Learning experiences of Haaga-Helia's BITe students regarding JavaScript and its libraries and frameworks in 2019

Claudio Rodríguez

Bachelor's Thesis
Degree Programme in Business Information Technology
2020

Haaga-Helia
University of Applied Sciences

Abstract

Date 17.05.2020

Author
Claudio Rodríguez

Degree programme
Business Information Technology

Report/thesis title	**Number of pages and appendix pages**
Learning experiences of Haaga-Helia's BITe students regarding JavaScript and its libraries and frameworks in 2019	**47 + 5**

The advancements of technology and easy access to internet across the world has accelerated the changes and improvements of programming tremendously. Nowadays there are big communities of people contributing to the further development of programming languages. JavaScript has abruptly managed to make programming accessible and become one of the most popular programming languages nowadays.

Nonetheless the current fast pace at which technologies evolve has also increased the speed at which some programming languages and their environments must change and adapt. This brings new features and extra complexity to these technologies and therefore, developers and IT students must constantly continue learning the use new tools and methods, which can be quite demanding and possibly lead to stressful experiences.

The goal of this thesis was to investigate the learning experiences of Haaga-Helia's Business Information Technology students regarding JavaScript, its libraries and frameworks and their feelings towards these technologies. Also, another objective was to find out which aspects of the teaching of JavaScript in Haaga-Helia has been most useful to their leaning path and which aspects of the teaching they feel that could be improved.

In order to gather data for this research an electronic survey was conducted were the students could give background information about their JavaScript related studies at Haaga-Helia and their feelings regarding the use of these technologies. Additionally, a couple of semi-structured interviews were carried out on soon to graduate students who had already some working experience as developers, which helped to find what had been most useful from their learning experience at Haaga-Helia and possible improvement suggestions.

Keywords
JavaScript, Programming Learning, JavaScript Frameworks, JavaScript Libraries, ECMAScript.

Table of contents

1 Introduction 1
  1.1 Target group 1
  1.2 Research questions, objectives and scope 2
2 Theoretical framework 3
  2.1 Computers and programming 3
  2.2 Learning to program 4
    2.2.1 MOOCs and other tools as programming learning sources 5
  2.3 Programming languages 6
  2.4 JavaScript 7
    2.4.1 Why JavaScript? 9
    2.4.2 JavaScript and ECMA standard versions 10
    2.4.3 Browser compatibility 11
  2.5 Current popular JavaScript frameworks & libraries 13
    2.5.1 React 13
    2.5.2 Angular & Vue.js 14
    2.5.3 jQuery 16
  2.6 Memes and what they can reveal to us 17
3 Research Method 18
  3.1 Electronic surveys 18
  3.2 Semi-structured interviews 20
4 Results 21
  4.1 Survey 21
    4.1.1 Background Information section results 21
    4.1.2 Students' learning experiences of *JavaScript* 25
    4.1.3 Meme & closing sections' results 30
  4.2 Semi-structured interview results 37
5 Discussion 40
  5.1 Comparison to initial assumptions and research questions 40
    5.1.1 Question 1: Are BITe students discouraged in any way to continue learning and developing professionally in JavaScript related fields? 40
    5.1.2 Question 2: What are BITe students' feelings regarding JavaScript, its frameworks and libraries? 41
    5.1.3 Question 3: What possible improvements there could be on the JavaScript teaching methods in Haaga-Helia for BITe students? 42
  5.2 New information and final conclusions 43
References 45
Appendices 48

Appendix 1. Survey Questions 48
Appendix 2. Prize winner selection (Code snippet).. 50
Appendix 3. Survey transcripts (Confidential)... 51
Appendix 4. Semi-structured interview scripts (Confidential) 52

1 Introduction

The topic for this thesis is inspired by personal observations throughout my studies at Haaga-Helia of fellow classmates and myself. I have observed changes in confidence and motivation, feelings of fear of failure and uncertainty, as we, the students, progressed deeper into our programming studies.

These observations made me wonder which factors were affecting IT students and their learning curve when it comes to programming and learning new programming tools and methods. I wanted to find out how much of what I had observed during my studies was simply my own personal perceptions or if there were any generalized issues in the way we were taught and learned. Uncovering the world of programming (particularly JavaScript in the case of this thesis) can be challenging and discouraging at times but through further exploration of these issues, some light can be shed on possible ways to reduce the challenges of learning to program.

1.1 Target group

In order to avoid excessive generalization, acquire data that could be easier to analyse and narrow down the size of this research, the target group of this research is limited to the Business Information Technology (BITe) students of Haaga-Helia University of Applied Sciences. Information Technology (IT) students from other Universities and Institutions, as well as Haaga-Helia students of the Finnish language (TIKO) equivalent to BITe, are excluded from the study also. This is because there are differences in the course offerings among degree programmes and I want to focus on the BITe students because I have experienced the studies at BITe degree programme myself.

The BITe students included in this study consist of Haaga-Helia students that have started their BITe studies between 2015 and 2019. Although the results of this study can be more useful for students at the start of their degree programme, students at the end of their studies are also targeted. Their wider experience of the studies and their possible academic experience can prove extremely valuable, as they can provide useful insight of the different stages of their studies in the BITe programme and of how they have been able to apply their studies in working life.

1.2 Research questions, objectives and scope

This research aims to explore and answer the following questions:

- Are BITe students discouraged in any way to continue learning and developing professionally in JavaScript related fields?
- What are BITe students' feelings regarding JavaScript, its frameworks and libraries?
- What possible improvements there could be on the JavaScript teaching methods in Haaga-Helia for BITe students?

This project's objective is mainly to get information from Haaga-Helia's English IT degree programme students regarding their learning of JavaScript (JS), including what challenges they might have faced and what kind of feelings they have experienced when further learning the language, its libraries and frameworks (L&F). In addition to that, the objective of this thesis is to find possible inefficiencies on the way JavaScript has been and is introduced to students (from their perspective), as well as what aspects of the teaching that have been helpful and valuable, but also finding out what the students wish could be done differently.

In order to get qualitative data information, surveys will be sent to Haaga-Helia's BITe students, who have initiated their studies at different years. In addition to that, semi-structured interviews will be carried out to a couple of students who are already working on the IT field and preferably still working with JavaScript in some form.

This thesis will provide information based on the students reflections in 2019 about their learning path and the effects that learning JavaScript with its libraries and frameworks has had on them, along with possible areas of improvement on how these topics could be introduced and taught to students in the future.

Languages other than JavaScript will be kept out of the scope, in order to narrow down the scope and to avoid unnecessarily overcomplicating the research. Another motive to concentrate purely on JavaScript is that there are many BITe courses based on JavaScript and consequently it is the language I have most knowledge and interest of myself. In addition, I chose to focus on JavaScript since it is one of the most, if not the most, popular programming languages globally. For the above-mentioned reasons, frameworks and libraries from languages other than JavaScript are also kept out of the scope of this thesis.

2 Theoretical framework

As technology and automation keep evolving, there is a resurging and increasing fear of machines replacing jobs in a wide range of industries. Although it is an understandable fear, there is great potential for a bright future as labour-savings and process innovations, should in turn decrease prices and help create a new kind of labour demand (Piva & Vivarelli 2018). As technology keeps evolving and spreading to different areas, there will be great need of people to carry out these changes. An example of this is the need to program and maintain these new technological innovations as their use keeps increasing.

In the future people will be increasingly taught about programming earlier in life. Already in the present, there are different resources to accomplish such feat. This future situation is emphasised on Bryson Payne's introduction to his book "Teach Your Kids to Code":

> "Computer programming, or coding, is a crucial skill every child should be learning." (Payne 2015)

Being introduced to the basics of computer science and programming from an early age, could make it easier to understand more complicated concepts and methodologies for future programmers. But not only that, as it could also make future generations less afraid of experimenting with their code and that way, develop a better intuition when it comes to programming.

In contrast, many of us who have learned about programming as adults or late-adolescents, might have a harder time learning more advanced programming concepts. But fortunately, there are different ways to learn new concepts and skills, independent of the person's age and the same holds true for programming.

2.1 Computers and programming

Merriam-Webster defines computers as "a programmable usually electronic device that can store, retrieve, and process data" (Merriam-Webster Online 2019), a definition with only a few key elements that combined have an immense power to lead our society's prosperity.

Humans have excelled at making their working processes more efficient and improving the results of their actions. Eventually we developed machines to help us create greater output with a smaller input, up to the point we were able to manipulate energy sources other than human or animal power. We revolutionized the world with the development of steam powered machines, electrical devices and specially when computers came around.

Although the development of computers starting secretly during the horrific World War II (Piva & Vivarelli 2018), its use and purpose was diversified to unimaginable extents, being now essential in the normal functioning of our current society.

Programming allows humans to interact with computers and other electronic devices, such as phones, smart watches, smart door locks, irrigation systems, thermostats, etc. This is done by writing instructions (code) that the computer needs to interpret in order to carry out the desired action. Although it might sound simple, it really is not, and great amounts of work and research has led to the simplicity of today's computer interaction.

