmark

For HYF Denmark[4], the interview was conducted with the founder of the organisation that launched the initiative in Denmark back in 2017, as well as the current managing director. The interviewees began by introducing themselves and their roles at HYF Denmark, while also providing more information on what HYF Denmark stands for. Following this introduction, the research and its motivations were explained, along with a live demonstration of the JsStories tool. After the demonstration, the interviewees had some questions related to the tool's functionalities, including whether or not the knowledge graph or learning path of the students could somehow be monitored by mentors. This is currently not possible, however it would be a useful feature to add in the future. Furthermore, as the tool is currently based on the curriculum of HYFBE, the used knowledge graph was explained and the relevant modules were compared to those of HYF Denmark's currciulum. The interviewees were then questioned about the storytelling aspect and the stories used, specifically whether they think it could help improve student motivation, engagement and social inclusion. They believe that, in general, it could have a positive impact to read and take into account stories of diverse people and experiences, especially for non-traditional bootcamps aiming to teach socially vulnerable people. However, they mentioned that, although the tool is interesting for exercise, HYF Denmark's strong focus on getting their students into the job market means that they target highly motivated and passionate students who do not need additional motivation during the programme. The managing director further commented

[4]`https://www.hackyourfuture.dk/`

that perhaps stories involving real job market cases, working experiences and tips might be more appropriate for HYF Denmark.

Furthermore, when we discussed the possibility of using the tool in the student selection procedure, both interviewees were enthusiastic about it. The current selection procedure at HYF Denmark involves a technical assignment using HTML, CSS and some JavaScript. They noted that for the selection procedure it is important to find the right balance so that the assignment is challenging enough to assess the skills of the applicant but not overly challenging as to disqualify too many applicants. The primary purpose of this assignment is to test people's amount of dedication and time management skills. Therefore, for the technical assignment, freeCodeCamp is utilised as they are familiar with the workload and time requirements. However, one limitation of freeCodeCamp noted by the interviewees, is that the included exercises and projects often reflect a Western-centric perspective. For instance, a lot of exercises feature cats as cute pets. However, in some cultures, cats are not commonly kept as pets, which might seem unfamiliar to people new to Europe. Consequently, they found that the JsStories with the actual kind of stories used, would be well-suited for the student selection process as it would introduce them to coding through reading rich achievement stories the students can relate to. This would help enhance their initial motivation to pursue coding. Furthermore, as the interviewees did not have a technical background, we did not question them about the type of exercises selected for the PRIMM approach. The interview concluded with a discussion on the survey, during which both the founder and the managing director allowed for involvement of their students and mentors. Hence, we were given access to the official slack platform of HYF Denmark, enabling us to contact the students and mentors for the survey. Finally, the founder of the organisation suggested that the JsStories tool could potentially be tested in future student selection procedures at HYF Denmark.

### 6.1.3 Borders None

The fourth interview was conducted with a representative of the organisation called Borders None[5], located in Croatia. This organisation focuses on training refugees to pursue a career in web development. Compared to HYFBE and HYF Denmark, Borders None is a smaller organisation with fewer students. The interview was conducted with a member of the directing team at Borders None. First, a demonstration of the JsStories tool was provided, showing its functionalities while also explaining the motivation behind each aspect. Overall, the interviewee expressed enthusiasm for the tool. She characterised the tool as a solution that 'catches two birds with one stone', serving as both an exercise tool to enhance the students' knowledge as well as a motivation booster. As a social worker, the interviewee regularly uses storytelling and believes it may be useful in numerous situations. In this particular case, she believes the storytelling aspect can boost the motivation of students to complete the program, as well as enhance their sense of belonging. She further explained that providing students with real-life stories of people they can relate to may be highly beneficial, as students are interested in whether there are graduates who have completed the course and found employment in the field of web development. The possibility of using the JsStories tool in the student selection procedure was then discussed. The interviewee explained that at Borders None, the programme is divided into a beginner course and an advanced course. The beginner course covers HTML and CSS, after which the students must pass a final project. JavaScript and other more advanced topics are covered in the advanced

[5] `https://www.bordersnone.com/coders-without-borders/`

course. Students may enrol in the advanced course if they have completed the beginner course or if they have prior knowledge of HTML and CSS, demonstrated through a project. For the beginner course, students are selected based on discussions with them, assessing their motivation and skills. Therefore, the interviewee believes that the tool is only suitable for use during the program and homework sessions, but not in their selection procedure, as they do not include a technical assignment. Furthermore, she mentioned that Borders None has a high dropout rate, mainly because students lose motivation or lack self-confidence as foreigners. Hence, she suggested that an adaptation of the tool covering HTML and CSS might be useful in the beginner course to foster initial motivation and a sense of belonging among the students. This might help them stay motivated and boost their chances of completing both courses. Moreover, as language courses are offered at Borders None, the interviewee commented that an adapted version of the tool aimed at teaching languages, using such stories, would also be beneficial. She states that learning from concrete examples one can relate to makes it much easier to focus and remember. Therefore, she always advises teachers at Borders None to make examples as concrete possible and to link them to the students' situations and preferences. When asked about the design of the tool, she mentioned liking the book-form design and found the tool very intuitive. As a conclusion to the interview, she agreed to share the survey with former students and teachers at Borders None on their slack platform.

### 6.1.4 Open Cultural Center

Open Cultural Center[6], founded in 2016, is a non-profit organisation that is present in both Spain and Greece. Their objective is to facilitate the integration process of refugees and migrants by offering educational courses in various fields such as languages and technology, as well as providing sports and cultural activities. In Spain, the organisation runs two open-access code academies: Migracode[7] and CodeWomen[8]. The former is open to everyone, while the latter is focused on empowering women. We conducted an interview with the programme manager of Migracode. As with the other interviews, the research was explained and a demonstration of the tool was provided. The interviewee valued the tool, finding it a promising and interesting learning resource. Regarding the design of the tool, the interviewee liked the book format and the integration of an editor in some of the exercises. Additionally, he appreciated that multiple kind of exercises were included, allowing students to interact with code in various ways. As the tool is currently quite small, we also discussed the number of exercises it should incorporate in order to maximise the tool's usefulness. The interviewee particularly liked the idea of identifying students' mistakes and recommending optional exercises based on that. He commented that learners usually enjoy having optional exercises recommended to them, offering them the choice to complete them or not. He further noted that without proper testing, it is difficult to determine the effectiveness of the tool. However, he stated that including the tool in the program would likely be beneficial as it provides an additional learning resource. Furthermore, the interviewee mentioned that Flexbox Froggy[9] is used in their program for CSS. This gamified learning resource is highly appreciated by the students as it provides an engaging way to learn CSS. Currently, ways to incorporate more gamification into their program are explored. He believes that the storytelling aspect of the JsStories tool is a powerful feature that can further enhance engagement.

[6] `https://openculturalcenter.org/`
[7] `https://migracode.org/`
[8] `https://codewomenbarcelona.org/`
[9] `https://flexboxfroggy.com/`

As the students can relate to the incorporated stories they can better engage with it, making it easier to focus and learn. He further mentioned that the tool might enhance the social belonging among students providing a first step for improvement of social inclusion.

The applicability of the tool within the program of Migracode was then discussed. The interviewee believes that depending on how well it aligns with their curriculum, the tool could definitely fit into the program as a companion tool for students to practice. Moreover, when asked if he believes the tool could also be used in the student selection procedure, he expressed strong appreciation for the idea. As part of Migracode's selection procedure, students are required to complete a course on Khan Academy[10] and develop a small project using HTML and CSS. Currently, the project does not include JavaScript, but the interviewee mentioned that they are considering to include it. He explained that when starting with the JavaScript module, some students feel overwhelmed learning about the programming concepts for the first time, which sometimes leads to drop outs. This confirms our findings at HYFBE. Therefore, the interviewee believes it makes a lot of sense to use the tool in the student selection procedure. It will both introduce students to JavaScript before they join the program, as well as, offer them insights into what to expect from the program. When discussing other directions for the tool, the interviewee shared that an adaptation of the tool for their language lab would be beneficial. Regarding the survey, the interviewee planned to share it with students and teachers during the on-site course sessions. At the conclusion of the interview, the interviewee mentioned that we could reach out to them for future testing of the tool at Migracode.

### 6.1.5 Social Hackers Academy

The last interview was held with the co-founder and executive director of the non-profit organisation Social Hackers Academy[11] (SHA). SHA is primarily aimed at teaching web development to refugees and other socially vulnerable groups. After having demonstrated the tool and outlined the research, the interviewee started by expressing gratefulness for the effort put into this research aiming to improve the social inclusion of vulnerable groups. He then continued by explaining that although SHA initially started as an organisation to help refugees and migrants, they have now shifted to also allow less vulnerable people to join the program by paying a small fee, while more vulnerable people receive a scholarship. The lack of funding, which is not as available as in other countries, made this adjustment necessary. Moreover, following the Covid-19 crisis, the SHA program moved completely online. As a result, they have students from numerous countries although more than half their students still reside in Greece. When asked about the storytelling aspect and the impact it could have on the students, the interviewee indicated being a big fan of storytelling and found it an interesting way to learn. Overall, he found the tool to be a great concept and confirmed that also their students have a lot of difficulties with JavaScript. However, he noted that the tool's usefulness depends on whether the learning objectives can be achieved using the tool, which can only be determined by testing the tool in a real setting. Furthermore, he stated that it also depends on the learning modalities. Not everyone learns the same way, therefore the tool might not be effective for all. When asked about the applicability of the tool in the SHA program, the interviewee stated that, as he is not the Head of Education and that he does not have a technical background, he felt unqualified to answer these questions. He therefore

[10]https://www.khanacademy.org/
[11]https://socialhackersacademy.org/

suggested to forward our request to their Head of Education so that she might be able to take some time to meet and answer our questions. Unfortunately, we did not hear back from her.

## 6.2 Survey

The survey was carried out through a Google Form[12], which incorporated a video demonstration of the tool along with a number of questions including multiple choice, linear scale, and short answer questions. Given that the tool is currently only available as a desktop application and considering the busy schedules of the students, as reported by the representatives of the organisation, we chose to incorporate a video demonstration. As such, the students are not required to install the application and test it in order to fill in the questionnaire, thus saving valuable time. As discussed in the interview section, the survey was eventually shared in three organisations: Borders None, HYF Denmark and Open Cultural Center (specifically Migracode), primarily via the organisations' Slack platform. A total of 7 responses were collected, which is unfortunately a rather low response rate. However, this outcome was expected due to the busy schedules of the students and the fact that the anonymous survey was shared in large groups of students and teachers, resulting in individuals that feel less inclined to participate as they might assume other students or teachers would.

As sections and conditional questions were used, the survey was designed to allow both students and teachers to fill it in. Nine questions applied to both groups. Teachers were provided with an additional question, while students were given 2 more. The first question is a multiple-choice question in which participants can select the organisation they are affiliated with out of 7 options. Three respondents are affiliated with HYF Denmark, 2 with Open Cultural Center and 2 with Borders None. In the following multiple choice question the participants could indicate if they are a (former) student or a (former) teacher. Of the 7 respondents, only one identifies as a teacher, specifically a former teacher. The other 6 respondents are current students. The participants were then asked if they believe that the use of storytelling in the tool could positively influence the motivation and willingness of students to complete exercises and potentially help lower dropout rates. The question was provided with a linear scale ranging from 1 (Strongly disagree) to 10 (Strongly agree). Two of the participants answered with the highest score of 10, 3 with the score of 9 and 2 with the score of 8. These answers indicate that all participants believe the tool may somewhat help improve the motivation of students. The next question, using the same linear scale, focused on the impact of the specific type of stories used. The respondents were asked whether they believe any kind of story would have the same impact as the current real-life stories of HYFBE alumni. For this question, the participants had differing opinions. The majority answered with a neutral score, as 2 participants responded with a score of 5 and 3 respondents with a score of 6. The other two participants answered with scores of 3 and 8. These answers suggest that although the majority of the respondents is not persuaded that any kind of story might have the same impact, they may believe that using other stories could also be valuable, depending on the kind of story. Based on the results, we realise that an additional question aimed at assessing the impact of the stories currently used on the sense of belonging among students would have been beneficial.

[12]`https://www.google.com/intl/en_be/forms/about/`

Furthermore, the respondents were asked if they believe the JsStories tool could be useful in the program they are affiliated with. Two participants answered with a score of 7, 3 with a score of 8 and the remaining 2 people with a score of 9. Hence, all participants agreed that the tool may be beneficial in their program. Based on their responses to a question regarding this, the students would have liked the opportunity to use the JsStories tool during their time following the program. Out of the 6 students, 2 answered with the highest score of 10, 3 with a score of 8 and one person with a score of 7. Additionally, the potential use of the tool in the student selection procedure, giving students insights into what to expect, was questioned. The participants seemed to value this idea as 4 out of the 7 participants gave it a score of 8, 2 people a score of 9 and the remaining person a score of 10. Furthermore, the design of the tool received favourable scores of 7, 8 and 10. Moreover, all respondents favoured a web-version of the tool over a desktop application, which is something that should be considered in future adaptations of the tool. The participants were also asked, in an optional short-answer question, if they believe the tool could be adapted for another purpose. One respondent suggested that an adaptation of the tool could focus on learning HTML. The other participants either did not answer or mentioned that they appreciated the current version of the tool focusing on JavaScript. Additionally, a short-answer question regarding the selection of exercises for the PRIMM approach was provided to the teachers. The sole teacher respondent approved the current selection of exercises, finding them appropriate for PRIMM. On the other hand, students were also asked if they would be interested in contributing their story for potential use in the tool, while maintaining anonymity. 5 out of the 6 students expressed interest. This implies that for future expansions of the tool students would potentially be willing to contribute with their stories. Finally, a section was provided where participants could leave any additional feedback and remark, as well as a section where they could leave their email-address if they wished to be contacted for additional information on the survey or for a follow-up meeting. One of the respondents left a short remark '*add helper*', by which the respondent probably meant that a type of helper feature should be provided such as hints.

## 6.3 Main Findings of the Evaluation

- Participants of the evaluation were enthusiast about the idea of using JsStories as a companion tool for students to practice throughout the programme, primarily noting its potential to enhance student engagement. Furthermore, using the tool in the student selection procedure of the organisation received even greater appreciation from participants due to the specific stories currently included. Reading alumni stories before joining the programme can help prospective students know what to expect from the programme and can foster a sense of belonging as they might relate to the stories.
- The usefulness of JsStories as a learning tool depends on how well it aligns with the organisation's curriculum. To further enhance its effectiveness, additional exercises such as optional exercises or links to external resources could be included in the stories. Moreover, additional functionalities to support teachers might be added such as a functionality to monitor student's progress to provide better support for their students.
- A web-based version of the JsStories application is favoured over a desktop application.
- Based on the feedback, an adapted version of the JsStories tool might also be beneficial for other purposes. For instance, the tool could be used to learn a different programming language or specific computer science concepts such as high-level Internet protocols.

Furthermore, it could potentially be adapted for language courses and typing courses. Additionally, different kinds of stories could be used, such as stories targeting the job market to prepare students for it.

# Chapter 7

# Discussion and Future Work

This chapter provides a discussion and addresses potential future work to further improve the JsStories tool, taking into account the results of the evaluation. It includes adaptations in terms of design and purpose, as well as extra features that could further enhance the tool.

## 7.1 Discussion

In this research, we aimed to come up with a solution to address the high dropout rates in organisations like HYFBE. We introduced the JsStories tool, a JavaScript learning tool focused on improving social inclusion through storytelling and enhancing the learning environment using the PRIMM approach and knowledge graphs. The findings of the evaluation indicate that the use of real anonymised stories of HYFBE alumni may foster a sense of belonging among students and boost their motivation, which is a first step towards improving social inclusion. This is because students might relate to the characters, situations and experiences in such stories based on their own background. Furthermore, the PRIMM approach was also appreciated as it provides support for the students guiding them step by step and allowing them to interact with code in different ways. However, a limitation of the evaluation, conducted through interviews and a survey, is that the tool was not tested by the target group. A thorough evaluation including testing of the tool over an extended period of time might provide more useful insights on the PRIMM approach, the selected type of exercises as well as the personalised learning environment provided through the use of a knowledge graph and story locking. Moreover, while the tool was initially designed for use during the programme, we also evaluated its potential use during the student selection procedure of organisations such as HYFBE, which was highly appreciated and encouraged by all participants of the evaluation. Currently, most of the Migracode schools use platforms such as freeCodeCamp and Khan Academy for their technical assignments in the student selection process. However, these platforms have some limitations such as using a western-centric perspective in the exercises. Hence, incorporating the JsStories tool with the current stories in the student selection procedure may be beneficial as it allows prospective students to get a first exposure to programming within the context of the programme they are applying to. This should therefore definitely be further explored in future work.

## 7.2 Future Work

### 7.2.1 Design Improvement

Although the design of the tool was given favourable ratings in the evaluation, there are still areas it could benefit from improvements. Currently, a book design is used for the tool and exercises are limited to a single page. This leads to suboptimal user experience for the modify exercises, which include a Parsons puzzle, as horizontal scrolling is required to have a clear view of the drag-and-drop area. One potential improvement could be to provide an overview of the exercise on the book page itself, along with a button to open a larger screen where the exercise can be fully viewed and completed. Alternatively, an exercise could be spread over two book pages. However, the first option is preferable as the entire window screen could be potentially used. Similar adjustments could be made for the run exercise as it also requires horizontal scrolling to view the complete trace table, depending on the number of variables and the variable name sizes. Due to time constraints, these improvements have not been implemented yet. In addition, it was noted during the evaluation that the tool is not compatible with the dark mode yet, which should be taken into account in future adaptations of the tool.

### 7.2.2 Additional Educational Material

In the current version of the JsStories tool, a total of six types of exercises are provided for the different PRIMM phases. However, to enhance the tool and allow more variety, further research could be done on additional effective types of exercises that could be incorporated. As discussed in the interview with the former Education Coordinator at HYFBE (see Section 6.1.1), the tool could also offer more opportunities for practice as it is currently limited. For instance, optional exercises could be included so that after completing an exercise, the student can choose to further exercise on the covered topic as needed. In addition, students may be required to complete supplementary exercises based on the number of attempts for an exercise and the mistakes they made. Furthermore, a sandbox could be incorporated at the end of a book, providing an overview of all covered topics along with extra resources and exercises. The exercises currently integrated in the books all align with the storylines. However, aligning exercises with a storyline can be time-consuming. One option could therefore be to include mandatory exercises that fit the storyline, as is currently the case, along with random, automatically generated exercises for optional practice. Alternatively, other approaches beyond the tool itself could also be explored. For instance, instead of incorporating additional exercises in the tool, options could be to provide links to external resources and exercises related to the topics or to incorporate the tool in an extern Learning Management System (LMS) providing more custom exercises and resources.

### 7.2.3 Prior Knowledge Test

One of the limitations of the tool is that the present version does not take into consideration students' prior knowledge. Hence, every student begins with the same set of locked and unlocked books, regardless of their understanding of the subjects covered in these books. However, it would be preferable to leave the choice of whether to further exercise on familiar subjects or start with books covering subjects beyond their knowledge level, to the students. Therefore, it would be beneficial to include a prior knowledge test for students to complete after signing up, to assess their current knowledge level and understanding. This test should not be limited to plain multiple-choice or short-answer questions. Instead, it could for instance also be used to introduce the tool using storytelling and thus serve a dual purpose. Using the test's results, the knowledge graph will be updated at the start, unlocking books that cover topics the students already master or are ready to learn. Consequently, future research could focus on exploring approaches to efficiently assess students' prior knowledge.

### 7.2.4 Students' Progress

As the tool is intended for use in an educational setting, additional features could be incorporated to provide support for the teachers. Using the data contained in the knowledge graph, one extra functionality could be to allow teachers to monitor students' progress with data visualisation tools. Furthermore, additional data such as topics or exercises the students found challenging along with details on the mistakes and number of attempts they made, could be provided. This would enable teachers to provide students with targeted help or extra material on these topics. If multiple students encounter difficulties with these topics, the teacher could also decide to review these during the course lessons. Hence, there is certainly room for further research to identify additional features that would enhance the tool's value.

### 7.2.5 Web-based Application

The JsStories tool is currently available as a desktop application. However, based on the results from the evaluation, a web-based version would be preferable as it requires no installations from the users ans simply runs on the browser. While the transition from desktop application to web-based version is possible, it would require some further adaptations to the setup of the Strapi server to enable multi-user authentication, as it currently only supports single-user functionality.

### 7.2.6 Other Directions

Other directions and purposes for the tool can also be explored in future research. Currently, the focus of the tool is on the JavaScript programming language. Nonetheless, as discussed in the evaluation, adaptations of the tool could for instance be centered on CSS, HTML or other subjects such as typing or language learning. Moreover, the tool was initially intended to be used as an exercise tool to accompany students in programs like the one provided by HYFBE. However, based on the feedback received in the evaluation, the tool could potentially also be used in student selection procedures or in pre-courses. Nonetheless, this would require some adaptations necessary for novices, such as providing supplemental explanation for the exercises. Additionally, to evaluate the efficacy of using the adapted tool in the student selection procedure, a thorough evaluation in a real setting is necessary. As HYF Denmark and Open Cultural Center were willing to participate in the testing phase, the evaluation could potentially be integrated in their student selection procedure. Ideally, a form of A/B testing would be employed for the evaluation.

For half of the program applicants, the organisation would follow their regular student selection procedure, including the current technical assignment. The other half of the students would be provided with the tool and tasked to complete certain stories as part of the technical assignment. Following this, the experience of both groups would be compared based on a qualitative analysis aimed at evaluating the students' motivation to complete the program, their sense of belonging, the insights gained on JavaScript and their perception of the technical assignment. Additionally, at the end of the program, the dropout rates could be analysed to evaluate whether the use of the tool in the selection procedure had an impact on them. Furthermore, the current version of the tool, which is intended for use during the program, could also benefit from such an extended evaluation, as only a qualitative analysis was conducted without testing it in a real setting.

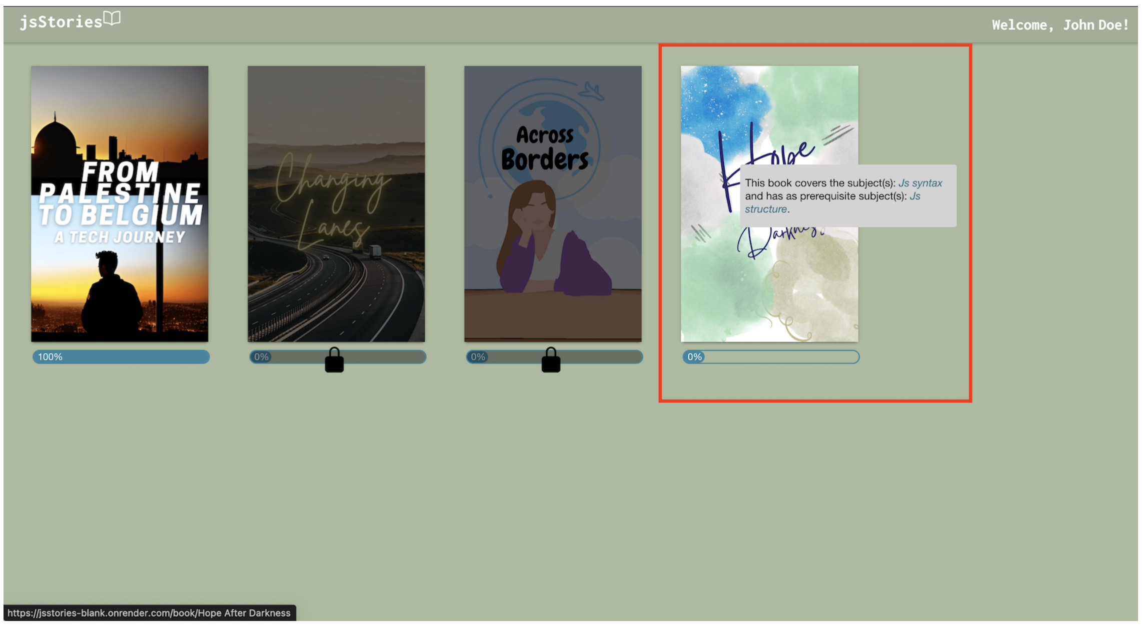
# Chapter 8

# Conclusion

The work undertaken in this thesis focused on addressing the limitations in teaching and learning programming, which contribute to high dropout rates in organisations aimed at providing socially vulnerable people with computer science education. We introduced the JsStories tool, which is built based on insights gained from both literature and a visit at HYFBE. The tool focuses on the JavaScript programming language based on the perceived difficulties of students. Additionally, the tool aims to provide a solution taking into account suggestions for improvement found in literature. Firstly, the tool is based on storytelling. The real pseudonimysed stories of HYFBE's alumni aim to foster a sense of belonging among socially vulnerable students in an engaging way, thereby enhancing their motivation. The tool thus takes into account the background and needs of students, as discussed in the suggestions for improvement. Secondly, since one of the suggestions implied the need for a flexible learning environment where students can learn at their own pace, the tool was built on top of a knowledge graph representing HYFBE's curriculum. Based on the knowledge graph, story locking has been implemented, ensuring that students are provided with exercises at their own knowledge level. Thirdly, limitations in terms of teaching methods and learning materials were addressed by using the PRIMM approach for the exercises. This approach focuses on improving student's understanding of code before they are required to write code. In terms of implementation, the Strapi CMS was used to facilitate the management of stories and exercises, even by non-technical organisational personal. The results of the qualitative analysis, which included interviews and a survey, indicate that the tool could bring added value to organisations such as HYFBE. It could serve as a first step towards improving social inclusion and potentially help lower the dropout rates. However, a more extended evaluation in one of these organisations, including testing of the tool throughout a longer period of time in a realistic scenario, would be beneficial. Furthermore, the current tool is only a first prototype that can certainly be improved in terms of added functionality and implementation. Finally, as the idea of using the tool in the student selection procedure received unanimous approving feedback, this option should definitely be further explored.

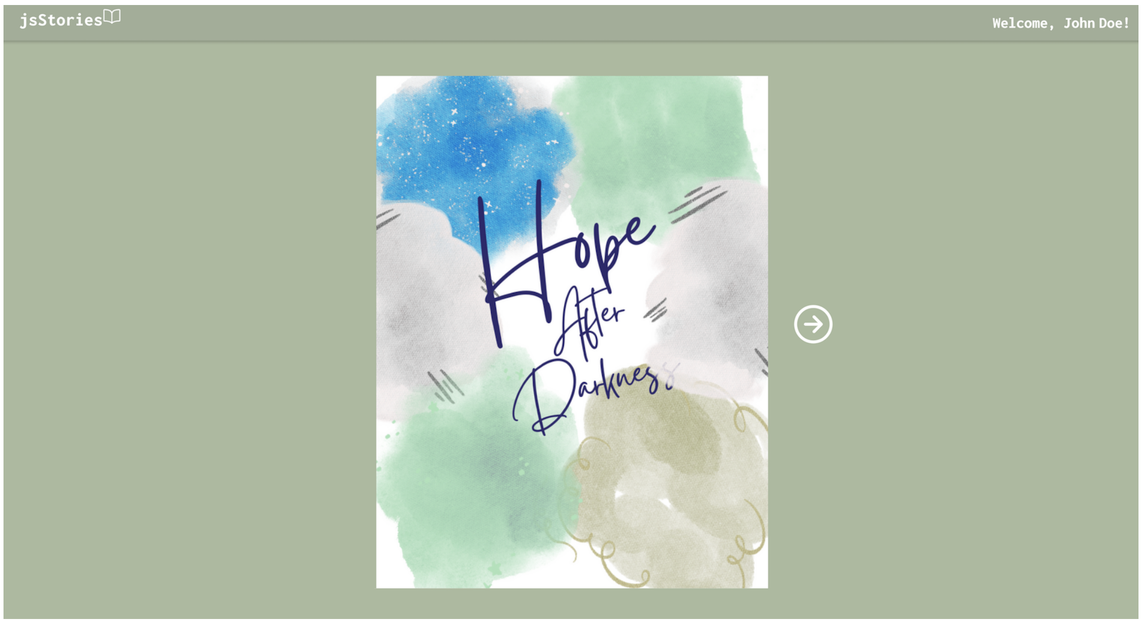
# Appendix A

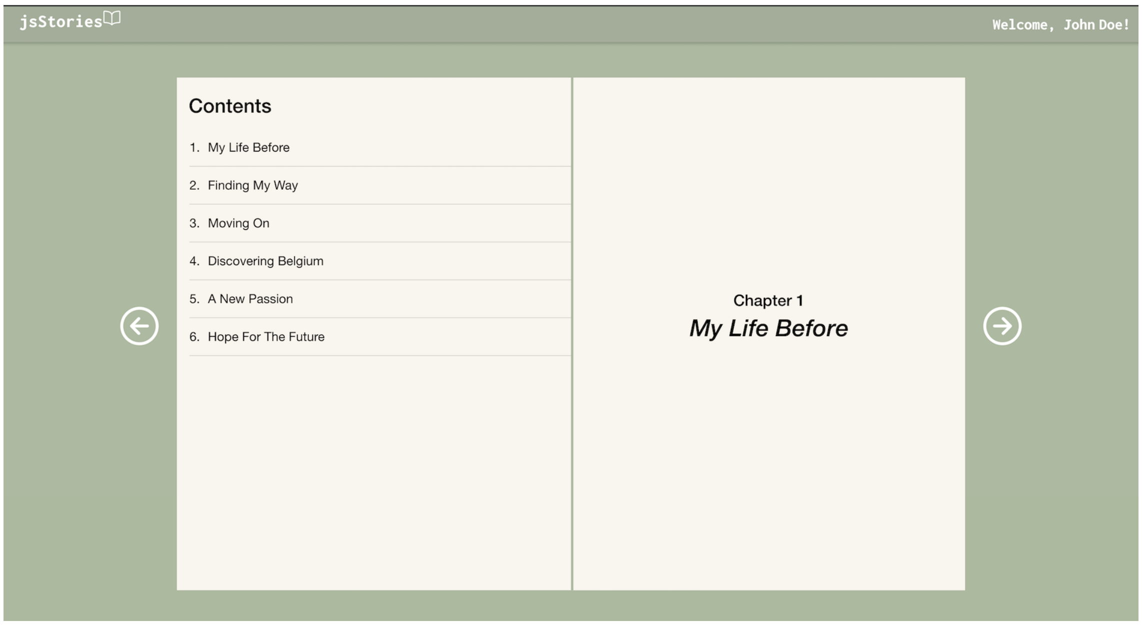
# JsStory: Hope After Darkness



This book covers the subject(s): *Js syntax* and has as prerequisite subject(s): *Js structure*.

https://jsstories-blank.onrender.com/book/Hope After Darkness

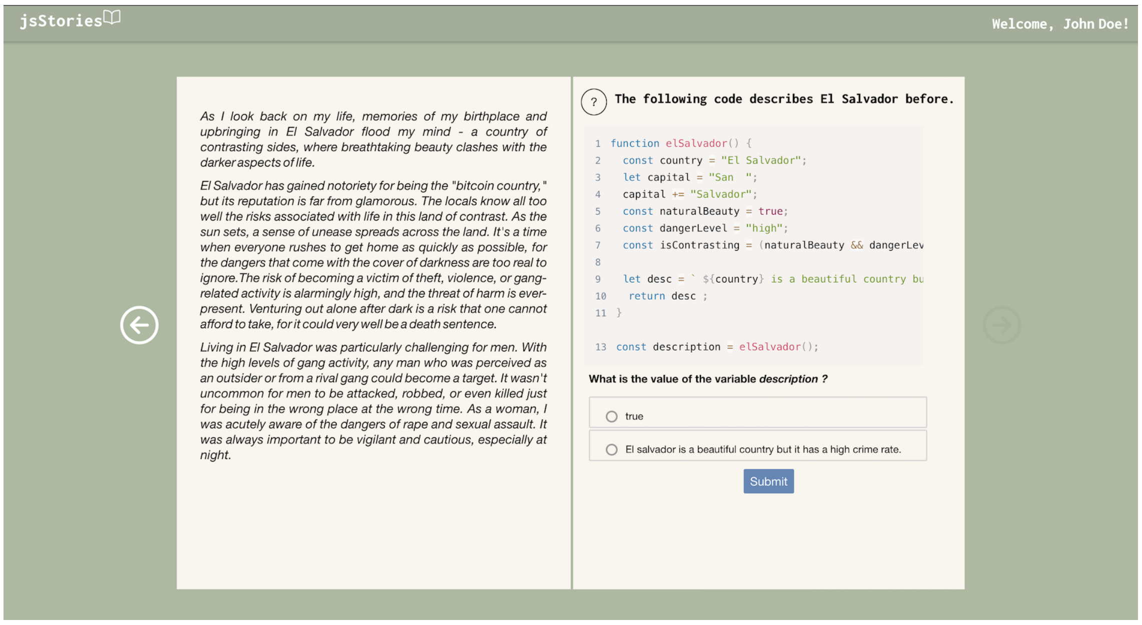




Contents

1. My Life Before
2. Finding My Way
3. Moving On
4. Discovering Belgium
5. A New Passion
6. Hope For The Future

Chapter 1
*My Life Before*



*As I look back on my life, memories of my birthplace and upbringing in El Salvador flood my mind - a country of contrasting sides, where breathtaking beauty clashes with the darker aspects of life.*