Nowadays the average person uses programs to interact with computers or other smart devices. These programs in turn, have been programmed with the use of logic and programming languages so the users can easily interact with them. As the role of computers keeps being more entangled with humanity's daily life, it becomes more important for people to learn about the world of computers, as they are also a useful tool for varied disciplines. Future generations will be expected to be trained in Computer Science from an early age and that way become better productive members of society. (Wakil, Khdir, Sabir & Nawzad 2019)

2.2 Learning to program

As the widespread of computers is fairly new, competences such as Computational Thinking have not been greatly developed by older generations, but the interest and need to learn programming are not exclusive for new generations either. However, people of varied age groups and backgrounds may have varied cognitive abilities when learning to program, thus the same teaching methods won't be the best for every individual. (Hsu, Chang & Hung 2018)

Mark Guzdial's '*What's the Best Way to Teach Computer Science to Beginners?'* article questions the effectivity of traditional programming teaching methods to newcomers. Teaching theory to students and having them trying to program and figure things out on their own has not been the most effective way to teach people with no previous experience. Some studies have found that students who were first shown how a programming problem has been solved, were more successful when it was their turn to try on their own. (Guzdial 2015)

This issue is also reflected in a case, where in an effort to make Android programming introduction easier to newcomers, the easy to use software *App Inventor* for app creation was introduced to students. App Inventor was easily adopted by the students, but its simplicity made some students work automatically, without properly thinking about their tasks and learning. In this case it was also emphasized how there is simply no tool or method that would satisfy everyone's needs at once. (Robertson 2014)

Project-based, problem-based, game-based and cooperative learning are some ways in which computer science can be taught (Hsu, Chang & Hung 2018), but regardless of the option, there is no single option that can be the best way of teaching it on its own. It is for that reason that a mixed approach could bring more positive results.

2.2.1 MOOCs and other tools as programming learning sources

In contrast to some decades ago, where access to computers was limited to few individuals and only experts with great knowledge of computers' assembly and use, in the present the scenery has changed completely. During the last decades people went from learning about computers and programming through military and few academic institutions (Piva & Vivarelli 2018), to having a plethora of learning sources available to them. The way of learning about programming expanded from lessons in classrooms or computer laboratories and textbooks, to easily access a myriad of learning material of all kinds, thanks to the internet. This way regardless of your physical location or the presence of an instructor learning is easier than ever.

The emergence of Massive Online Open Courses (MOOCs) in 2012, became a great opportunity not only to add other learning sources to be used at a personal pace, but even to be used without going alongside a traditional university degree. Although, what constitutes a MOOC is not officially defined. They are commonly known as online courses that can be available for a great amount of people. They might be open courses in terms of being free of charge or having little to no barriers to join the course (such as prerequisites). Another of their characteristics is usually having their own community of students or users, were members can discuss subjects or ask or provide support in the learning process. (Haber 2014)

Thanks to the easy access to computers and the internet, there are plenty of useful free resources available. Instead of joining a MOOC, one can look for a small specific subject instead. In many cases, YouTube can easily provide a wide range of tutorials of varied lengths, where instructors can also advertise more complete versions of their tutorials hosted in sites like Udemy or may recommend other sources like, Medium articles, freeCodeCamp or Zenva, etc.

Obviously, each approach has its limitations as I have come to experience during my own studies. In educational institutions there could be a teacher who's teaching method doesn't match with one's most optimal learning style, which could cause one to lose interest in an otherwise interesting or useful subject. On the other hand, using free tutorials, can give one more freedom to choose a more suitable learning style, but can pose a bigger risk of being outdated with current practices, lack of feedback with the instructor or even acquiring bad practices straight from the instructor.

Paid tutorial of the likes found in Frontend Masters, Udemy, freeCodeCamp, among others, may bring some accountability or greater chance of receiving feedback or interacting with a community. But in some cases, the progress through the course could be at such irregular intervals that you may never complete some courses, or it may become outdated over time. Some of these courses may stay updated with current standards and practices, but for some others a new course may be created instead, which you would have to pay in order to access.

2.3 Programming languages

Without programming languages, programs would have had to be written in binary form (computer language), but thanks to programming languages, we can write programs in a way that is more easily understandable and achievable to successfully create, for humans writing the code and the ones maintaining it, in the future.

Although the code written in conventional programming languages, will still need to be transpired into computer language, it is still significantly less time consuming and error prone.

No matter which programming language you use, the need of algorithms is imperative for us to tell a computer what to do and how. Each program can have their own set of rules and limitations, but they all need the use of logic to function.

Merriam-Webster defines algorithm as “broadly: a step-by-step procedure for solving a problem or accomplishing some end” (Merriam-Webster Online 2019). This broad definition ties closely to the core purpose of programming.

Programming languages can be classified and distinguished from one another, through their features in what is referred as *programming paradigms*. The two most common programming paradigms are *imperative* and *declarative* programming. Imperative programming characterizes itself by instructing the machines how to change their state, while declarative programming by only the properties of the desired result, but not the way it will be computed. (Nørmark 2019)

A programming language does not need to be restricted to one programming paradigm, in fact many of the most common programming languages have become multi-paradigm. Such is the case of JavaScript, that can belong to functional (declarative sub-category), as well as object-oriented programming (imperative sub-category). (Wikipedia 2020)

2.4 JavaScript

JavaScript was developed in 1995 at Netscape Communications, in an effort to bring more flexibility and dynamism to the web, since html on its own was too narrow. This way Netscape got a competitive advantage over other web browser competitors. Netscape employed Brendan Eich to develop this new language and he wrote its first prototype on ten days. The first version was initially code-named Mocha, but its official name became LiveScript. This name would not last long, as in a strategic move it was decided to take advantage of Java’s great growth and popularity, resulting in yet another name change to what is now known as JavaScript. (Rauschmayer 2014)

Whether or not that name strategy was the best choice, JavaScript’s present popularity is undeniable, as shown in Figures 1 and 2, with the results of 2019’s *Stack Overflow Developer Survey*.

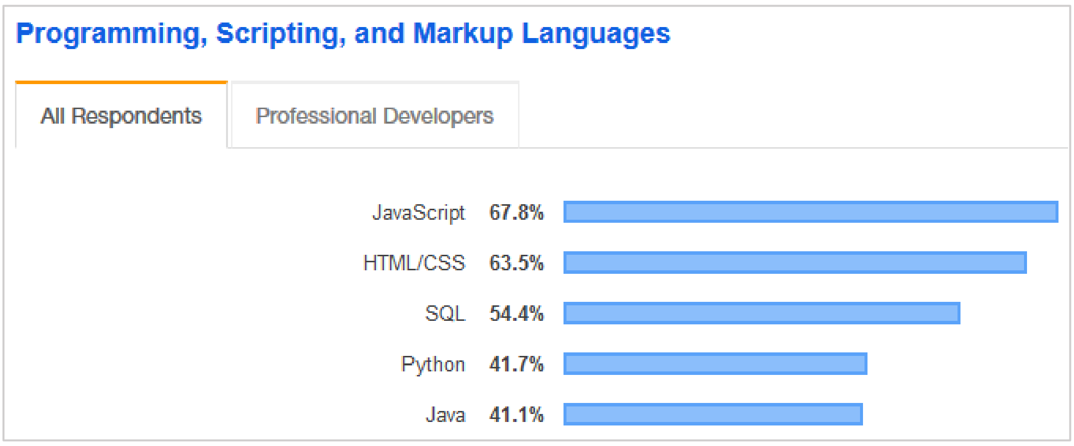


Figure 1. 2019’s Stack Overflow Developer Survey - Most popular languages, for all reponses (87354 responses).

While Figure 1 collects Stack Overflow users' preferences, without distinction of them being professionals or not, Figure 2 only regards the answers of professional developers. For both categories, JavaScript leads the statistics with almost 80% of the respondents declaring to use JavaScript. Even though the use of Stack Overflow is not imperative for developers or to be able to know JavaScript, resorting to Stack Overflow is an extremely popular practice on the field of programming and consequently a good source to represent JavaScript global popularity and use.

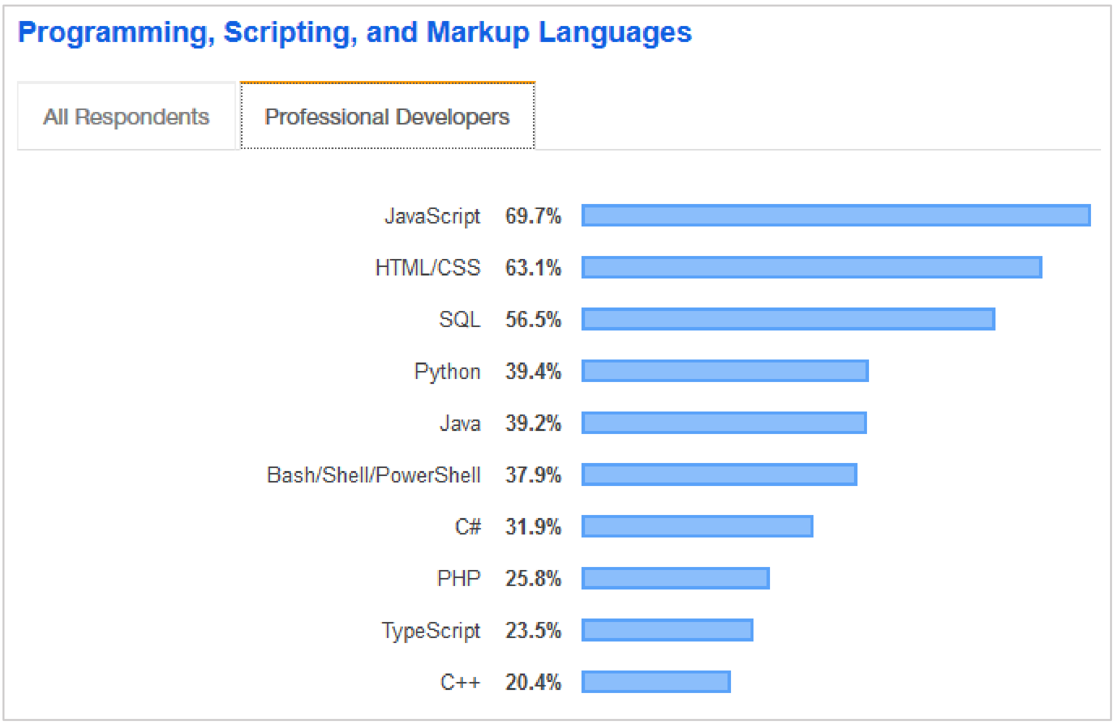


Figure 2. 2019's Stack Overflow Developer Survey - Most popular languages, for professional developers (72525 responses).