*El Salvador has gained notoriety for being the "bitcoin country," but its reputation is far from glamorous. The locals know all too well the risks associated with life in this land of contrast. As the sun sets, a sense of unease spreads across the land. It's a time when everyone rushes to get home as quickly as possible, for the dangers that come with the cover of darkness are too real to ignore.The risk of becoming a victim of theft, violence, or gang-related activity is alarmingly high, and the threat of harm is ever-present. Venturing out alone after dark is a risk that one cannot afford to take, for it could very well be a death sentence.*

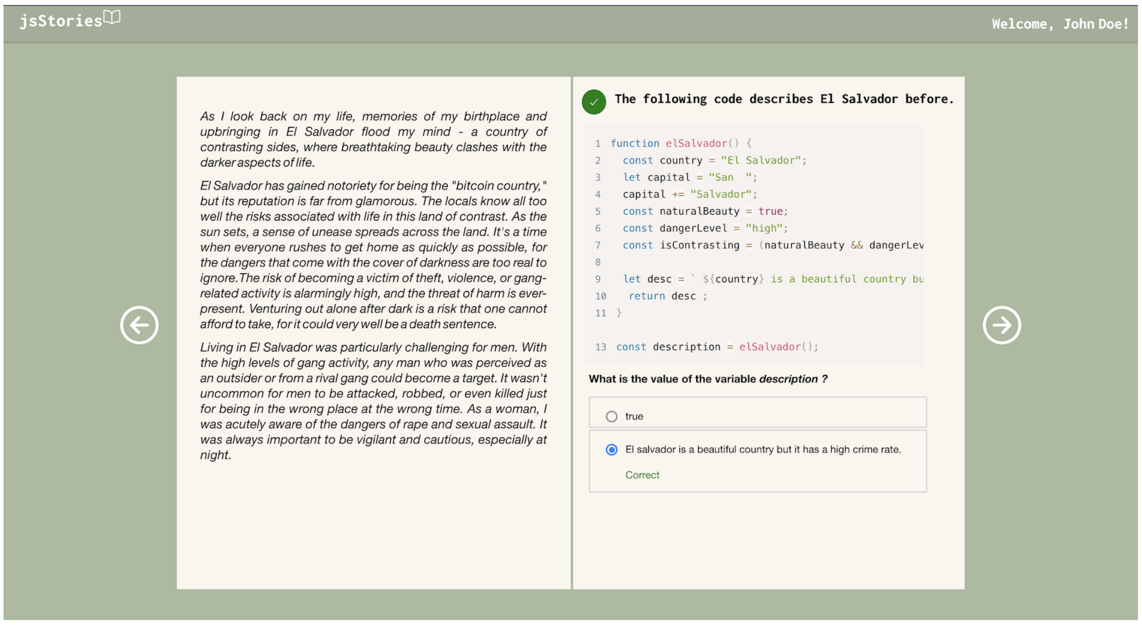
*Living in El Salvador was particularly challenging for men. With the high levels of gang activity, any man who was perceived as an outsider or from a rival gang could become a target. It wasn't uncommon for men to be attacked, robbed, or even killed just for being in the wrong place at the wrong time. As a woman, I was acutely aware of the dangers of rape and sexual assault. It was always important to be vigilant and cautious, especially at night.*

**The following code describes El Salvador before.**

```
function elSalvador() {
  const country = "El Salvador";
  let capital = "San  ";
  capital += "Salvador";
  const naturalBeauty = true;
  const dangerLevel = "high";
  const isContrasting = (naturalBeauty && dangerLev

  let desc = ` ${country} is a beautiful country bu
   return desc ;
 }

const description = elSalvador();
```

**What is the value of the variable *description* ?**

- ○ true
- ○ El salvador is a beautiful country but it has a high crime rate.


jsStories
Welcome, John Doe!

*As I look back on my life, memories of my birthplace and upbringing in El Salvador flood my mind - a country of contrasting sides, where breathtaking beauty clashes with the darker aspects of life.*

*El Salvador has gained notoriety for being the "bitcoin country," but its reputation is far from glamorous. The locals know all too well the risks associated with life in this land of contrast. As the sun sets, a sense of unease spreads across the land. It's a time when everyone rushes to get home as quickly as possible, for the dangers that come with the cover of darkness are too real to ignore.The risk of becoming a victim of theft, violence, or gang-related activity is alarmingly high, and the threat of harm is ever-present. Venturing out alone after dark is a risk that one cannot afford to take, for it could very well be a death sentence.*

*Living in El Salvador was particularly challenging for men. With the high levels of gang activity, any man who was perceived as an outsider or from a rival gang could become a target. It wasn't uncommon for men to be attacked, robbed, or even killed just for being in the wrong place at the wrong time. As a woman, I was acutely aware of the dangers of rape and sexual assault. It was always important to be vigilant and cautious, especially at night.*

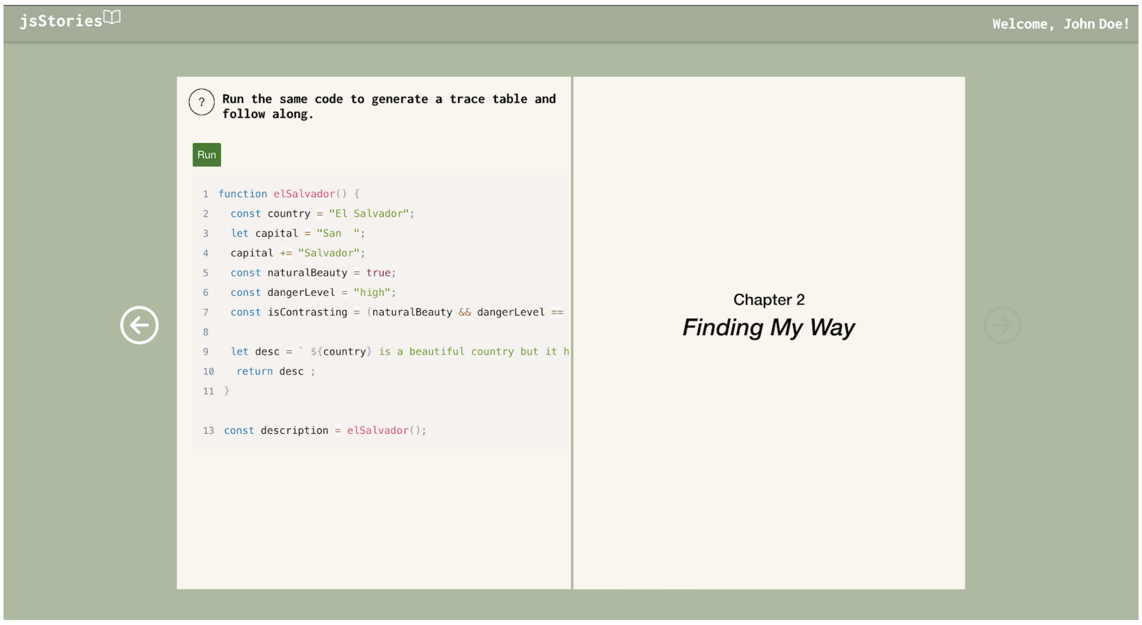
**The following code describes El Salvador before.**

```
function elSalvador() {
  const country = "El Salvador";
  let capital = "San  ";
  capital += "Salvador";
  const naturalBeauty = true;
  const dangerLevel = "high";
  const isContrasting = (naturalBeauty && dangerLev

  let desc = ` ${country} is a beautiful country bu
   return desc ;
 }

const description = elSalvador();
```

**What is the value of the variable *description* ?**

- ○ true
- [✓ El salvador is a beautiful country but it has a high crime rate.]

Correct

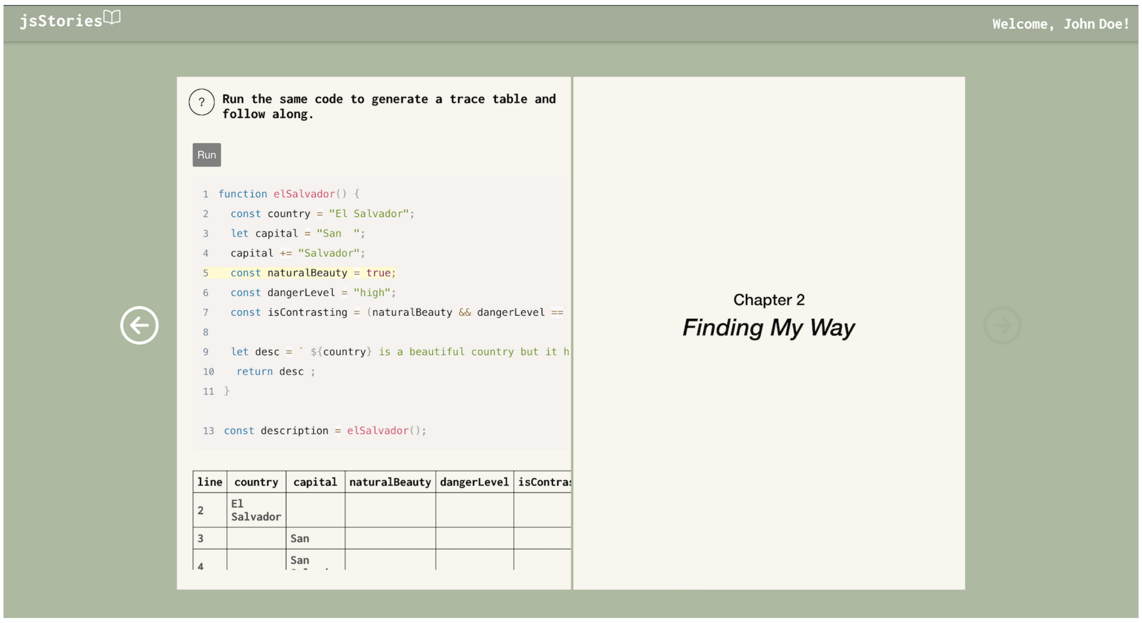


Run the same code to generate a trace table and follow along.

```
function elSalvador() {
  const country = "El Salvador";
  let capital = "San  ";
  capital += "Salvador";
  const naturalBeauty = true;
  const dangerLevel = "high";
  const isContrasting = (naturalBeauty && dangerLevel ==

  let desc = ` ${country} is a beautiful country but it h
   return desc ;
}

const description = elSalvador();
```

Chapter 2

*Finding My Way*



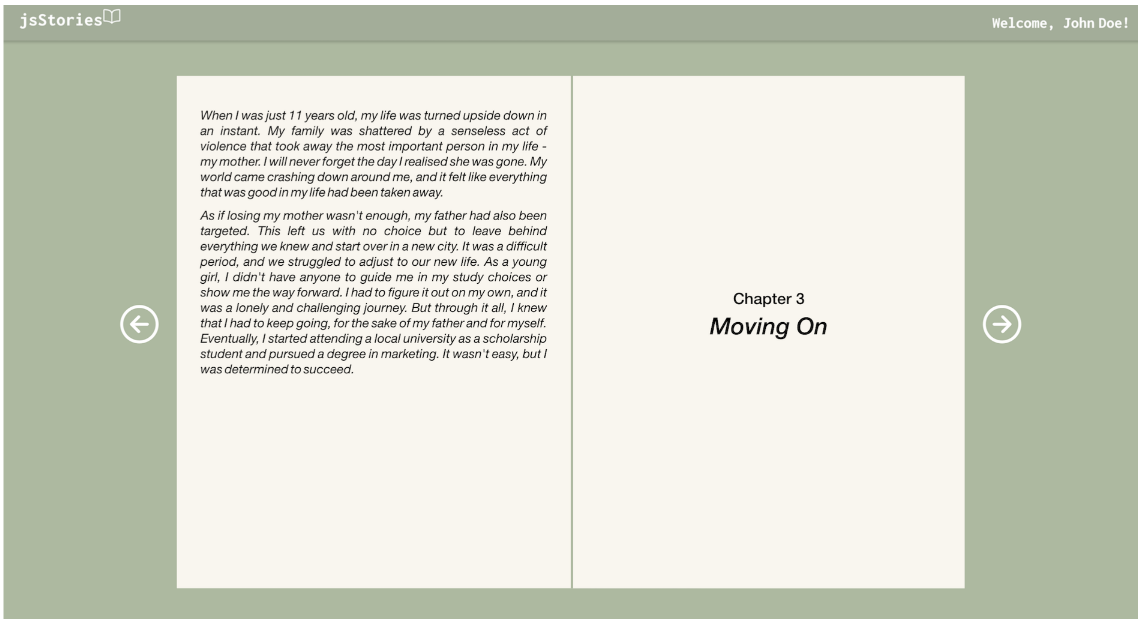
Run the same code to generate a trace table and follow along.

```
function elSalvador() {
  const country = "El Salvador";
  let capital = "San  ";
  capital += "Salvador";
  const naturalBeauty = true;
  const dangerLevel = "high";
  const isContrasting = (naturalBeauty && dangerLevel ==

  let desc = ` ${country} is a beautiful country but it h
   return desc ;
}

const description = elSalvador();
```

| line | country | capital | naturalBeauty | dangerLevel | isContras |
|---|---|---|---|---|---|
| 2 | El Salvador | | | | |
| 3 | | San | | | |
| 4 | | San | | | |

Chapter 2

*Finding My Way*

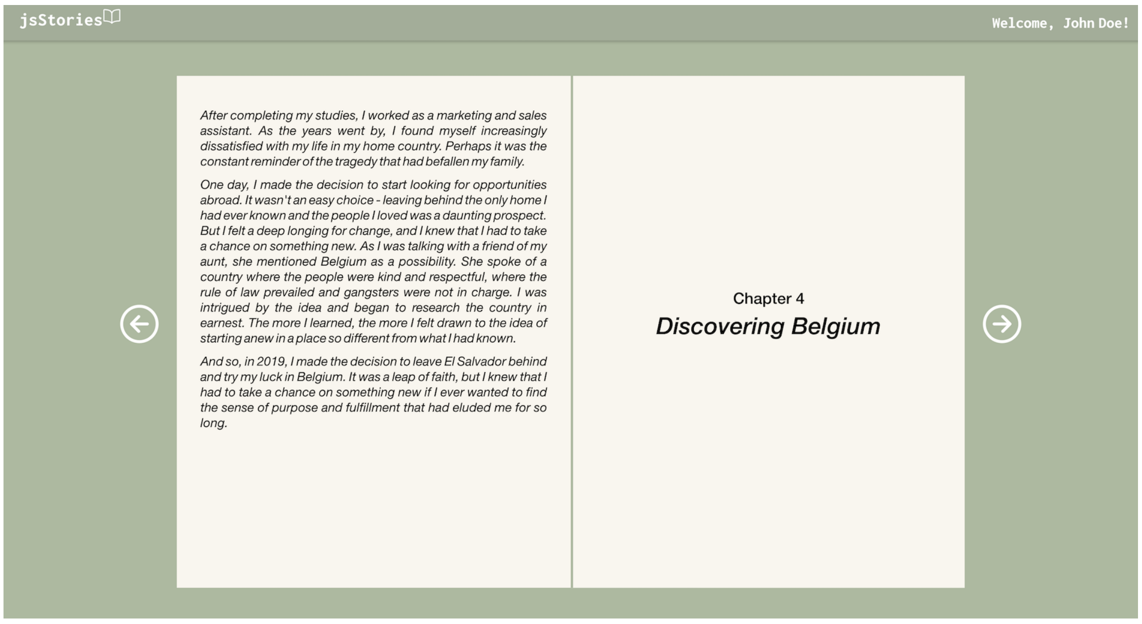


*When I was just 11 years old, my life was turned upside down in an instant. My family was shattered by a senseless act of violence that took away the most important person in my life - my mother. I will never forget the day I realised she was gone. My world came crashing down around me, and it felt like everything that was good in my life had been taken away.*