Another good source for verifying JavaScript's popularity is GitHub's recent *State of the Octoverse* and more specifically their top languages section shown on Figure 3. GitHub is widely used by developers around the globe to share and manage code, for professional purposes and personal projects. The 'Top languages' ranking on the *State of the Octoverse (Figure 3)*, shows once again the clear dominance of JavaScript and its use over the years, while other languages have been experiencing changes in their popularity. From this ranking it is also worth noting TypeScript's growth, which is tightly related to JavaScript, since TypeScript is a *'typed superset of JavaScript that compiles to plain JavaScript'* (Typescript 2020), which only supports JavaScript dominance.

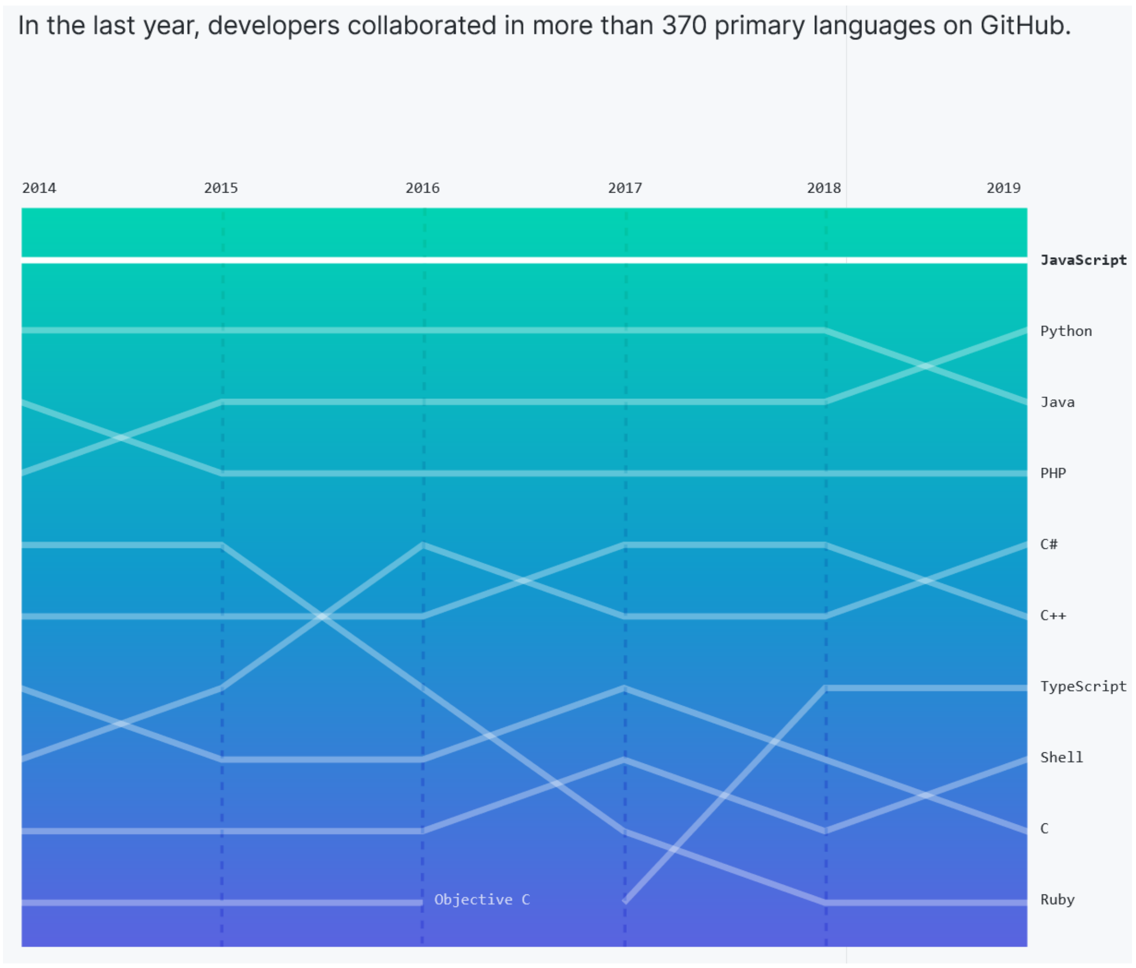


Figure 3. Top languages in GitHub by repository contributions from 2019's The State of the Octoverse.

2.4.1 Why JavaScript?

With the massification of the internet, JavaScript became an easy choice to teach programming language to newcomers. Unlike other programming languages, if you want to run JavaScript code, you only need a computer, an internet connection and a web browser. Other programming languages require a program that compiles the human code, into computer code, which may require tedious configuration and extra knowledge. In contrast JavaScript has the advantage in its ability to run code on a web browser's console through its internal engines. This feature rids users of configuration hurdles and allows them to easily start learning the basics of programming.

While JavaScript was only able to be used on the Frontend programming, the arrival of JavaScript's Node on 2009 expanded JavaScript's use to the Backend programming. This way knowing JavaScript was enough to run code on the server and on the client's side of the web, instead of switching from one language to the other in order to work on frontend or backend (Brown 2016). This allowed JavaScript's popularity and support to increase and along with that bring new libraries and frameworks, which eventually became very popular. It is perhaps for that reason that some important programming courses in Haaga-Helia have been focused on JavaScript technologies, such as in Frontend, Mobile Programming or Software Project courses.

2.4.2 JavaScript and ECMA standard versions

The European Computer Manufacturers Association (ECMA) was officially former in 1961, in order to respond to the increasing need for standardization that the continuous growth of computer use created. This brought together the mayor computer related companies at the time, such as IBM, in order to cooperate and define the best possible standardization and allow compatibility throughout different countries and manufacturers. (ECMA International 2020)

In a strategic move to keep competitors at bay, Netscape decided to standardize JavaScript via ECMA, as seen in Table 1. This came into fruition through the ECMA-262 standard, with the first these standardized versions commencing in 1996, allowing the varied browser vendors to implement JavaScript and thus letting the language be a viable option to the browser market. Due to Oracle (previously Sun) owning rights to term JavaScript, the official name used for the language had to be officially referred as ECMAScript, but regardless of this situation, the language keeps being known to people as JavaScript and perhaps with many of them unaware of ECMAScript being its official name, rather than just a version release name. (Rauschmayer 2014)

Table 1. ECMAScript versions so far (ECMAScript 2019 Language Specification, 2020).

Version	Official Name	Other Names	Release Year	Extra information
1st	ECMAScript 1st ed.		1997	Started on June 1996 Adopted on June 1997
2nd	ECMAScript 2nd ed.		1998	Editorial changes to keep it aligned with ISO/IEC 16262
3rd	ECMAScript 3rd ed.		1999	Adds regular expressions and try/catch exception handling Published as ISO/IEC 16262:2002 in June 2002
4th	ECMAScript 4th ed.	ES4	Unreleased	The significant work on this edition was incomplete and unpublished Useful parts of it were further developed and implemented on the 6th ed.

5th	ECMAScript 5th ed.	ES5	2009	Includes JSON support, strict mode, additional array manipulation functions, among others.
5.1	ECMAScript 5.1		2011	Minor corrections
6th	ECMAScript 6th ed.	ECMAScript 2015 ES6	2015	Its development started in 2009 Mayor enhancements for the languages Adds let and const
7th	ECMAScript 7th ed.	ECMAScript 2016	2016	Further development entirely on GitHub Adds new exponentiation operator (**) and Array.prototype.includes
8th	ECMAScript 8th ed.	ECMAScript 2017	2017	Small language and library enhancements, bug fixes and editorial updates Adds Async functions, among others
9th	ECMAScript 9th ed.	ECMAScript 2018	2018	Support for Asynchronous iteration Four new regular expression features Rest parameter and spread operator support for object properties
10th	ECMAScript 10th ed.	ECMAScript 2019	2019	Few new built-in functions Minor updates to syntax and semantics

2.4.3 Browser compatibility

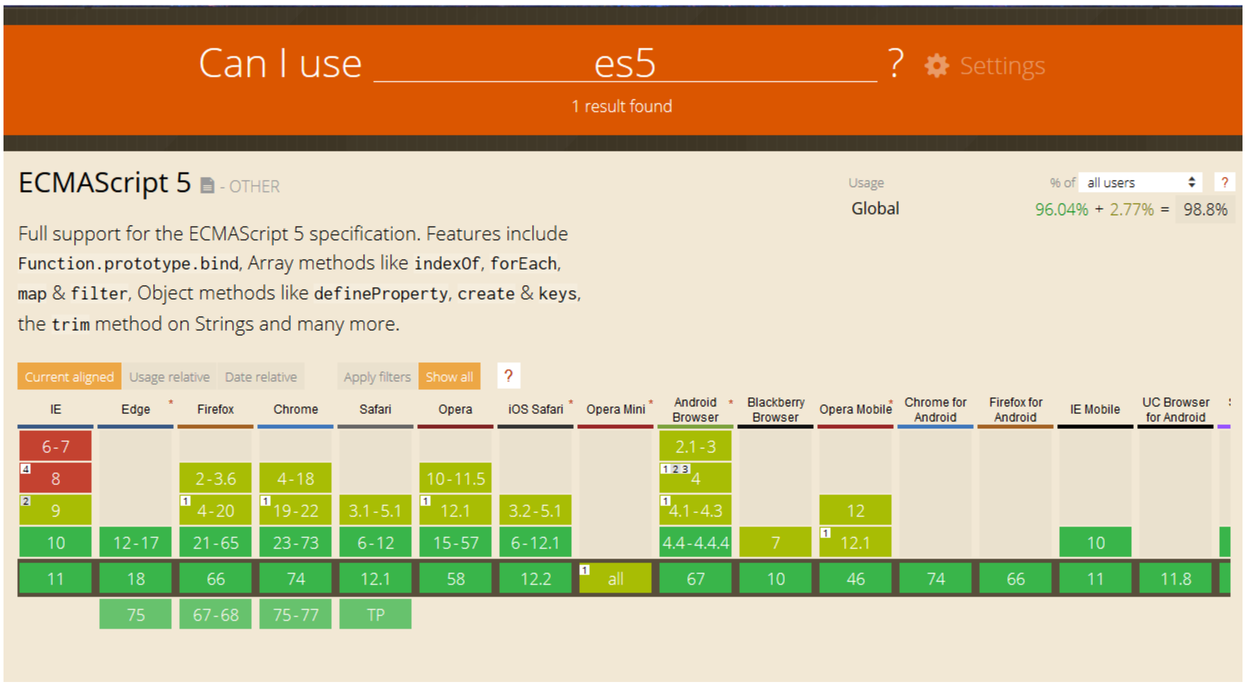
Netscape was the first company to recognise the great potential of the internet and take advantage of it by being the first to provide a great quality browser ‘Navigator’, which included more useful features and better user experience. This allowed Netscape to reach an 87% share of the browser market by 1996 and as Netscape’s Navigator took the lead by a great margin, in 1995 Microsoft began its strategy to catch “The Internet Tidal Wave”. (Spinello 2005)

Microsoft took advantage and even abused their Windows operating system’s dominance and relations to other services in such a way that in time gave the browser Internet Explorer (IE) an 80% of the browser market share by December 1999. Microsoft’s questionable tactics were however put to a halt when facing an antitrust lawsuit by the U.S. government. This lawsuit made it more difficult for Microsoft to block its competitors advancing in the middleware market of web browsers. (Spinello 2005)

The fact that there are different web browser options, means that each browser developing team must find their own way to enable new features of JavaScript to run on their browser's engine. Browser engines are the ones responsible of turning human language into computer code through programming language. It is for that reason that support for a newer ECMAScript's version may take time to implement on each browser.

When developing or maintaining globally popular internet services, it is imperative to keep browser compatibility in mind before refactoring code or implementing JavaScript's newest feature. This is because a considerable number of customers of such services could be using outdated browser versions that don't support some newer features. Failure to ensure the support of new features could threaten revenue and customer loyalty, instead of taking full advantage of such features.

A good practice for developers is checking if a desired JavaScript feature has browser support. Services such as *Can I Use,* give information about the current feature support (support tables) for varied web technologies across different browser options (latest versions). Figures 4 and 5, depict ES5 and ES6 support, as of May 2019, where green, lime and red colours boxes indicate support, partial support and no support respectively.



Can I use es5 ? Settings

1 result found

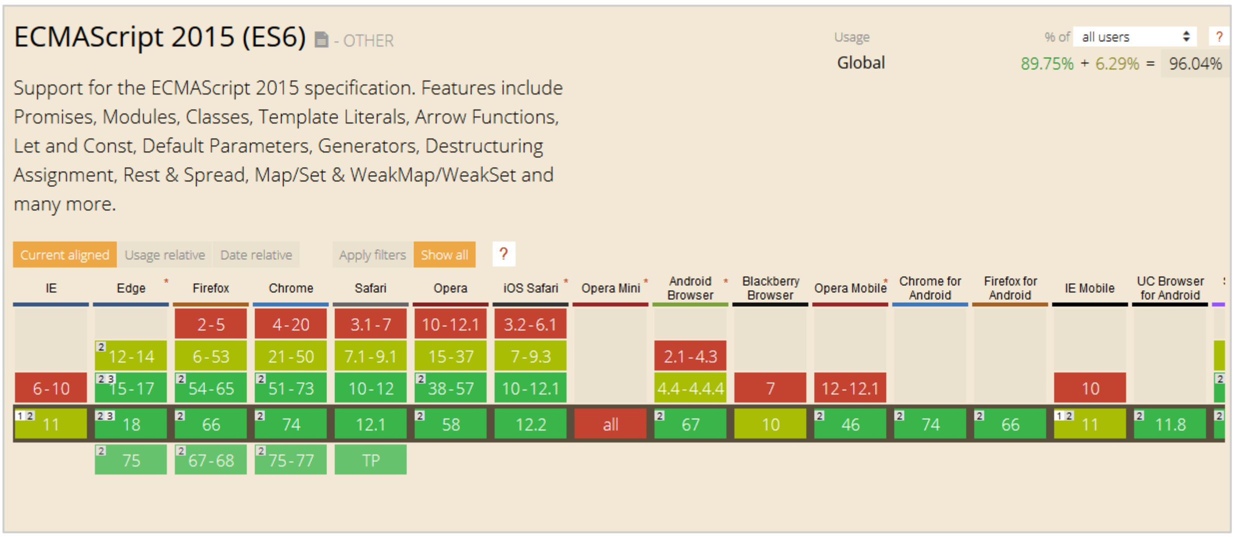
ECMAScript 5 - OTHER

Usage Global — % of all users: 96.04% + 2.77% = 98.8%

Full support for the ECMAScript 5 specification. Features include `Function.prototype.bind`, Array methods like `indexOf`, `forEach`, `map` & `filter`, Object methods like `defineProperty`, `create` & `keys`, the `trim` method on Strings and many more.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Blackberry Browser	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android
6-7								2.1-3						
[4] 8		2-3.6	4-18		10-11.5			[1 2 3] 4						
[2] 9		[1] 4-20	[1] 19-22	3.1-5.1	[1] 12.1	3.2-5.1		[1] 4.1-4.3		12				
10	12-17	21-65	23-73	6-12	15-57	6-12.1		4.4-4.4.4	7	[1] 12.1			10	
11	18	66	74	12.1	58	12.2	[1] all	67	10	46	74	66	11	11.8
	75	67-68	75-77	TP										

Figure 4. ECMAScript 5 (ES5) browser compatibility status on May 2019 by Can I Use.


ECMAScript 2015 (ES6) - OTHER

Support for the ECMAScript 2015 specification. Features include Promises, Modules, Classes, Template Literals, Arrow Functions, Let and Const, Default Parameters, Generators, Destructuring Assignment, Rest & Spread, Map/Set & WeakMap/WeakSet and many more.

Usage: % of all users

Global: 89.75% + 6.29% = 96.04%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android
		2-5	4-20	3.1-7	10-12.1	3.2-6.1								
	[2] 12-14	6-53	21-50	7.1-9.1	15-37	7-9.3		2.1-4.3						
6-10	[2 3] 15-17	[2] 54-65	[2] 51-73	10-12	[2] 38-57	10-12.1		4.4-4.4.4	7	12-12.1			10	
[1 2] 11	[2 3] 18	[2] 66	[2] 74	12.1	[2] 58	12.2	all	[2] 67	10	[2] 46	[2] 74	[2] 66	[1 2] 11	[2] 11.8
	[2] 75	[2] 67-68	[2] 75-77	TP										

Figure 5. ECMAScript 2015 (ES6) browser compatibility status on May 2019 by Can I Use.

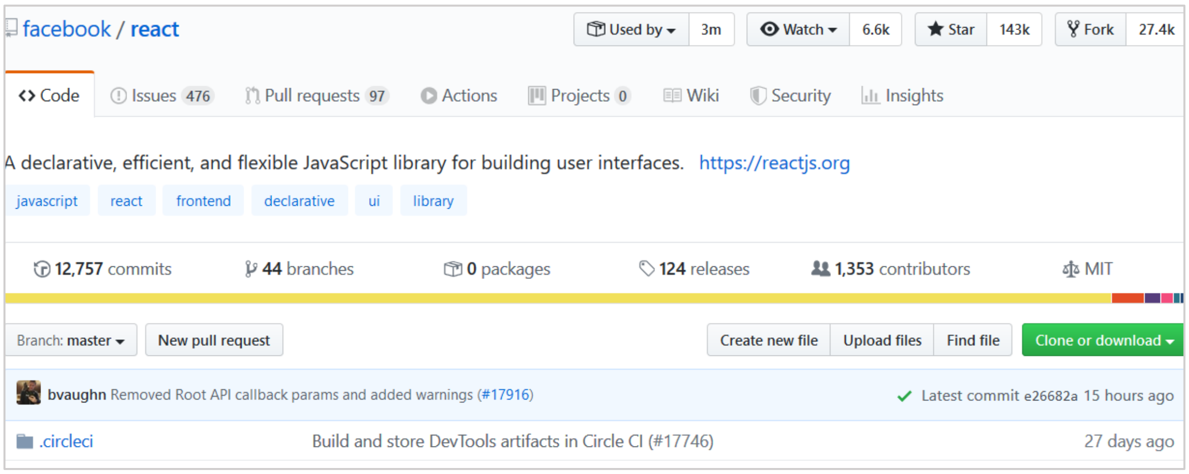
2.5 Current popular JavaScript frameworks & libraries

As there are a plethora of ways to write code that execute the same action, a mix of knowledge, creativity and luck contribute to writing code in the most efficient way. This feat is not always so easy or evident to achieve, even for people possessing strong coding skills. Time is also a valuable resource that needs to be optimized as much as possible. For that reason, libraries and frameworks emerged and allowed developers to focus on the task at hand, rather than reinventing the wheel.