*As if losing my mother wasn't enough, my father had also been targeted. This left us with no choice but to leave behind everything we knew and start over in a new city. It was a difficult period, and we struggled to adjust to our new life. As a young girl, I didn't have anyone to guide me in my study choices or show me the way forward. I had to figure it out on my own, and it was a lonely and challenging journey. But through it all, I knew that I had to keep going, for the sake of my father and for myself. Eventually, I started attending a local university as a scholarship student and pursued a degree in marketing. It wasn't easy, but I was determined to succeed.*

Chapter 3

*Moving On*



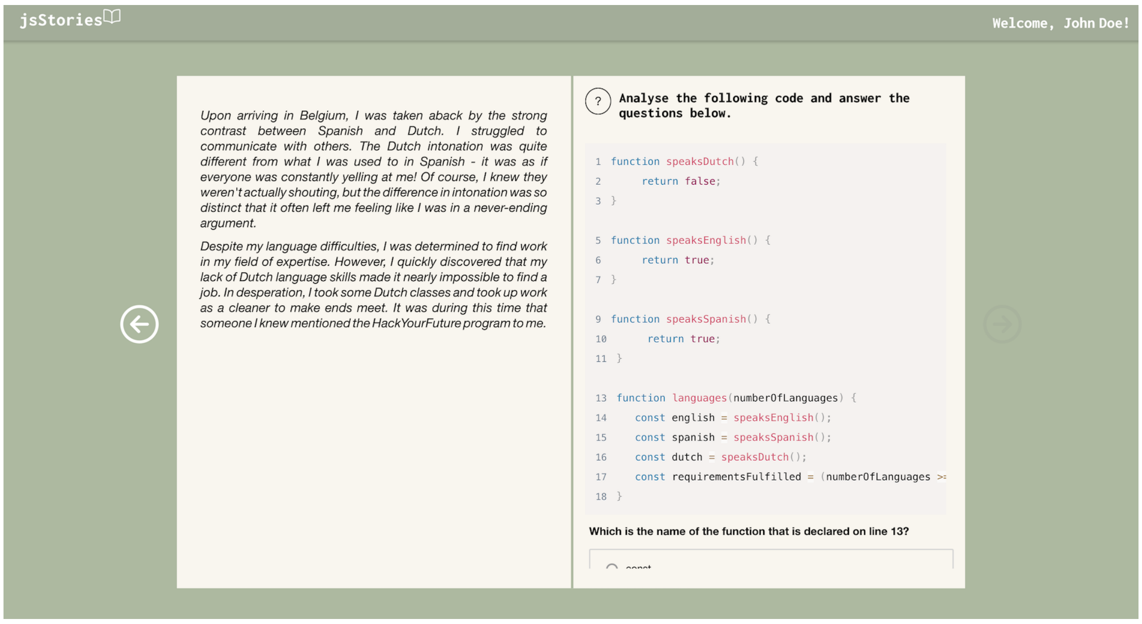
*After completing my studies, I worked as a marketing and sales assistant. As the years went by, I found myself increasingly dissatisfied with my life in my home country. Perhaps it was the constant reminder of the tragedy that had befallen my family.*

*One day, I made the decision to start looking for opportunities abroad. It wasn't an easy choice - leaving behind the only home I had ever known and the people I loved was a daunting prospect. But I felt a deep longing for change, and I knew that I had to take a chance on something new. As I was talking with a friend of my aunt, she mentioned Belgium as a possibility. She spoke of a country where the people were kind and respectful, where the rule of law prevailed and gangsters were not in charge. I was intrigued by the idea and began to research the country in earnest. The more I learned, the more I felt drawn to the idea of starting anew in a place so different from what I had known.*

*And so, in 2019, I made the decision to leave El Salvador behind and try my luck in Belgium. It was a leap of faith, but I knew that I had to take a chance on something new if I ever wanted to find the sense of purpose and fulfillment that had eluded me for so long.*

Chapter 4

*Discovering Belgium*



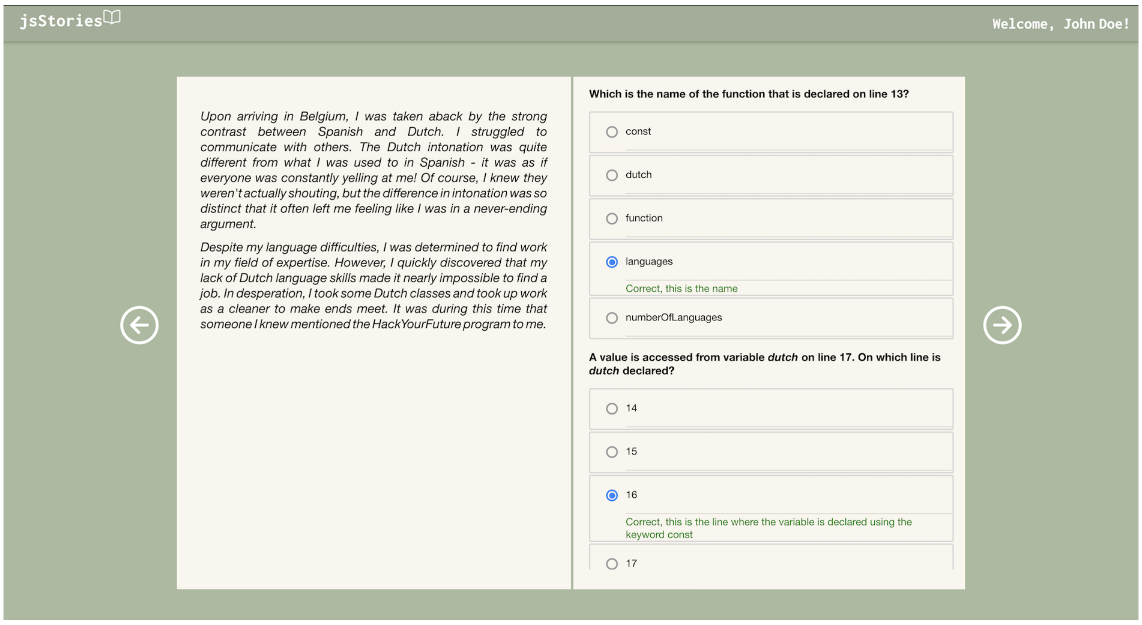
*Upon arriving in Belgium, I was taken aback by the strong contrast between Spanish and Dutch. I struggled to communicate with others. The Dutch intonation was quite different from what I was used to in Spanish - it was as if everyone was constantly yelling at me! Of course, I knew they weren't actually shouting, but the difference in intonation was so distinct that it often left me feeling like I was in a never-ending argument.*

*Despite my language difficulties, I was determined to find work in my field of expertise. However, I quickly discovered that my lack of Dutch language skills made it nearly impossible to find a job. In desperation, I took some Dutch classes and took up work as a cleaner to make ends meet. It was during this time that someone I knew mentioned the HackYourFuture program to me.*

Analyse the following code and answer the questions below.

```
1  function speaksDutch() {
2       return false;
3  }

5  function speaksEnglish() {
6       return true;
7  }

9  function speaksSpanish() {
10       return true;
11  }

13  function languages(numberOfLanguages) {
14     const english = speaksEnglish();
15     const spanish = speaksSpanish();
16     const dutch = speaksDutch();
17     const requirementsFulfilled = (numberOfLanguages >=
18  }
```

**Which is the name of the function that is declared on line 13?**



*Upon arriving in Belgium, I was taken aback by the strong contrast between Spanish and Dutch. I struggled to communicate with others. The Dutch intonation was quite different from what I was used to in Spanish - it was as if everyone was constantly yelling at me! Of course, I knew they weren't actually shouting, but the difference in intonation was so distinct that it often left me feeling like I was in a never-ending argument.*

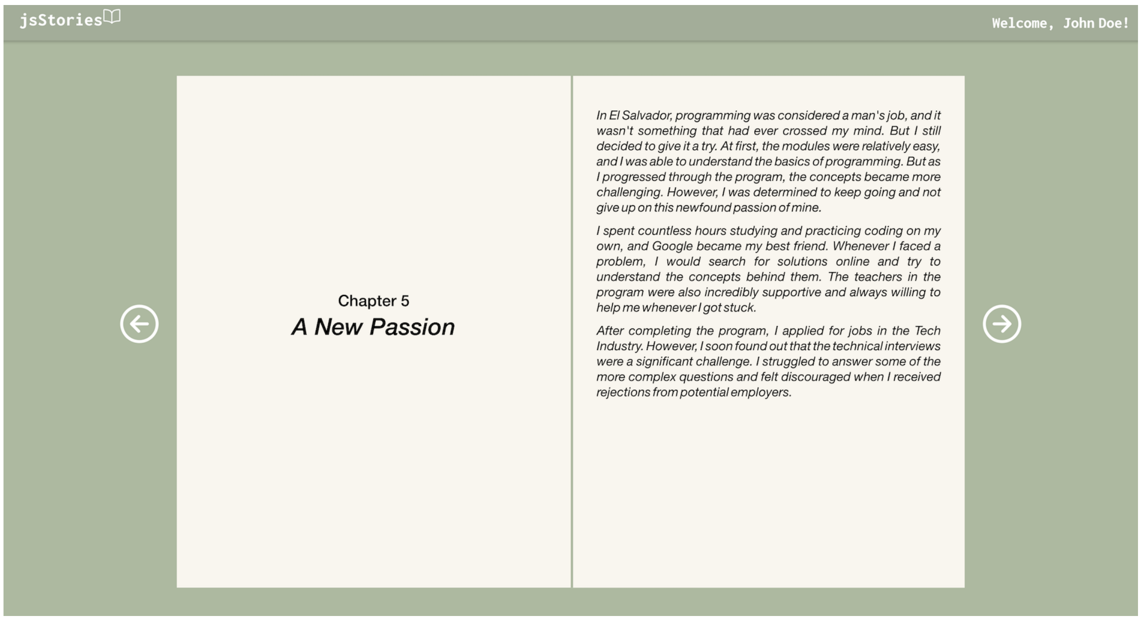
*Despite my language difficulties, I was determined to find work in my field of expertise. However, I quickly discovered that my lack of Dutch language skills made it nearly impossible to find a job. In desperation, I took some Dutch classes and took up work as a cleaner to make ends meet. It was during this time that someone I knew mentioned the HackYourFuture program to me.*

**Which is the name of the function that is declared on line 13?**

- const
- dutch
- function
- [✓ languages]
  Correct, this is the name
- numberOfLanguages

**A value is accessed from variable *dutch* on line 17. On which line is *dutch* declared?**

- 14
- 15
- [✓ 16]
  Correct, this is the line where the variable is declared using the keyword const
- 17



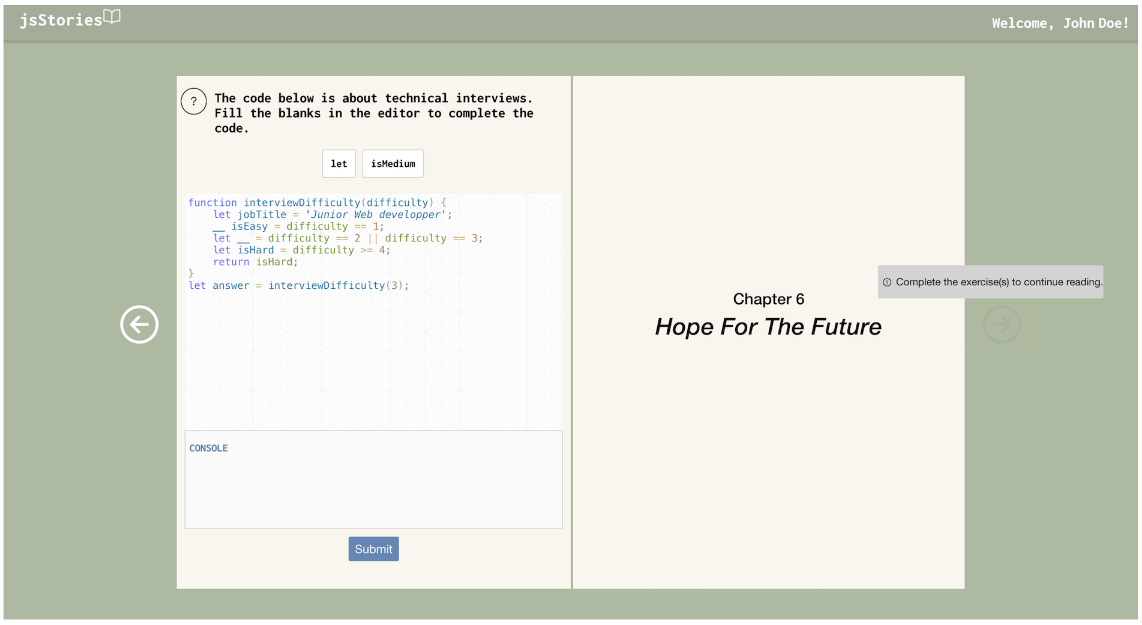
Chapter 5

*A New Passion*

*In El Salvador, programming was considered a man's job, and it wasn't something that had ever crossed my mind. But I still decided to give it a try. At first, the modules were relatively easy, and I was able to understand the basics of programming. But as I progressed through the program, the concepts became more challenging. However, I was determined to keep going and not give up on this newfound passion of mine.*

*I spent countless hours studying and practicing coding on my own, and Google became my best friend. Whenever I faced a problem, I would search for solutions online and try to understand the concepts behind them. The teachers in the program were also incredibly supportive and always willing to help me whenever I got stuck.*

*After completing the program, I applied for jobs in the Tech Industry. However, I soon found out that the technical interviews were a significant challenge. I struggled to answer some of the more complex questions and felt discouraged when I received rejections from potential employers.*

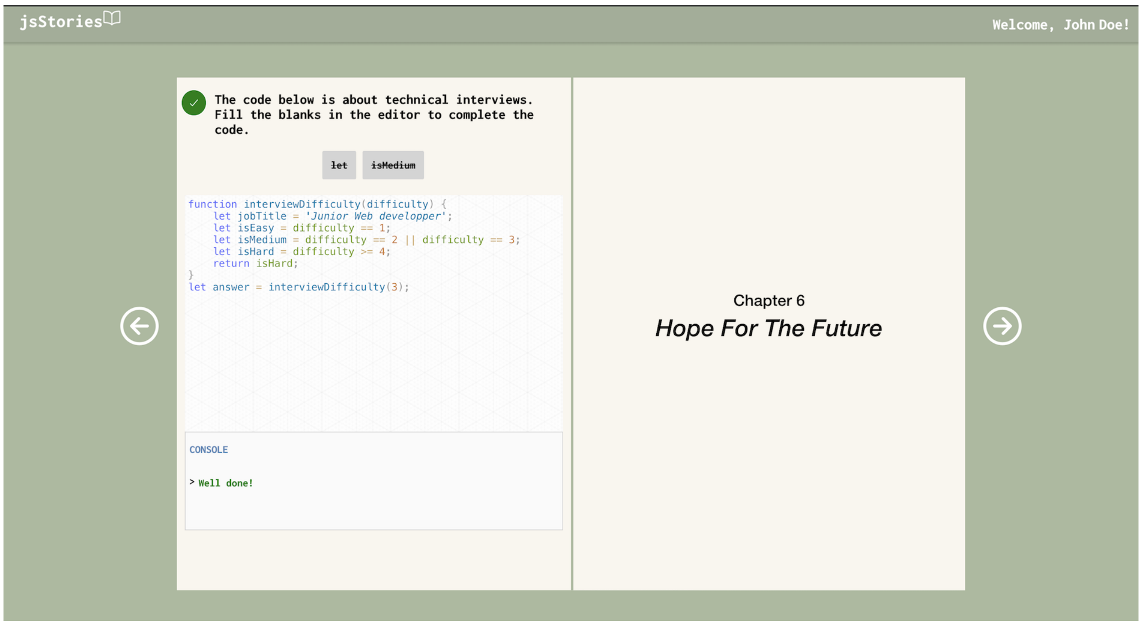


**The code below is about technical interviews. Fill the blanks in the editor to complete the code.**

```
function interviewDifficulty(difficulty) {
    let jobTitle = 'Junior Web developper';
    __ isEasy = difficulty == 1;
    let __ = difficulty == 2 || difficulty == 3;
    let isHard = difficulty >= 4;
    return isHard;
}
let answer = interviewDifficulty(3);
```

Chapter 6

*Hope For The Future*



The code below is about technical interviews. Fill the blanks in the editor to complete the code.