With the help of libraries and frameworks, developers can use other developer's code, in order to satisfy their needs. The most popular libraries and frameworks specially are constantly reviewed, improved and maintained by big organizations like Google or Facebook or groups of developers. This way their security and efficiency are mostly stable or improved and therefore, so could be the code of those who use them.

2.5.1 React

Although React is many times referred to as a framework, it is instead a very useful and popular JavaScript library. As React's official page states in its title React is in simple terms "*A JavaScript library for building user interfaces"* (React 2020). This library has been developed and maintained by Facebook since its release in 2013 and was created in response to solving the challenges of handling larger and data-driven applications (Banks & Porcello 2017). React's usefulness expanded beyond the confines of Facebook having become an extremely popular option for many developers, as seen in Figure 6, with over 3 million repositories using React and close to 150 thousand GitHub users giving a star to React's official GitHub repository.


facebook / react

Used by 3m | Watch 6.6k | Star 143k | Fork 27.4k

Code | Issues 476 | Pull requests 97 | Actions | Projects 0 | Wiki | Security | Insights

A declarative, efficient, and flexible JavaScript library for building user interfaces. https://reactjs.org

javascript react frontend declarative ui library

12,757 commits | 44 branches | 0 packages | 124 releases | 1,353 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

bvaughn Removed Root API callback params and added warnings (#17916) ✓ Latest commit e26682a 15 hours ago

.circleci | Build and store DevTools artifacts in Circle CI (#17746) | 27 days ago

Figure 6. React GitHub page (January 2020).

React's way of organizing applications by components, has made it easier to keep a DRY (Don't Repeat Yourself) way of coding and allowed developers to create faster and more efficient user interfaces by isolating and reusing code and manipulating only necessary parts of the DOM to avoid full re-rendering of the whole UI. (Rascia 2018)

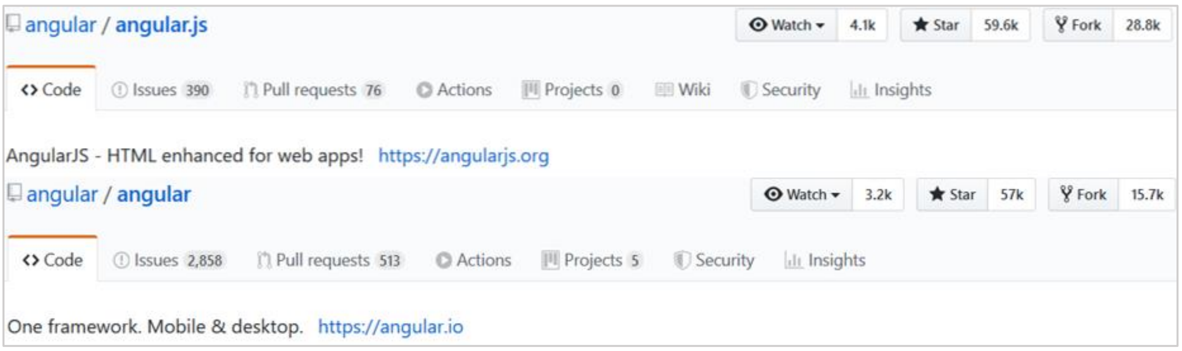
Another good aspect of React, is being able to experience its benefits on other platforms, as Facebook took advantage of React's success and implemented it to mobile applications through React Native. Although coding in React Native is not exactly the same as coding in React, it has many similarities and familiar concepts that are easy to grasp when knowing React, providing a convenient advantage for targeting Android and iOS platforms. With the use of one language (JavaScript) developers have the opportunity to cover web and mobile development instead of relying on two different ones (such as Java and Swift respectively). This way knowledge of React makes it easier to expand and compete in mobile platforms. (Boduch 2017)

2.5.2 Angular & Vue.js

Together with React two of the other most popular JavaScript frameworks are Angular and Vue.js. Even though React is a library and not a framework per se, it is still constantly put alongside frameworks. Angular has been developed and maintained by Google to this day, while Vue has been developed by one of Google's former employees, Evan You (尤雨溪) while he was still working at Google. (Honeypot 2020)

Angular was released at the end of 2010 as AngularJS (Angular 2020) and at the time, it brought unfamiliar concepts such as data binding, separation of concerns and dependency injection, which eventually let it become a trend setter across emerging frameworks. Eventually AngularJS renewed itself to the core in order to adapt to a new era of web development, but this new change would turn Angular into a completely new version of itself (initially called Angular 2.0, but currently referred as Angular). Although it was a necessary change, it was a drastic one and not backwards compatible, which consequently harmed Angular's popularity and frustrated an important part of its followers.

Although Angular is not as popular as React on GitHub repositories, Figure 7 shows that Angular is still very popular with 57 thousand starts given by GitHub's users and 3.2 thousand followers, while Angular's previous mayor version *AngularJS* has 900 hundred more followers. This situation could perhaps be attributed to Angular's drastic change and the attractiveness of newer emerging JavaScript frameworks.

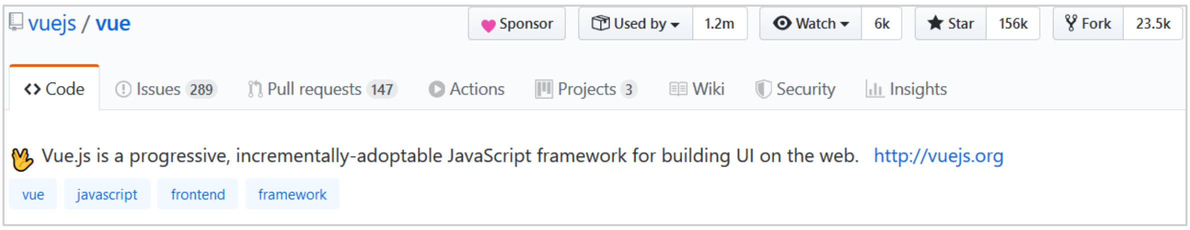


AngularJS - HTML enhanced for web apps! https://angularjs.org

One framework. Mobile & desktop. https://angular.io

Figure 7. AngularJS and Angular GitHub pages.

Vue on the other hand was released in 2014 with the intention of combining the best features of React and Angular and has since continued to gain popularity over the years, being backed by a strong community (Macrae 2018). As seen in Figure 8, Vue has been used in 1.2 million repositories in GitHub and starred by 156 thousand users by January 2020 becoming even more popular than React and Angular on GitHub but used only by almost half of repositories than React on GitHub.



Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web. http://vuejs.org

Figure 8. Vue.js GitHub page.

2.5.3 jQuery

For people that are just starting to learn about programming (particularly in JavaScript), the mention of jQuery may sound alien or simply as something they heard somewhere but without much knowledge about it. At times when Internet Explorer had dominion over the browser market, browser compatibility was at its worse and DOM manipulation was difficult to achieve, but then jQuery came to the rescue. (Henry 2017)

jQuery at the time (circa 2010), provided the benefits of DOM manipulation, event handling and HTTP requests, but most importantly it was consistent. At the time jQuery was the best way of ensuring the proper working of code across different browsers, but as it was so well developed to solve the issues of its own era, jQuery was incapable of adjusting to a new era of web development. Figure 9 shows the clear decline of jQuery as it was around 2017 when JSX and React began to take off towards leadership and jQuery to its demise. (Henry 2017)