~~let~~ ~~isMedium~~

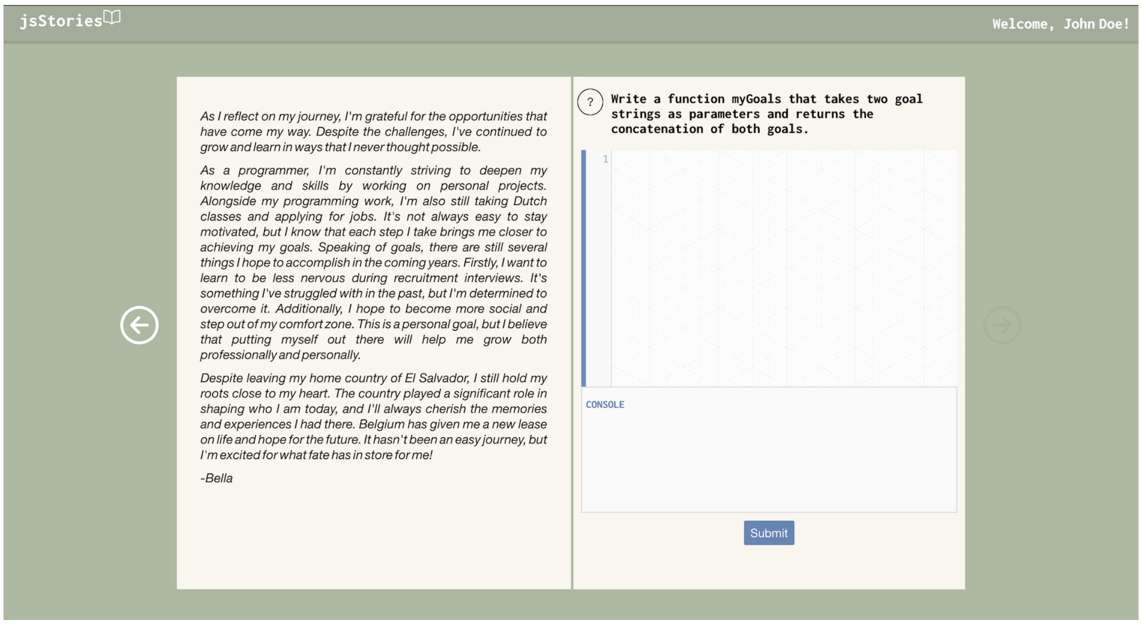
```
function interviewDifficulty(difficulty) {
    let jobTitle = 'Junior Web developper';
    let isEasy = difficulty == 1;
    let isMedium = difficulty == 2 || difficulty == 3;
    let isHard = difficulty >= 4;
    return isHard;
}
let answer = interviewDifficulty(3);
```

CONSOLE

```
> Well done!
```

Chapter 6

*Hope For The Future*



*As I reflect on my journey, I'm grateful for the opportunities that have come my way. Despite the challenges, I've continued to grow and learn in ways that I never thought possible.*

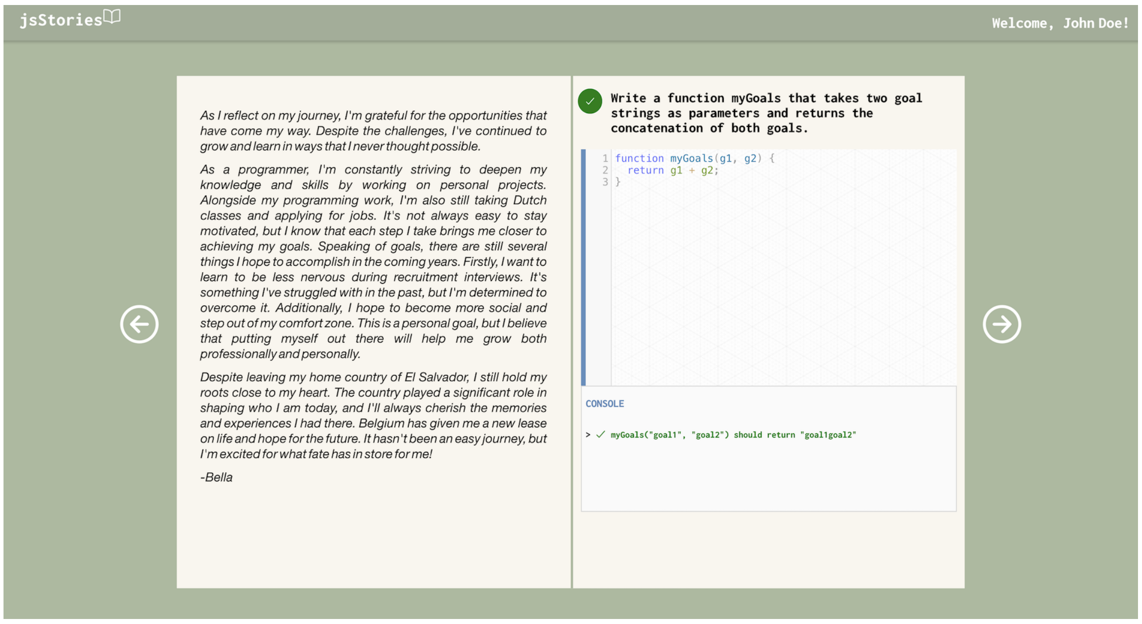
*As a programmer, I'm constantly striving to deepen my knowledge and skills by working on personal projects. Alongside my programming work, I'm also still taking Dutch classes and applying for jobs. It's not always easy to stay motivated, but I know that each step I take brings me closer to achieving my goals. Speaking of goals, there are still several things I hope to accomplish in the coming years. Firstly, I want to learn to be less nervous during recruitment interviews. It's something I've struggled with in the past, but I'm determined to overcome it. Additionally, I hope to become more social and step out of my comfort zone. This is a personal goal, but I believe that putting myself out there will help me grow both professionally and personally.*

*Despite leaving my home country of El Salvador, I still hold my roots close to my heart. The country played a significant role in shaping who I am today, and I'll always cherish the memories and experiences I had there. Belgium has given me a new lease on life and hope for the future. It hasn't been an easy journey, but I'm excited for what fate has in store for me!*

*-Bella*

Write a function myGoals that takes two goal strings as parameters and returns the concatenation of both goals.

CONSOLE



*As I reflect on my journey, I'm grateful for the opportunities that have come my way. Despite the challenges, I've continued to grow and learn in ways that I never thought possible.*

*As a programmer, I'm constantly striving to deepen my knowledge and skills by working on personal projects. Alongside my programming work, I'm also still taking Dutch classes and applying for jobs. It's not always easy to stay motivated, but I know that each step I take brings me closer to achieving my goals. Speaking of goals, there are still several things I hope to accomplish in the coming years. Firstly, I want to learn to be less nervous during recruitment interviews. It's something I've struggled with in the past, but I'm determined to overcome it. Additionally, I hope to become more social and step out of my comfort zone. This is a personal goal, but I believe that putting myself out there will help me grow both professionally and personally.*

*Despite leaving my home country of El Salvador, I still hold my roots close to my heart. The country played a significant role in shaping who I am today, and I'll always cherish the memories and experiences I had there. Belgium has given me a new lease on life and hope for the future. It hasn't been an easy journey, but I'm excited for what fate has in store for me!*

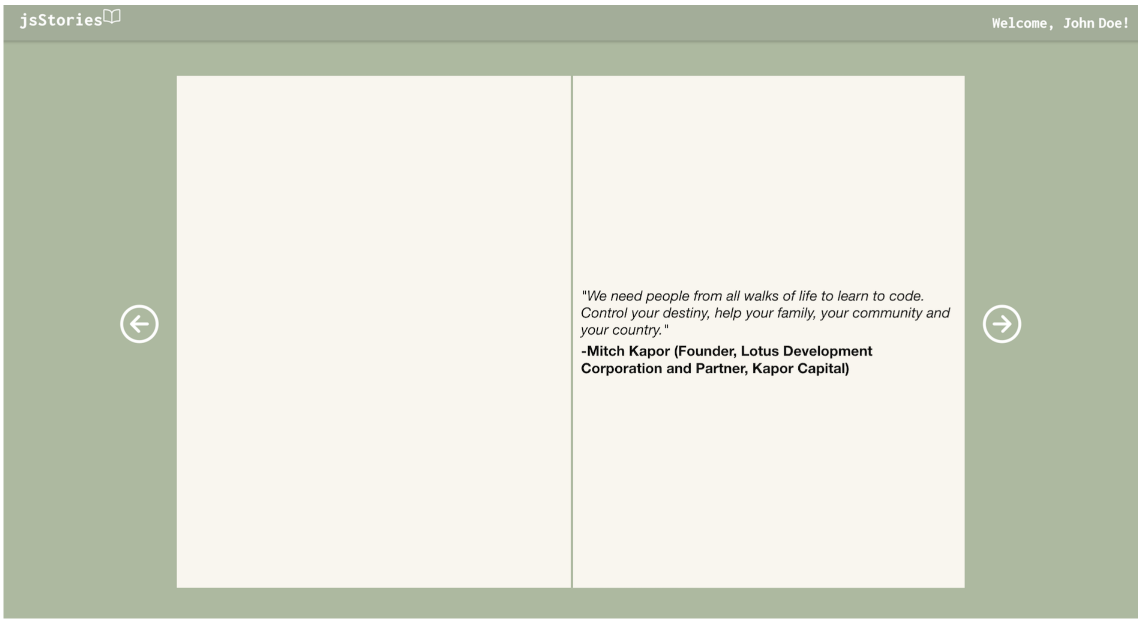
*-Bella*

Write a function myGoals that takes two goal strings as parameters and returns the concatenation of both goals.

```
function myGoals(g1, g2) {
  return g1 + g2;
}
```

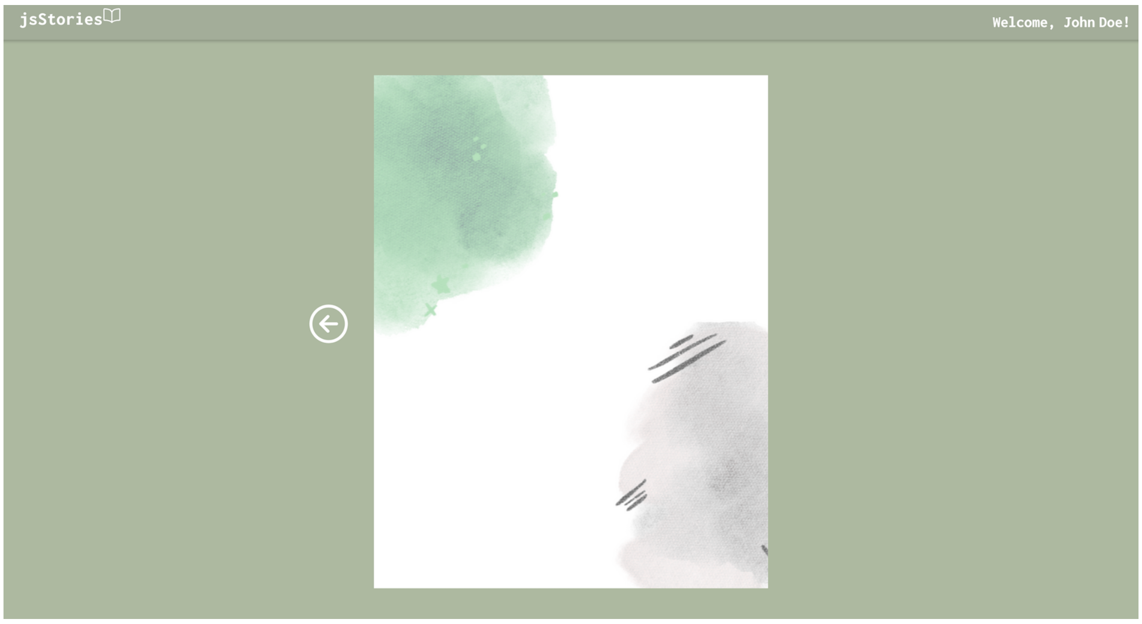
CONSOLE

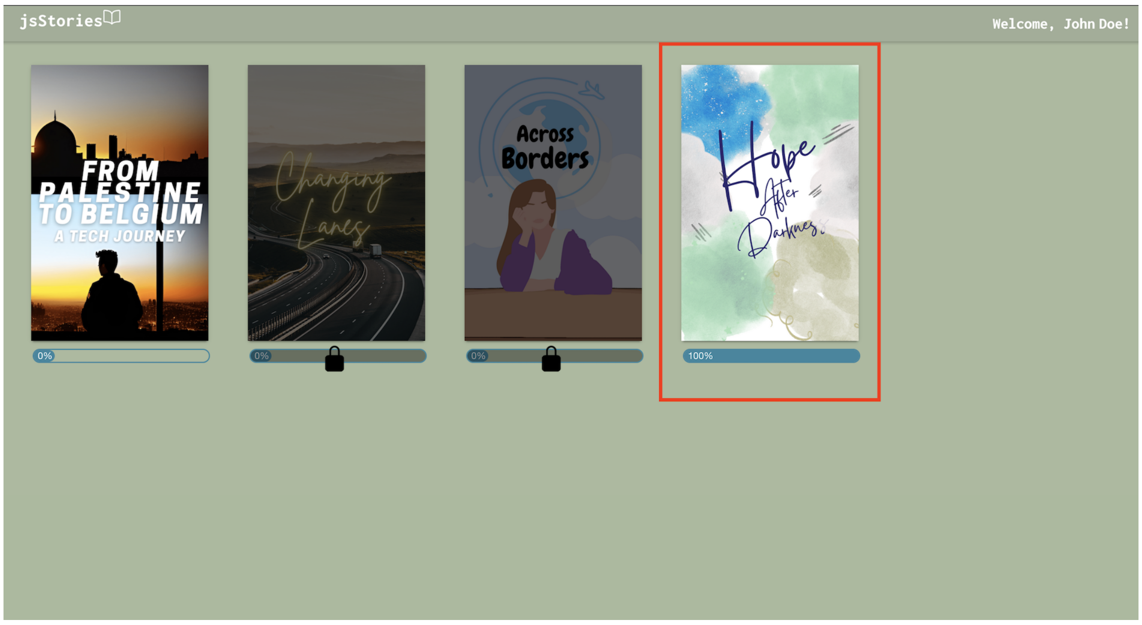
```
> ✓ myGoals("goal1", "goal2") should return "goal1goal2"
```



*"We need people from all walks of life to learn to code. Control your destiny, help your family, your community and your country."*

**-Mitch Kapor (Founder, Lotus Development Corporation and Partner, Kapor Capital)**


jsStories
Welcome, John Doe!


jsStories
Welcome, John Doe!
FROM PALESTINE TO BELGIUM
A TECH JOURNEY
0%
Changing Lanes
0%
Across Borders
0%
Hope After Darkness
100%

# Appendix B

# Survey Results

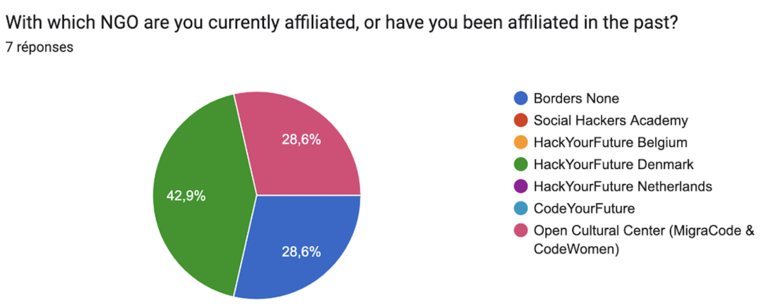


Figure B.1: Question 1.

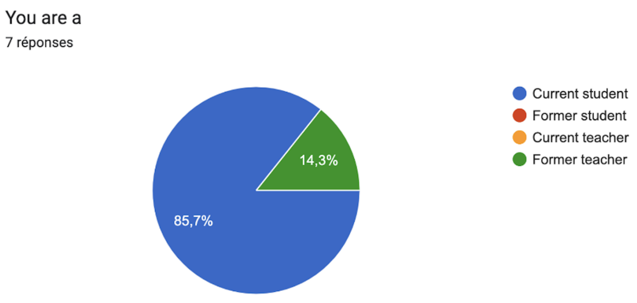


Figure B.2: Question 2.

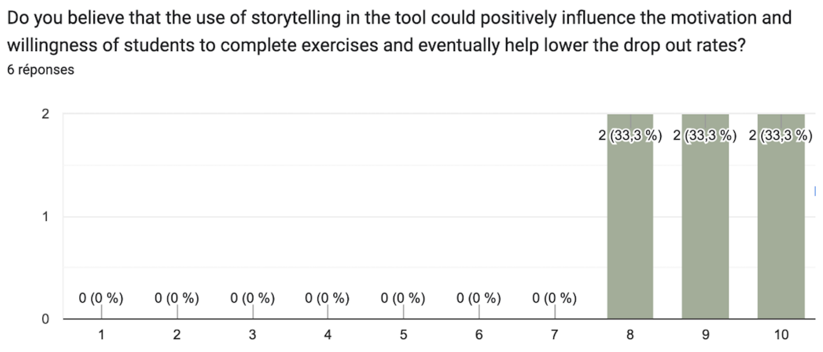


Figure B.3: Student responses to Question 3.

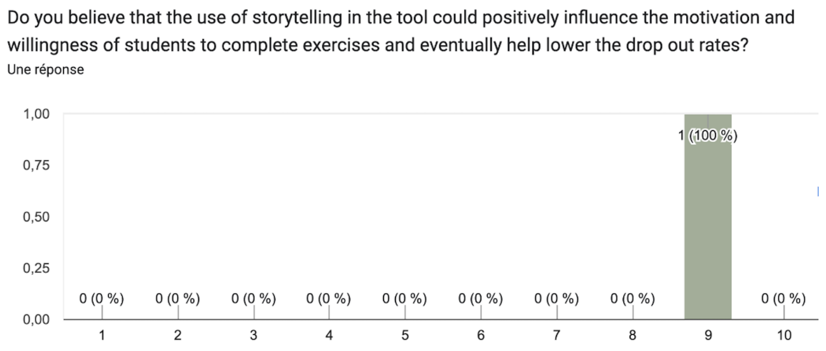


Figure B.4: Teacher responses to Question 3.

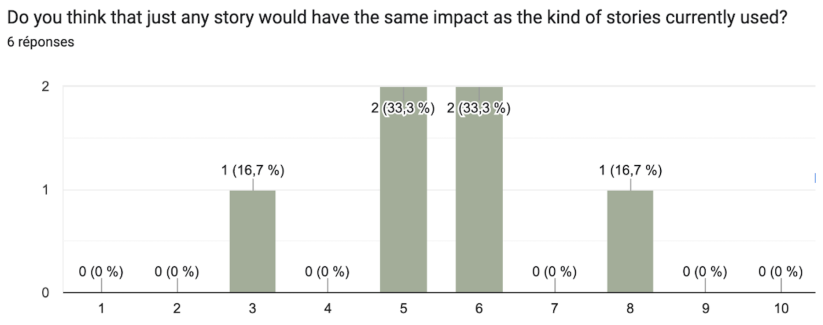


Figure B.5: Student responses to Question 4.

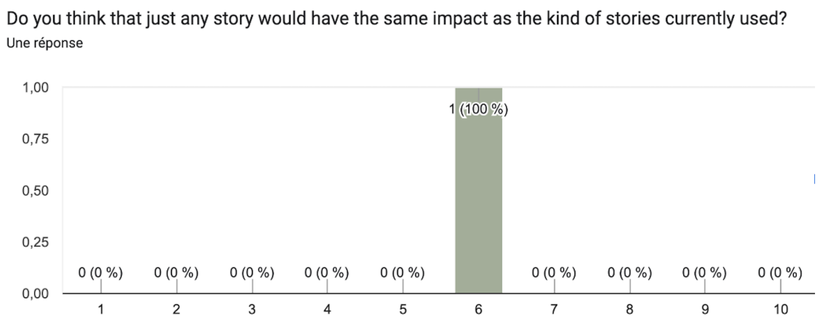


Figure B.6: Teacher responses to Question 4.

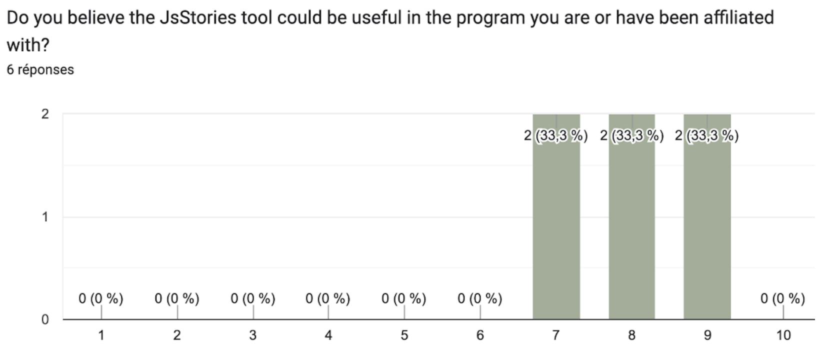


Figure B.7: Student responses to Question 5.

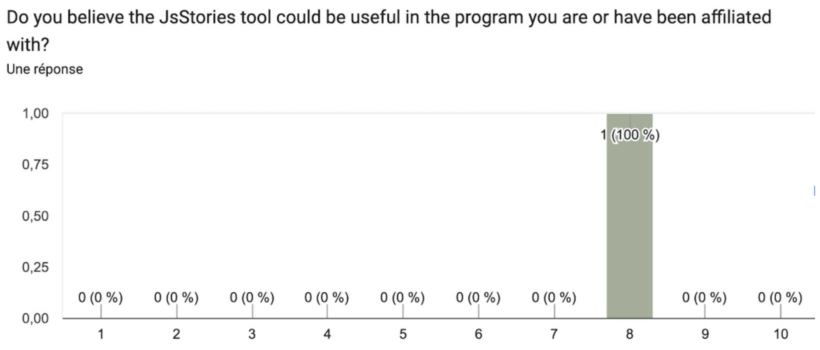


Figure B.8: Teacher responses to Question 5.

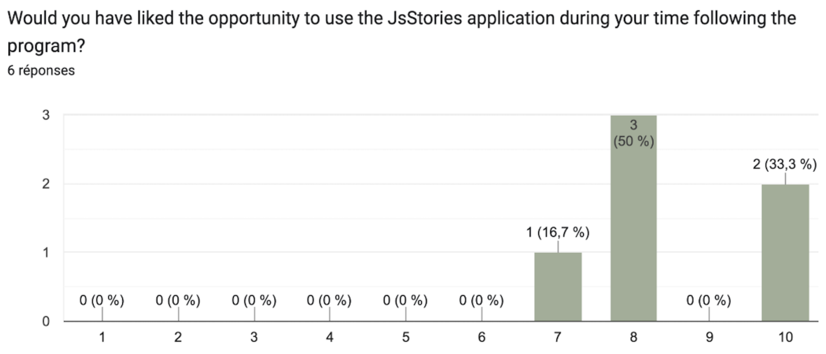


Figure B.9: Question 6.

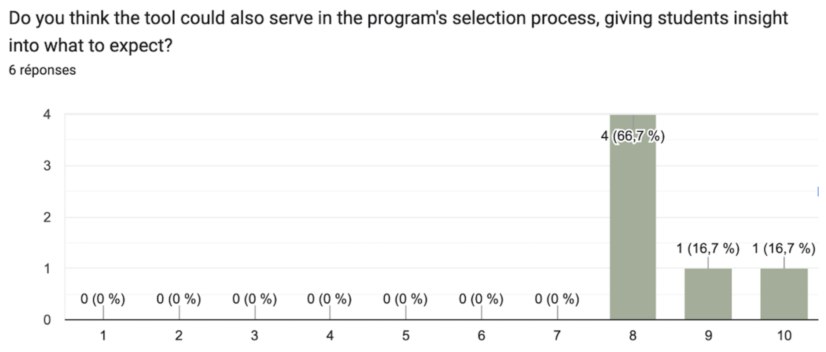


Figure B.10: Student responses to Question 7.

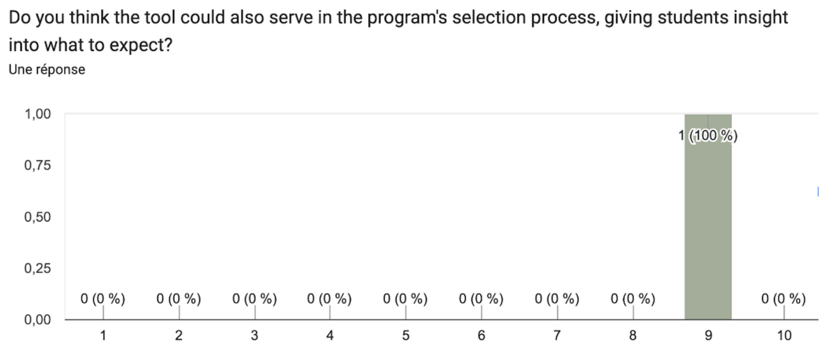


Figure B.11: Teacher responses to Question 7.

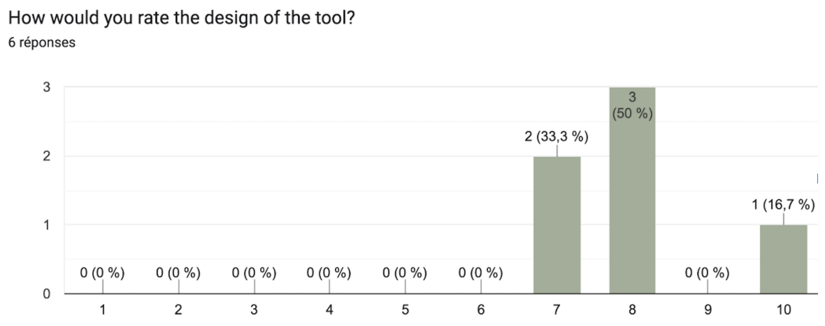


Figure B.12: Student responses to Question 8.

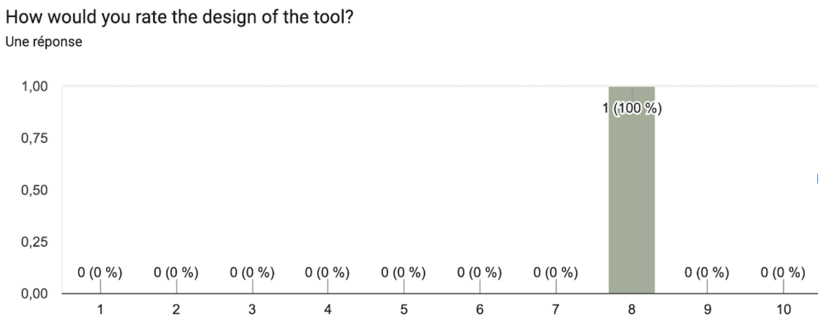


Figure B.13: Teacher responses to Question 8.

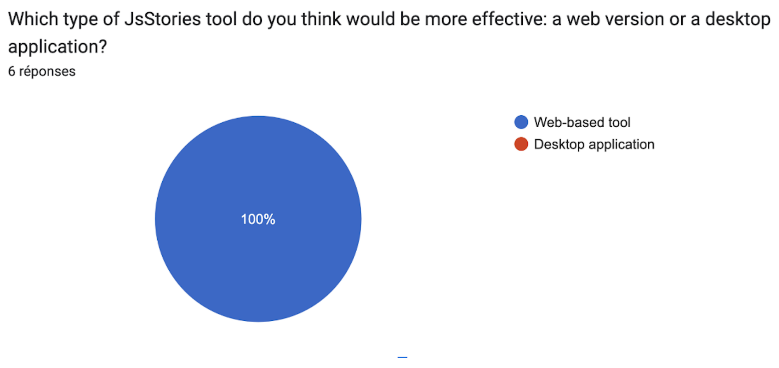


Figure B.14: Student responses to Question 9.

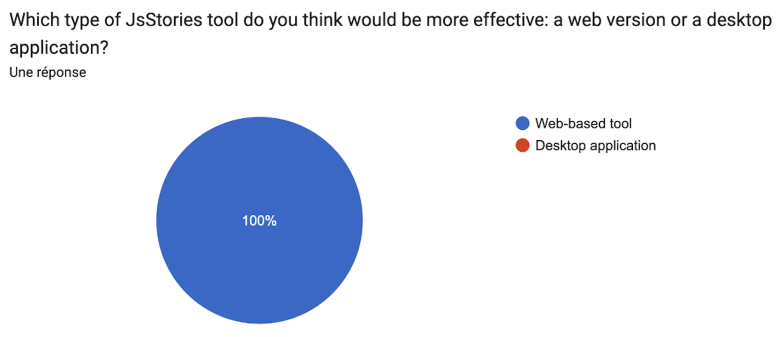
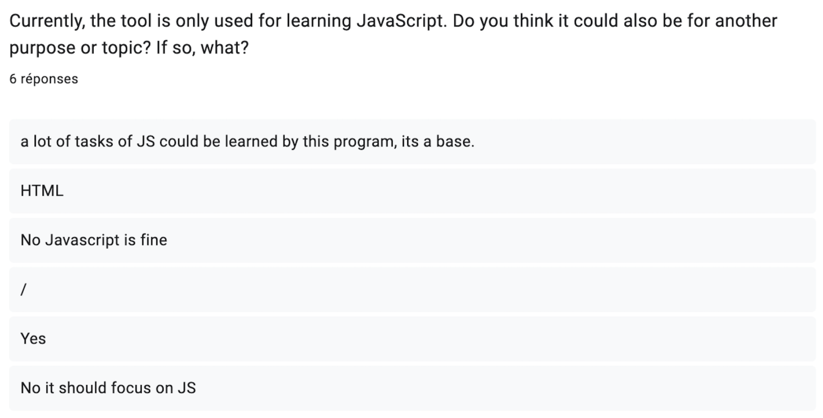

Which type of JsStories tool do you think would be more effective: a web version or a desktop application?
Une réponse

Figure B.15: Teacher responses to Question 9.


Currently, the tool is only used for learning JavaScript. Do you think it could also be for another purpose or topic? If so, what?
6 réponses

a lot of tasks of JS could be learned by this program, its a base.

HTML

No Javascript is fine

/

Yes

No it should focus on JS

Figure B.16: Question 10.

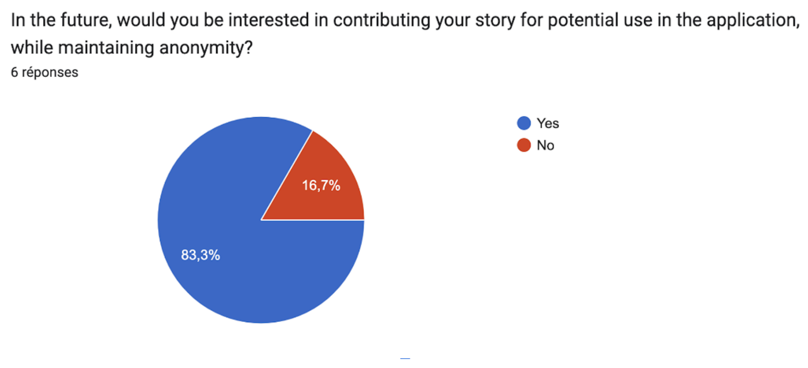
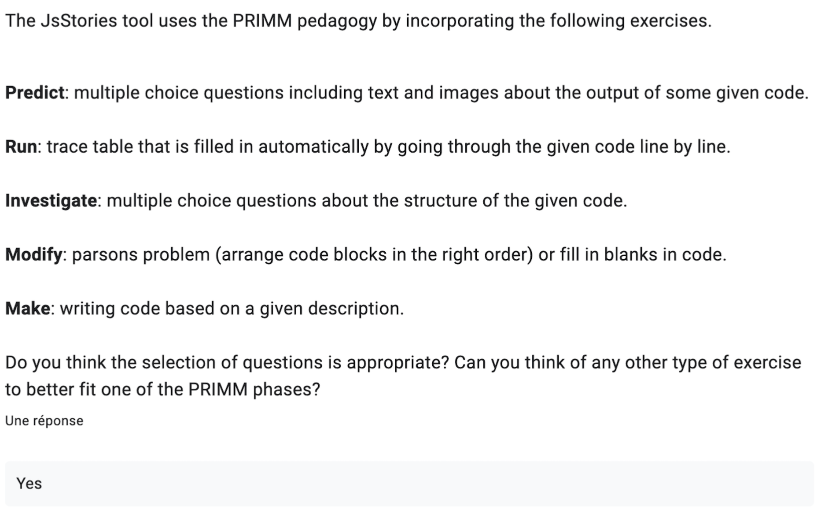

In the future, would you be interested in contributing your story for potential use in the application, while maintaining anonymity?
6 réponses

Figure B.17: Question 11.



The JsStories tool uses the PRIMM pedagogy by incorporating the following exercises.