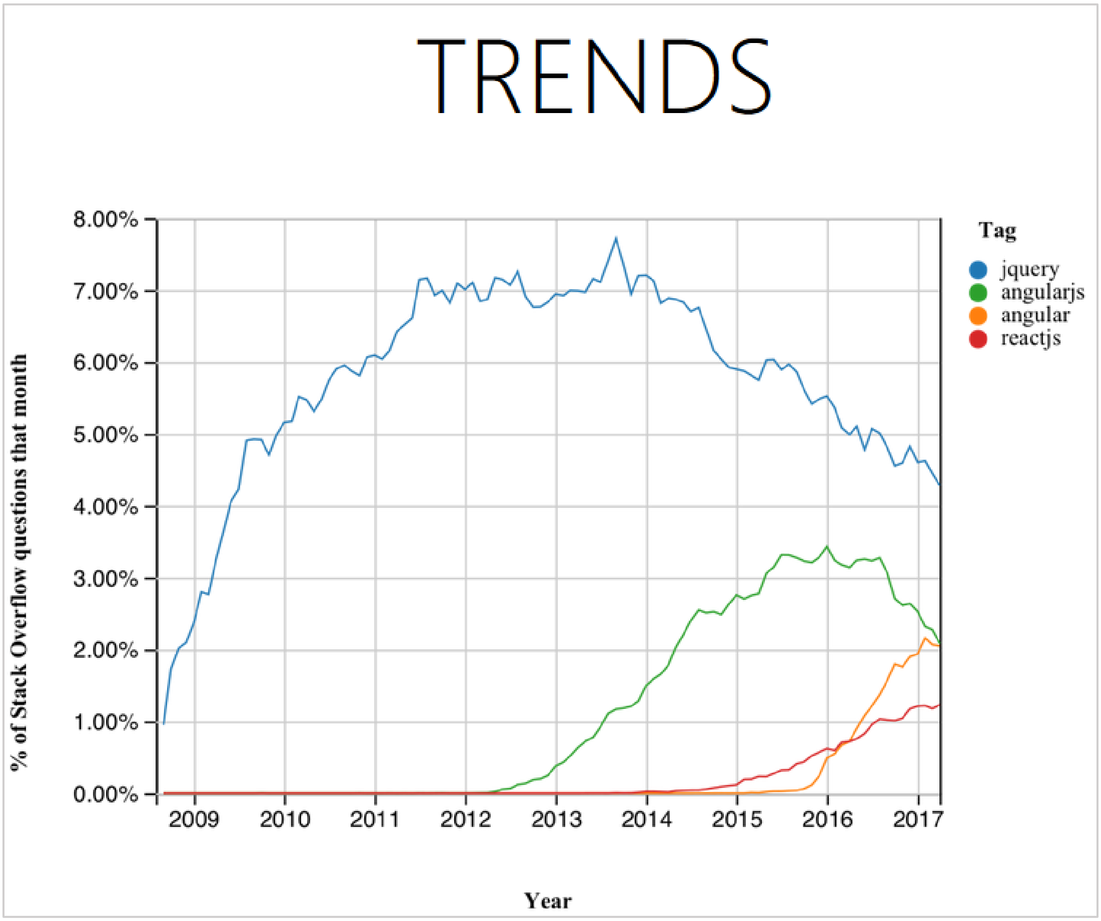


Figure 9. Stack Overflow question trends by language.

2.6 Memes and what they can reveal to us

Nowadays memes are associated with ridiculous humour driven edited pictures or text, whose subjects vary from widespread political issues to subcultural themes, such as particular tv shows, music or even related to your closest group of friends. Memes can range from clever subtle humour, to completely ridiculous or politically incorrect, they can be carefully elaborated and edited on a professional level or just be of the poorest quality where the lack of effort is evident. But they all have something in common, as they are a representation of cultural phenomena that people experience and have in common with others.

To the surprise of many, the term *meme* has its roots in nothing related to the internet. In fact, it was biologist Richard Dawkins who in 1976 used the term in his book *The Selfish Gene,* to represent people's transmission of culture by means of copying or imitation. Examples of such description can be found for example in the way we use phrases we hear from others, fashion trends or even religious beliefs. Dawkins used the word *meme* as a shorter version of the Greek term *mimema*, which means "something which is imitated", and that way connecting it to the word *gene*. (Shifman 2012)

As a way to transmit and share cultural phenomena, memes can be easily taken and changed according to our own views and interpretations, just as a single phrase that goes from ear to ear and coming back as a completely different phrase to the person that started it. The difference with the popular current use of memes is that all these versions of cultural phenomena can stay on the internet and resurface back to people's attention over time.

Limor Shifman states "While memes are seemingly trivial and mundane artifacts, they actually reflect deep social and cultural structures." (Shifman 2012). Such depth can be observed in other subcultures, such as the programming culture. Computer science related memes can serve as a window to a programmers' views and experiences regarding the lifestyle, challenges and misconceptions, experienced by the people involved in this line of work. The fact that these memes can circulate the internet are proof that people are able to relate to them in some degree, otherwise they simply would not spread around.

3 Research Method

In order to answer the research questions for this thesis, nonexperimental methods of research were carried out instead of performing experiments, since they would require a lot more resources and time. Therefore, the measuring of this study was carried out without altering the conditions in which students learned at Haaga-Helia. (Hoy 2012)

The research questions were based on the students' experiences throughout their BITe programming studies. In order to best answer these questions a mix of qualitative and quantitative methods were used. The quantitative aspect of the research served to test the validity of my research questions against data provided by students, while the qualitative aspects of the research helped the understanding of the student's thoughts and feelings towards JavaScript, shedding light into new ideas to improve their learning. (Hoy 2012)

In order to answer the first two research questions *"Are BITe students discouraged in any way to continue learning and developing professionally in JavaScript related fields?"* and *"What are BITe students' feelings regarding JavaScript, its frameworks and libraries?"*, data was gathered quantitatively through an electronic-survey, while in order to answer the last question *"What possible improvements there could be on the JavaScript teaching methods in Haaga-Helia for BITe students?"* data was gathered mostly qualitatively through semi-structured interviews. (Saunders, Lewis & Thornhill 2009)

The data that was collected will help primarily to determine how Haaga-Helia's BITe students have experienced their learning path of JavaScript and its libraries and frameworks. Additionally, the analysis of this data showed if the students' experiences and motivation to learn JavaScript changed as they progressed in their studies. The results can give the opportunity to pinpoint areas of improvement in the way JavaScript is taught in Haaga-Helia BITe degree programme.

3.1 Electronic surveys

To gather the data for this research, an electronic survey was sent to BITe students. The survey consisted of questions regarding the students' JavaScript related studies at Haaga-Helia and their feelings and attitudes towards the use of these technologies.

Instead of survey methods like post mail or telephone interview methods, I chose to conduct an electronic survey through Webropol software, for which Haaga-Helia has a licence available for students. Online surveys have the advantage of being low cost, fast and efficient, as well as having effective contingency questions and direct data entry, which fits better for this research and its target group. Even though electronic surveys have some potential disadvantages, as for example possibility of coverage bias, reliance on software and overload of digital surveys, the benefits of conducting such survey outnumbers its drawbacks. (Sue & Ritter 2012)

To reach all the relevant groups for this study and mitigate chances of coverage bias, I used the help of our BITe programme academic advisor Riitta Blomster, as she has access to the WhatsApp chats of different student groups. Riitta's help allowed me to share the survey link along with a short descriptive message and send a reminder about it after a few days.

An additional way of reaching all relevant groups, was giving all the respondents the chance to participate in a lottery and win movie tickets. The respondents could participate in the lottery if they provided some basic contact information at the end of the survey, which was managed separately from their answers in order to keep their anonymity. The winners were then selected at random using JavaScript code on a browser's console, which can be found on Appendix 2. This lottery hopefully helped to receive more answers and motivate the respondents to not abandon the survey halfway through.

The survey was divided in three sections, *basic background information*, the students' learning experiences of *JavaScript* and *memes* and their representing of the students' feelings about JavaScript (survey questions available on Appendix 1). The first section aimed to gather some general information such as age range, gender, interest in BITe programming courses, etc. Secondly, the *JavaScript* section aimed to pinpoint personal impressions and feelings towards JavaScript and the way it is taught. Finally, the *meme* section present coding-related memes and the participants were asked how much they agree with the situations the memes are showcasing and if they feel represented by them.

Figure 10. Meme sample.

Even when the survey was done anonymously, some answers might have been given without a 100% honesty, since consciously or not, some people may try to present a better image of themselves and avoid showing ignorance. In general people use humour to deal with stress and difficulties, which is what the memes shown in the survey represent, as seen in Figure 10.

The answers on the different sections were analysed together, which enabled to point out any inconsistencies throughout all the responses.

3.2 Semi-structured interviews

Since the electronic surveys are targeted to all BITe students, the survey questions had to be planned in a way that satisfies most respondents independent of their programming or JavaScript experience and skills. For that reason, a couple of semi-structured interviews were carried out, where I had the chance to ask more complex questions and allow on the spot follow-up questions. (Sue & Ritter 2012)

These interviews were conducted with BITe students that were already done with or finishing their internships. The aim was to gather information from students with enough experience of working in the field, while still having a fresh experience of their studies. Data from these interviews provided additional information about the students' experiences, both positive and negative, about their studies in Haaga-Helia and the teaching methods of JavaScript and programming in general, along with possible development suggestions.

4 Results

4.1 Survey

4.1.1 Background Information section results

The total number of respondents was 54 of which 57.41% were male and 42.59%, but none considered themselves to be in other gender categories. The great majority of respondents were Russian, Vietnamese or Finnish, while the remaining 54% of respondents represent 23 other countries as seen in Figure 11.