**Predict**: multiple choice questions including text and images about the output of some given code.

**Run**: trace table that is filled in automatically by going through the given code line by line.

**Investigate**: multiple choice questions about the structure of the given code.

**Modify**: parsons problem (arrange code blocks in the right order) or fill in blanks in code.

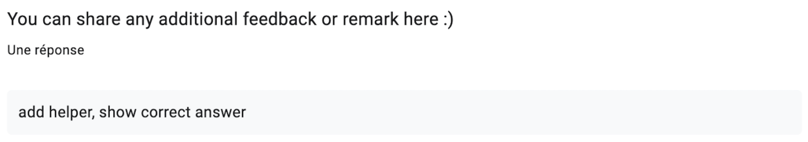
**Make**: writing code based on a given description.

Do you think the selection of questions is appropriate? Can you think of any other type of exercise to better fit one of the PRIMM phases?

Une réponse

Yes

Figure B.18: Question 12.

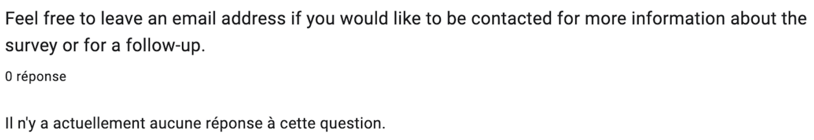


You can share any additional feedback or remark here :)

Une réponse

add helper, show correct answer

Figure B.19: Feedback section.



Feel free to leave an email address if you would like to be contacted for more information about the survey or for a follow-up.

0 réponse

Il n'y a actuellement aucune réponse à cette question.

Figure B.20: Contact section.

# Appendix C

# Screenshots

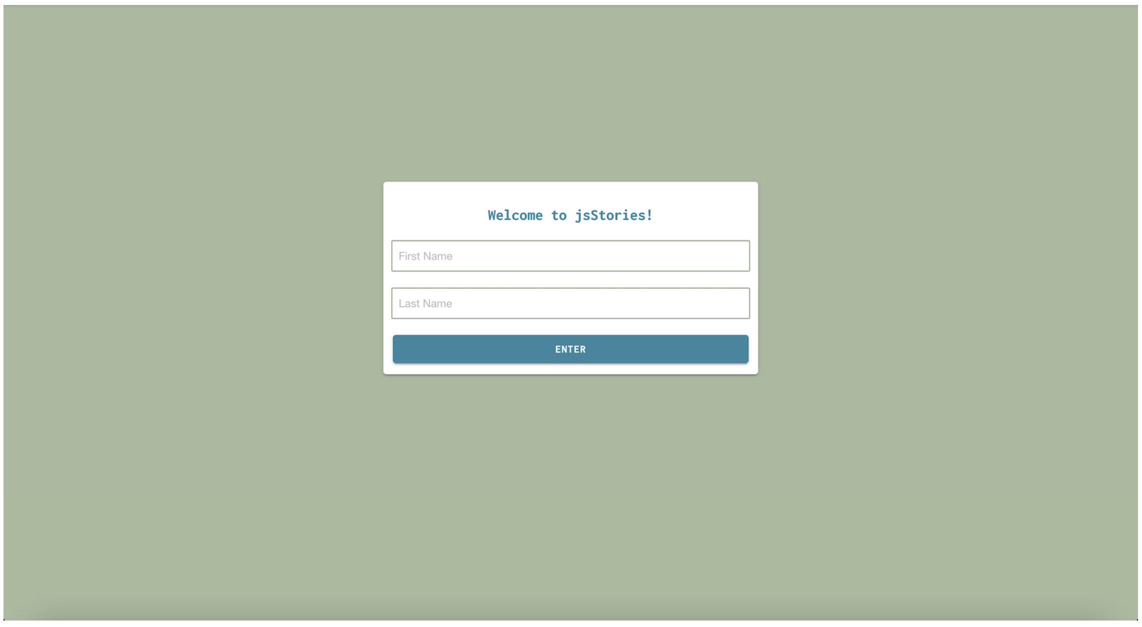
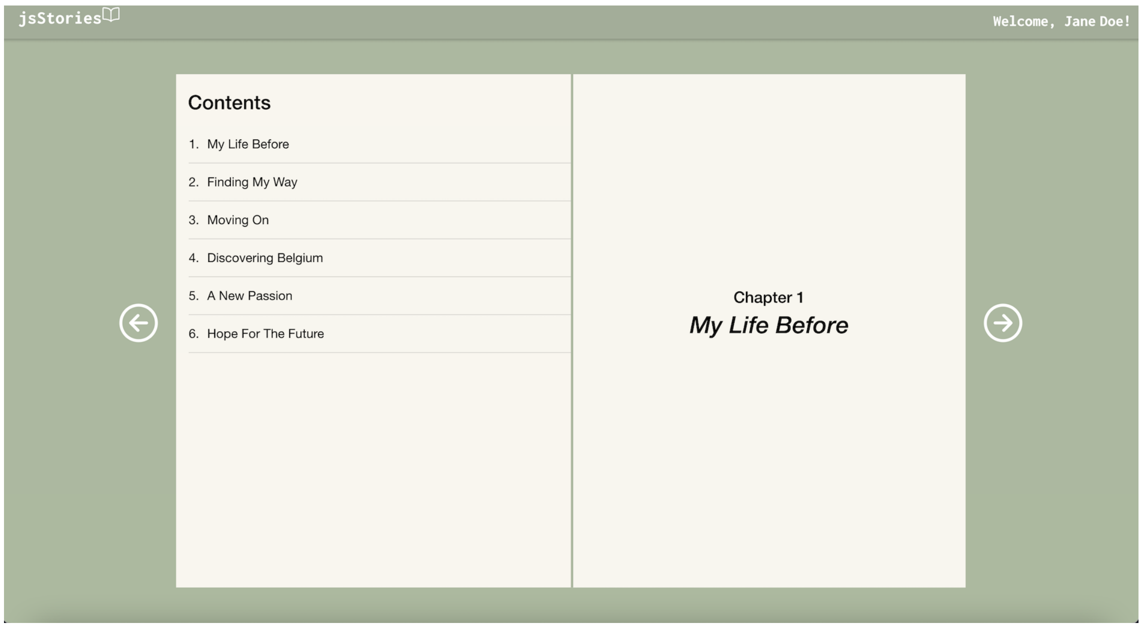


Figure C.1: The sign up page

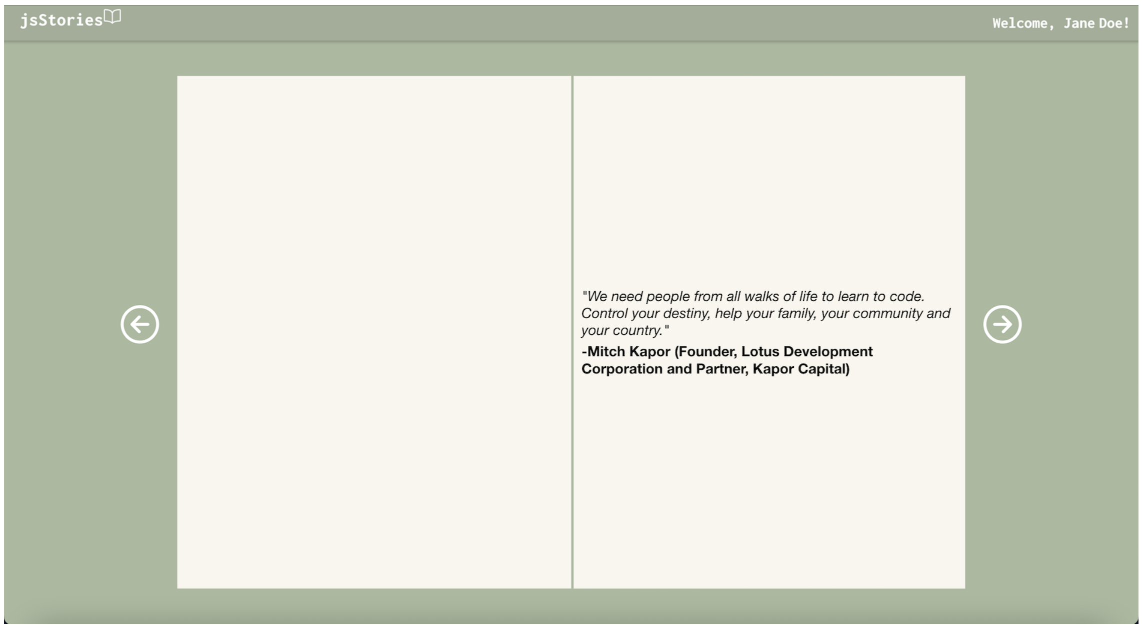


Contents

1. My Life Before
2. Finding My Way
3. Moving On
4. Discovering Belgium
5. A New Passion
6. Hope For The Future

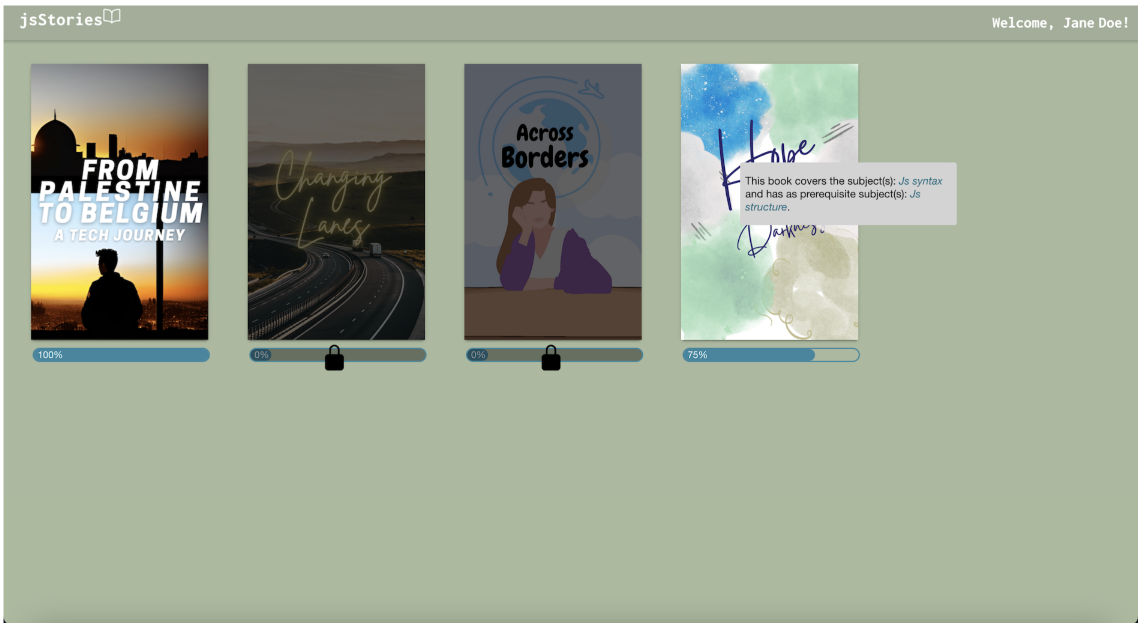
Chapter 1
*My Life Before*

Figure C.2: The contents page in a book



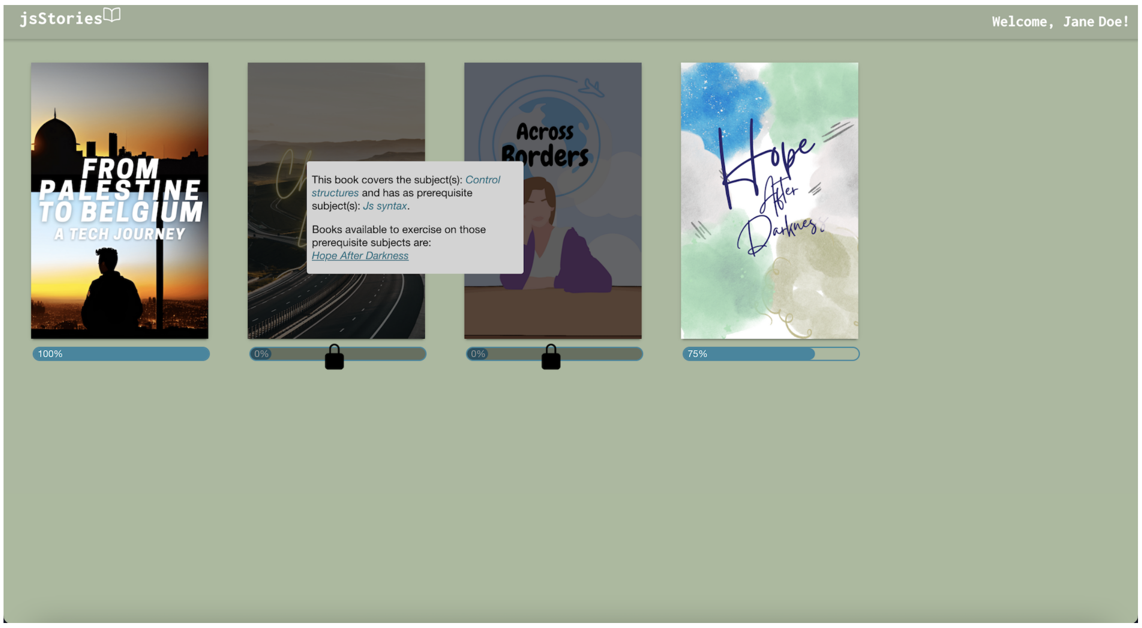
*"We need people from all walks of life to learn to code. Control your destiny, help your family, your community and your country."*
**-Mitch Kapor (Founder, Lotus Development Corporation and Partner, Kapor Capital)**

Figure C.3: The quote at the end of a book



This book covers the subject(s): *Js syntax* and has as prerequisite subject(s): *Js structure*.

Figure C.4: Tooltip info of an unlocked book



This book covers the subject(s): *Control structures* and has as prerequisite subject(s): *Js syntax*.

Books available to exercise on those prerequisite subjects are:
*Hope After Darkness*