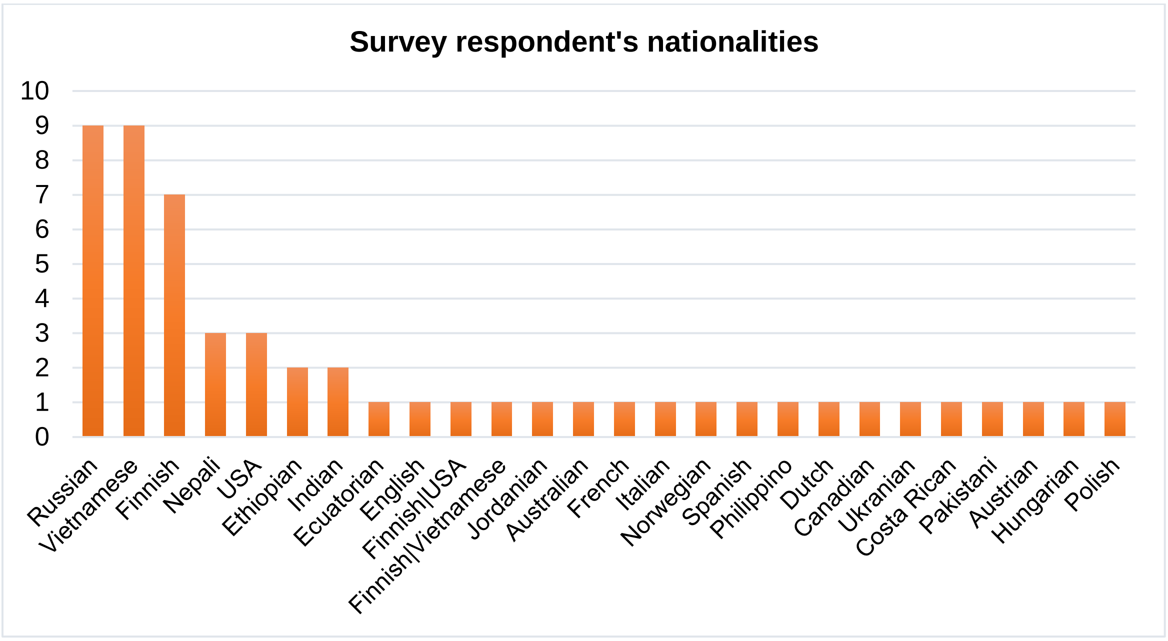


Figure 11. Survey respondents' nationalities.

The age range of the respondents, as shown in Figure 12 was between 20–42 years old, while no participant was under 20 or over 42. The most dominant age group was 20–25 years, which shows that 48% of the participants was born by the time Windows 95 came to the market. Therefore, half of the participants grew up around computers, while the other half was born when access to computers wasn't as common.

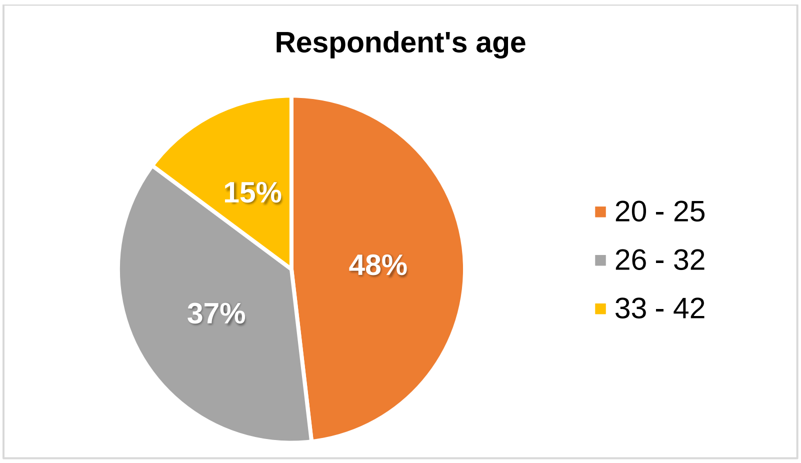


Figure 12. Respondents' age.

Figure 13 shows the participants' start of studies by year. The starting year of studies was somewhat evenly distributed between 2016–2019, while only one person started their BITe studies in 2015.

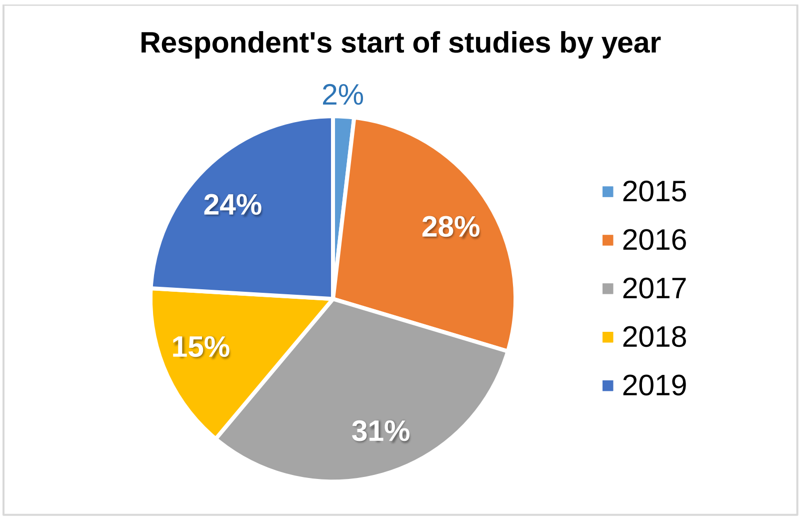


Figure 13. Respondent's start of studies by year.

Figure 14 shows whether an IT degree was a priority for the participants before starting their BITe studies. As seen from Figure 14, an IT degree was indeed a priority for more than half of the respondents. For the remaining 35% it simply was not a priority, while the other 11% was not sure. While this might not be a clear indication about their initial willingness to learn programming, one can assume that their initial interest in programming won't be as strong as for the people who had an IT career as a priority.

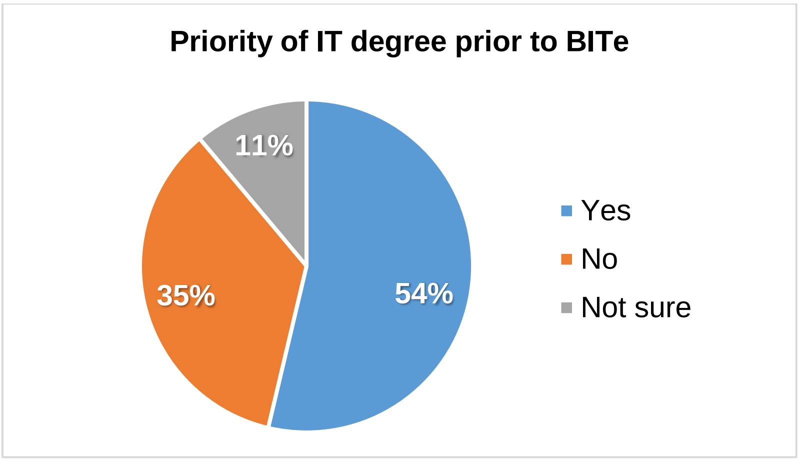


Figure 14. Priority of IT degree prior to BITe.

The participants were also asked if they pay for their studies at Haaga-Helia. Only three of them pay for their studies so this does not have a great impact on the results of this study.

Figure 15 shows the students' interest in BITe's programming courses at the start of their studies, where the great majority chose positively.

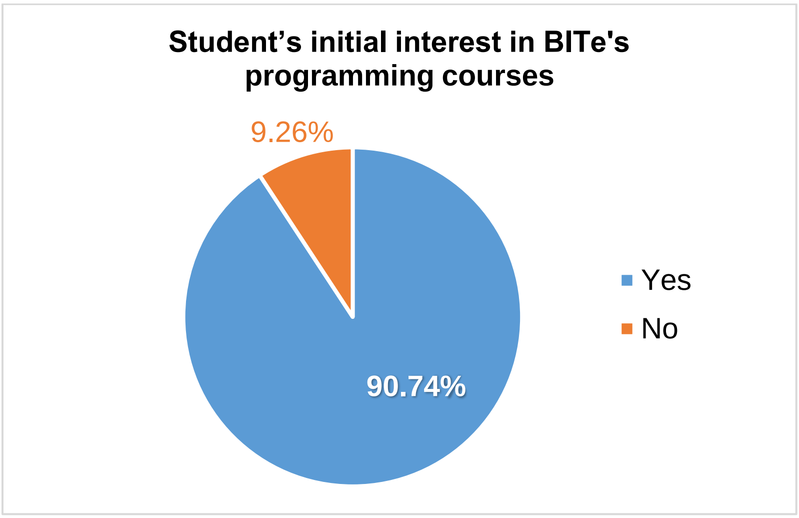


Figure 15. Student's initial interest in BITe's programming courses.

Figure 16 on the other hand shows the results of the students' interest on BITe's programming courses at the time of answering the survey, where 82.7 per cent of them had either completed those courses or was still interested in them.