Figure C.5: Tooltip info of a locked book

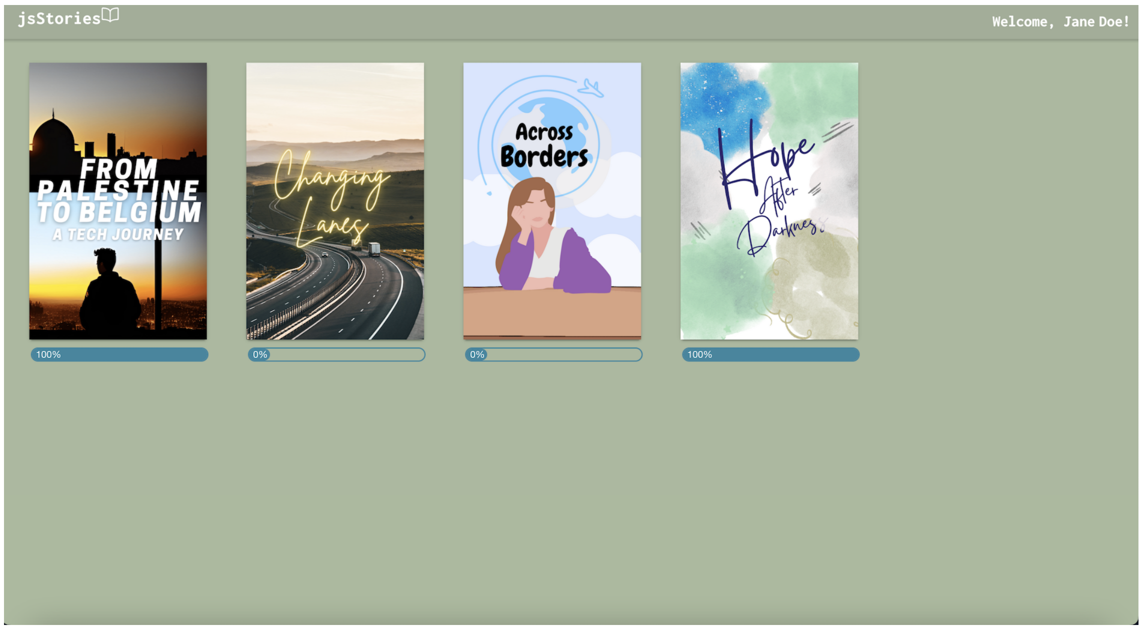


Figure C.6: All books being unlocked

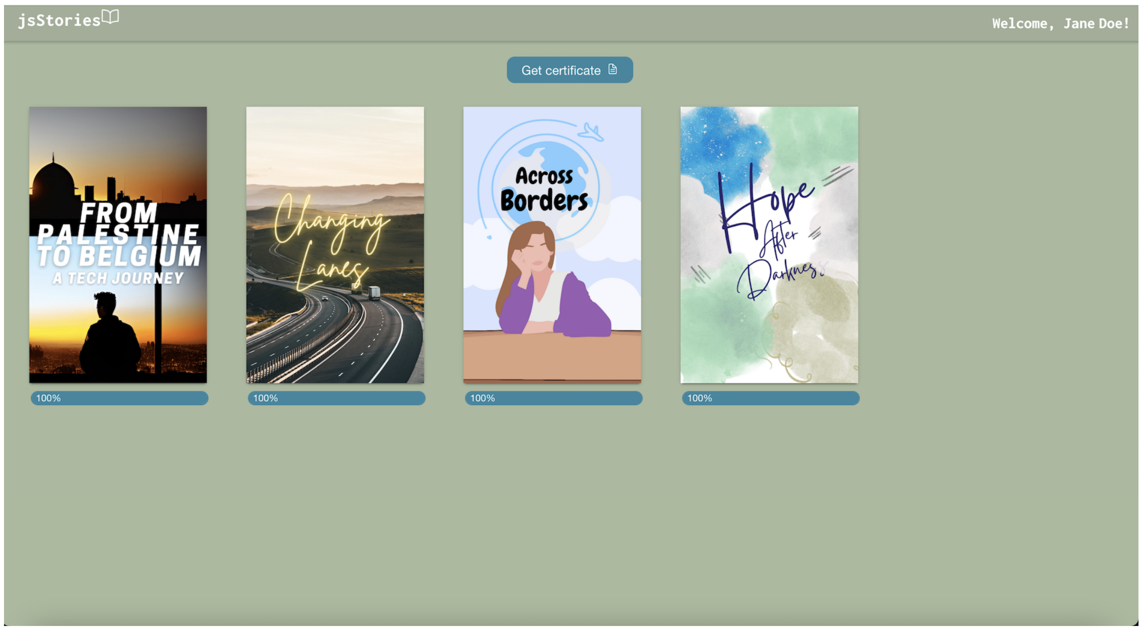
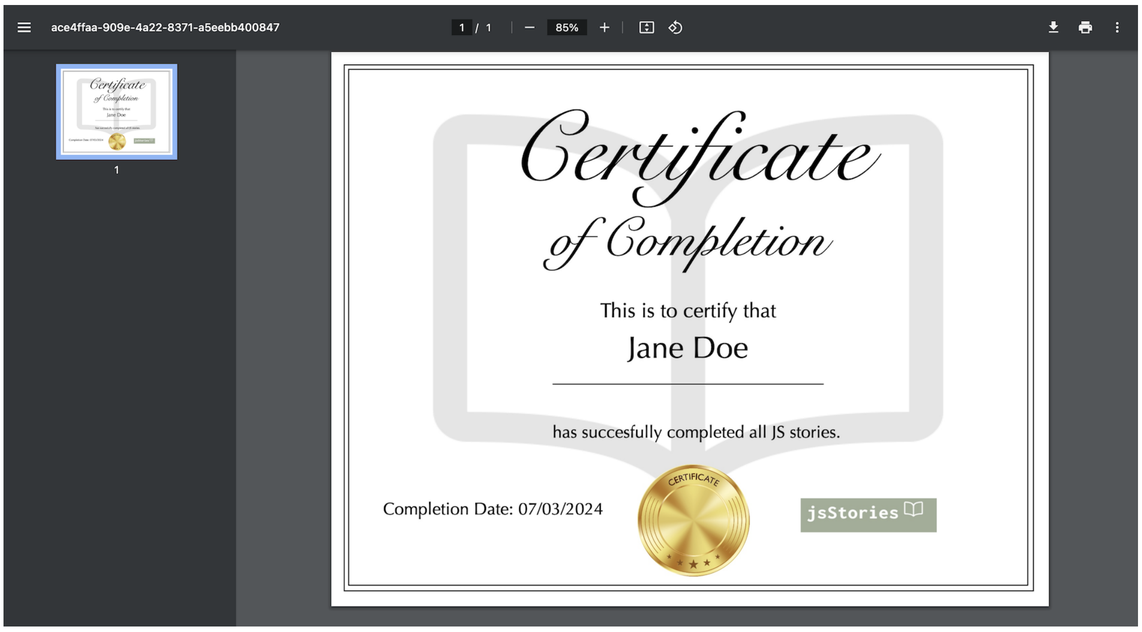


Figure C.7: Button to generate a certificate


ace4ffaa-909e-4a22-8371-a5eebb400847

*Certificate*

*of Completion*

This is to certify that

Jane Doe

has succesfully completed all JS stories.

Completion Date: 07/03/2024

Figure C.8: The generated certificate

# Bibliography

[1] Nancy E Adams. “Bloom’s Taxonomy of Cognitive Learning Objectives”. In: *Journal of the Medical Library Association* 103.3 (July 2015), p. 152. DOI: `https://doi.org/10.3163/1536-5050.103.3.010`.

[2] Lorin W Anderson and David R Krathwohl. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives*. Longman, 2001.

[3] Katrin Becker. “What’s the Difference Between Gamification, Serious Games, Educational Games, and Game-Based Learning”. In: *Academia Letters* 209 (Jan. 2021), pp. 1–4. DOI: `https://doi.org/10.20935/AL209`.

[4] Paul Black and Dylan Wiliam. *Inside the Black Box: Raising Standards Through Classroom Assessment*. Granada Learning, 1998.

[5] Benjamin S Bloom et al. *Taxonomy of Educational Objectives, Handbook I: Cognitive Domain*. Longman, 1956.

[6] Enzo Caminotti and Jeremy Gray. “The Effectiveness of Storytelling on Adult Learning”. In: *Journal of Workplace Learning* 24.6 (Aug. 2012), pp. 430–438. DOI: `https://doi.org/10.1108/13665621211250333`.

[7] Chin Soon Cheah. “Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review”. In: *Contemporary Educational Technology* 12.2 (Oct. 2020), p. 272. DOI: `https://doi.org/10.30935/cedtech/8247`.

[8] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. “Evaluating a New Exam Question: Parsons Problems”. In: *Proceedings of the 4th International Workshop on Computing Education Research*. Sydney, Australia, Sept. 2008, pp. 113–124. DOI: `https://doi.org/10.1145/1404520.1404532`.

[9] Sebastian Deterding et al. “From Game Design Elements to Gamefulness: Defining Gamification”. In: *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. Tampere, Finland, Sept. 2011, pp. 9–15. DOI: `https://doi.org/10.1145/2181037.2181040`.

[10] Lisa Ehrlinger and Wolfram Wöß. “Towards a Definition of Knowledge Graphs”. In: *SEMANTICS: Posters and Demos* 48.1-4 (Sept. 2016), p. 2.

[11] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. “Solving Parsons Problems Versus Fixing and Writing Code”. In: *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. Koli, Finland, Nov. 2017, pp. 20–29. DOI: `https://doi.org/10.1145/3141880.3141895`.

[12] Mary Forehand. "Bloom's Taxonomy: Original and Revised". In: *M. Orey (Ed.), Emerging Perspectives on Learning, Teaching, and Technology* (2005).

[13] Jonathan B Freeman. "Measuring and Resolving LGBTQ Disparities in STEM". In: *Policy Insights from the Behavioral and Brain Sciences* 7.2 (Oct. 2020), pp. 141–148. DOI: `https://doi.org/10.1177/2372732220943232`.

[14] Krista Graham. "TechMatters: Let's "Go Formative": Exploring a Digital Platform to Enhance Teaching and Learning". In: *LOEX Quarterly* 44.3 (2017), pp. 4–6.

[15] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does Gamification Work? A Literature Review of Empirical Studies on Gamification". In: *Proceedings of the 47th Hawaii International Conference on System Sciences*. Waikoloa, USA, Jan. 2014, pp. 3025–3034. DOI: `https://doi.org/10.1109/HICSS.2014.377`.

[16] *Hand-out Interview Techniques (long version)*. `https://skillslab.tue.nl/pathtoimg.php?id=51`. 2023.

[17] Aidan Hogan et al. "Knowledge Graphs". In: *ACM Computing Surveys* 54.4 (July 2021), pp. 1–37. DOI: `https://doi.org/10.1145/3447772`.

[18] Cruz Izu et al. "Fostering Program Comprehension in Novice Programmers-learning Activities and Learning Trajectories". In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. Aberdeen, Scotland, Dec. 2019, pp. 27–52. DOI: `https://doi.org/10.1145/3344429.3372501`.

[19] Tony Jenkins. "On the Difficulty of Learning to Program". In: *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*. Loughborough, UK, Aug. 2002, pp. 53–58.

[20] Shaoxiong Ji et al. "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.2 (Feb. 2021), pp. 494–514. DOI: `https://doi.org/10.1109/TNNLS.2021.3070843`.

[21] Shulamit Kahn and Donna Ginther. *Women and STEM*. Tech. rep. National Bureau of Economic Research, June 2017.

[22] Caitlin Kelleher. "Motivating Programming: Using Storytelling to Make Computer Programming Attractive to Middle School Girls". PhD thesis. Pittsburgh, USA: Carnegie Mellon University, School of Computer Science, Nov. 2006.

[23] Caitlin Kelleher and Randy Pausch. "Using Storytelling to Motivate Programming". In: *Communications of the ACM* 50.7 (July 2007), pp. 58–64. DOI: `https://doi.org/10.1145/1272516.1272540`.

[24] Irene Lee et al. "Computational Thinking for Youth in Practice". In: *ACM Inroads* 2.1 (Feb. 2011), pp. 32–37. DOI: `https://doi.org/10.1145/1929887.1929902`.

[25] Teemu Lehtinen, Lassi Haaranen, and Juho Leinonen. "Automated Questionnaires About Students' JavaScript Programs: Towards Gauging Novice Programming Processes". In: *Proceedings of the 25th Australasian Computing Education Conference*. Melbourne, Australia, Jan. 2023, pp. 49–58. DOI: `https://doi.org/10.1145/3576123.3576129`.

[26] Teemu Lehtinen, André L Santos, and Juha Sorva. "Let's Ask Students About Their Programs, Automatically". In: *Proceedings of IEEE/ACM 29th International Conference on Program Comprehension*. Madrid, Spain, May 2021, pp. 467–475. DOI: `https://doi.org/10.1109/ICPC52881.2021.00054`.

[27] Raymond Lister, Colin Fidge, and Donna Teague. “Further Evidence of a Relationship Between Explaining, Tracing and Writing Skills in Introductory Programming”. In: *ACM SIGCSE Bulletin* 41.3 (July 2009), pp. 161–165. DOI: `https://doi.org/10.1145/1562877.1562930`.

[28] Raymond Lister et al. “A Multi-national Study of Reading and Tracing Skills in Novice Programmers”. In: *ACM SIGCSE Bulletin* 36.4 (June 2004), pp. 119–150. DOI: `https://doi.org/10.1145/1044550.1041673`.

[29] LM van der Lubbe et al. “Bridging the Computer Science Teacher Shortage with a Digital Learning Platform”. In: *Proceedings of the 15th International Conference on Computer Supported Education*. Prague, Czech Republic, Apr. 2023, pp. 289–296. DOI: `https://doi.org/10.5220/0011971900003470`.

[30] Jenni Majuri, Jonna Koivisto, and Juho Hamari. “Gamification of Education and Learning: A Review of Empirical Literature”. In: *Proceedings of the 2nd international GamiFIN conference*. Pori, Finland, May 2018, pp. 11–19.

[31] Yoshi Malaise. *iSport: Towards A Smart Trainer Assitant*. Master’s thesis, Vrije Universiteit Brussel. June 2021.

[32] Yoshi Malaise and Beat Signer. “Explorotron: An IDE Extension for Guided and Independent Code Exploration and Learning”. In: *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. Koli, Finland, Nov. 2023, pp. 13–18. DOI: `https://doi.org/10.1145/3631802.3631816`.

[33] Michael McCracken et al. “A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students”. In: *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*. Canterbury, UK, Dec. 2001, pp. 125–180. DOI: `https://doi.org/10.1145/572133.572137`.

[34] Barbara Moskal, Deborah Lurie, and Stephen Cooper. “Evaluating the Effectiveness of a New Instructional Approach”. In: *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. Norfolk, USA, Mar. 2004, pp. 75–79. DOI: `https://doi.org/10.1145/1028174.971328`.

[35] Fiona Fui-Hoon Nah et al. “Gamification of Education: A Review of Literature”. In: *Proceedings of the 1st International Conference on HCI in Business*. Crete, Greece, June 2014, pp. 401–409. DOI: `https://doi.org/10.1007/978-3-319-07293-7_39`.

[36] Sylvia C Nassar-McMillan et al. “New Tools for Examining Undergraduate Students’ STEM Stereotypes: Implications for Women and Other Underrepresented Groups”. In: *New Directions for Institutional Research* 2011.152 (Dec. 2011), pp. 87–98. DOI: `https://doi.org/10.1002/ir.411`.

[37] Dale Parsons and Patricia Haden. “Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses”. In: *Proceedings of the 8th Australasian Conference on Computing Education*. Hobart, Australia, Jan. 2006, pp. 157–163.

[38] Ken Peffers et al. “A Design Science Research Methodology for Information Systems Research”. In: *Proceedings of 1st International Conference on Design Science Research in Information Systems and Technology*. Claremont, USA, Feb. 2006, pp. 83–106.

[39] Anthony Robins, Janet Rountree, and Nathan Rountree. “Learning and Teaching Programming: A Review and Discussion”. In: *Computer Science Education* 13.2 (Aug. 2010), pp. 137–172. DOI: `https://doi.org/10.1076/csed.13.2.137.14200`.

[40] Marsha Rossiter. “Narrative and Stories in Adult Teaching and Learning”. In: *ERIC Digest* (2002).

[41] Wolfgang Schnotz. “Reanalyzing the Expertise Reversal Effect”. In: *Instructional Science* 38.3 (Oct. 2010), pp. 315–323. DOI: `https://doi.org/10.1007/s11251-009-9104-y`.

[42] Carsten Schulte. “Block Model: An Educational Model of Program Comprehension as a Tool for a Scholarly Approach to Teaching”. In: *Proceedings of the 4th International Workshop on Computing Education Research.* Sydney, Australia, Sept. 2008, pp. 149–160. DOI: `https://doi.org/10.1145/1404520.1404535`.

[43] Sue Sentance, Jane Waite, and Maria Kallia. “Teachers’ Experiences of Using Primm to Teach Programming in School”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* Minneapolis, USA, Feb. 2019, pp. 476–482. DOI: `https://doi.org/10.1145/3287324.3287477`.

[44] Sue Sentance, Jane Waite, and Maria Kallia. “Teaching Computer Programming With PRIMM: A Sociocultural Perspective”. In: *Computer Science Education* 29.2-3 (Apr. 2019), pp. 136–176. DOI: `https://doi.org/10.1080/08993408.2019.1608781`.

[45] Beat Signer and Moira C Norrie. “As We May Link: A General Metamodel for Hypermedia Systems”. In: *Proceedings of ER 2007, International Conference on Conceptual Modeling.* Auckland, New Zealand, Nov. 2007, pp. 359–374. DOI: `https://doi.org/10.1007/978-3-540-75563-0_25`.

[46] Amit Singhal. *Introducing the Knowledge Graph: Things, Not Strings*. May 2012. URL: `https://blog.google/products/search/introducing-knowledge-graph-things-not/`.

[47] John Sweller et al. “The Expertise Reversal Effect”. In: *Educational Psychologist* 38.1 (2003), pp. 23–31. DOI: `https://doi.org/10.1007/978-1-4419-8126-4_12`.

[48] Christopher Watson and Frederick WB Li. “Failure Rates in Introductory Programming Revisited”. In: *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education.* Uppsala, Sweden, June 2014, pp. 39–44. DOI: `https://doi.org/10.1145/2591708.2591749`.

[49] Jacqueline Whalley and Nadia Kasto. “Revisiting Models of Human Conceptualisation in the Context of a Programming Examination”. In: *Proceedings of the 15th Australasian Computing Education Conference.* Adelaide, Australia, Jan. 2013, pp. 67–76. DOI: `https://dl.acm.org/doi/abs/10.5555/2667199.2667207`.

[50] Xiaohan Zou. “A Survey on Application of Knowledge Graph”. In: *Journal of Physics: Conference Series* 1487.1 (Jan. 2020). DOI: `https://doi.org/10.1088/1742-6596/1487/1/012016`.