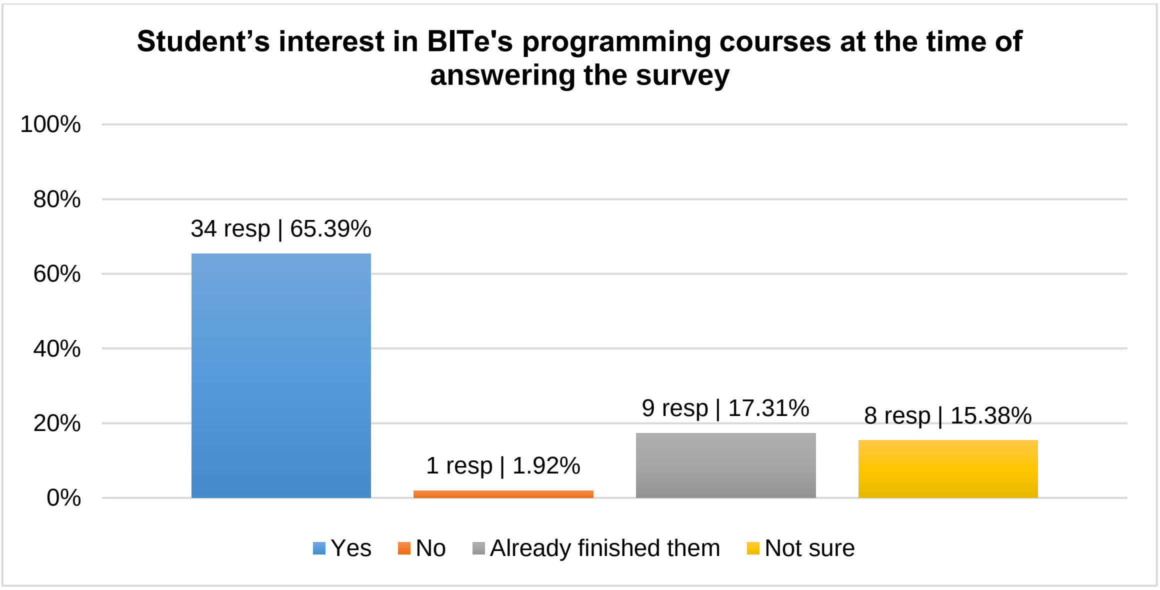


Figure 16. Student's interest in BITe's programming courses at the time of answering the survey.

4.1.2 Students' learning experiences of *JavaScript*

Figure 17 shows that a bit more than a quarter of the responders did know JavaScript before their studies at Haaga-Helia, while one third of them knew nothing about it and 37% were only familiar with its name. This shows that 70% of these students had the chance of learning JavaScript for the first time during their studies at Haaga-Helia.

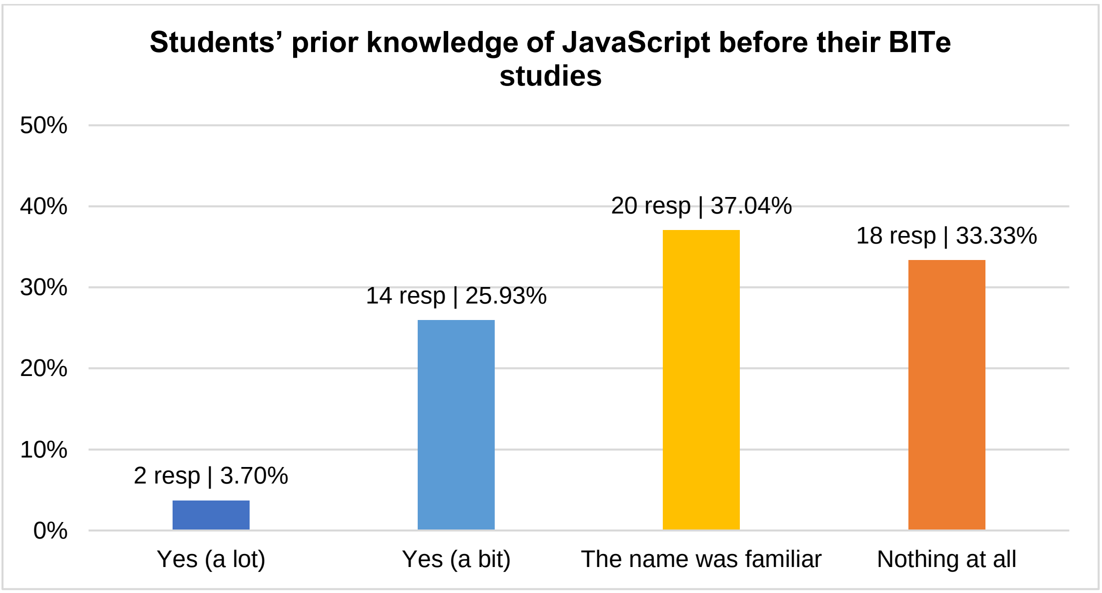


Figure 17. Students' prior knowledge of JavaScript before their BITe studies.

Figure 18 shows that the great majority of the survey respondents had already participated or were still participating in a JavaScript related course arranged by Haaga-Helia. Although 17% of them (nine respondents) had not participated on any BITe's JavaScript related courses, it did not specifically mean they didn't know JavaScript.

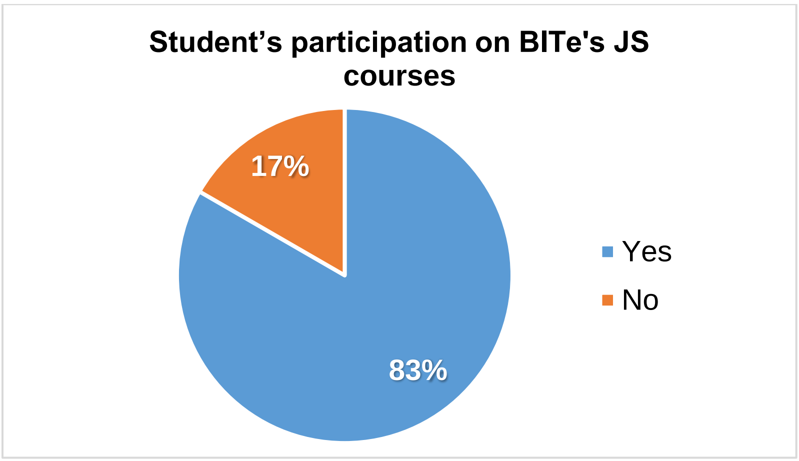


Figure 18. Student's participation on BITe's JS courses.

As seen previously on Figure 17 two respondents had previous JavaScript knowledge prior to their BITe studies, which could allow them to fast track some JavaScript related courses. Also, all new BITe students have an Orientation to Programming course, in which they learn the basics of JavaScript, but some respondents might not have been aware of that it when answering the survey.

The survey respondents were also asked about their confidence on their own JavaScript skills, as well as their skills to use JavaScript's frameworks & libraries, as depicted on Figure 19. From this information we can see that in general, the students have a lower confidence on their skills when using libraries & frameworks, but there was no case of a zero level of confidence on their skill for either category.

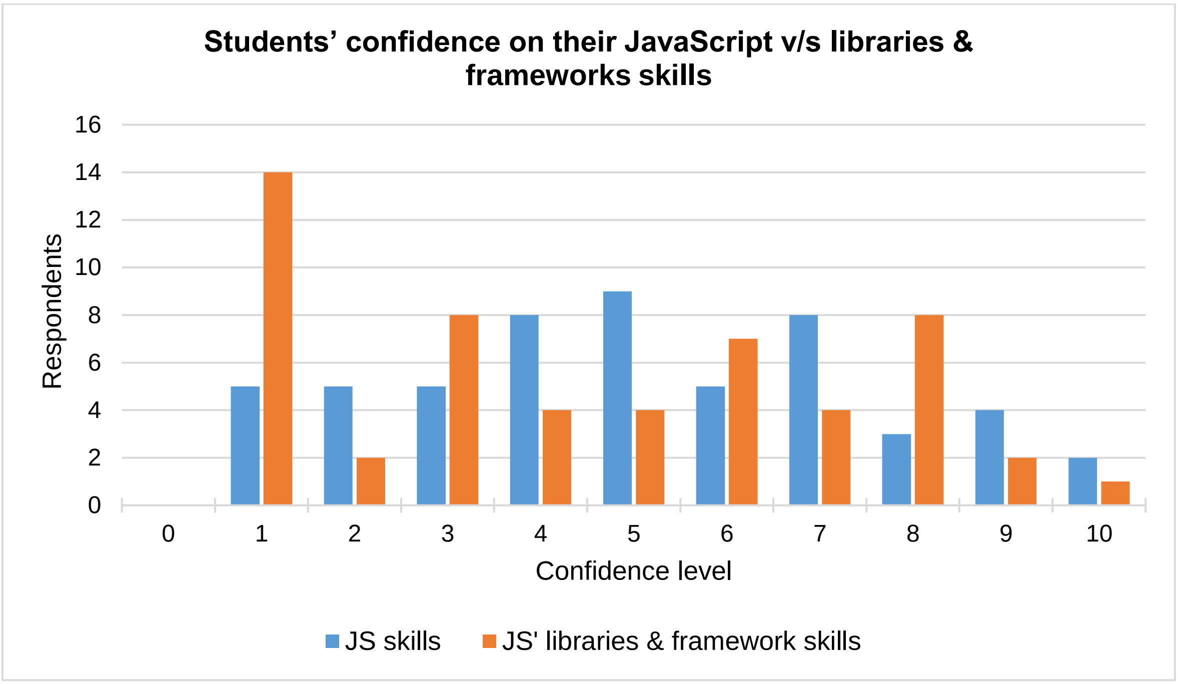


Figure 19. Students' confidence on their JavaScript v/s libraries & frameworks skills.

Table 2 shows that at the time of answering the survey, the respondents possessed an average medium level of confidence on their skills both categories, with a slightly higher level for the JavaScript category when answering the survey.

Table 2. Respondents' confidence on their skills (JavaScript v/s libraries & frameworks).

Skill confidence	Min value	Max value	Average	Median	Standard Deviation
JavaScript	1	10	5.06	5	2.5
Libraries & frameworks	1	10	4.44	4	2.79

As the JavaScript environment is in constant improvement and evolution, the surveyed students were asked about their ability to keep up with the changes of JavaScript's technologies (more specifically its libraries and frameworks). The results are shown in Figure 20 where a confidence level of 1 out of 10 got the highest concentration of answers.

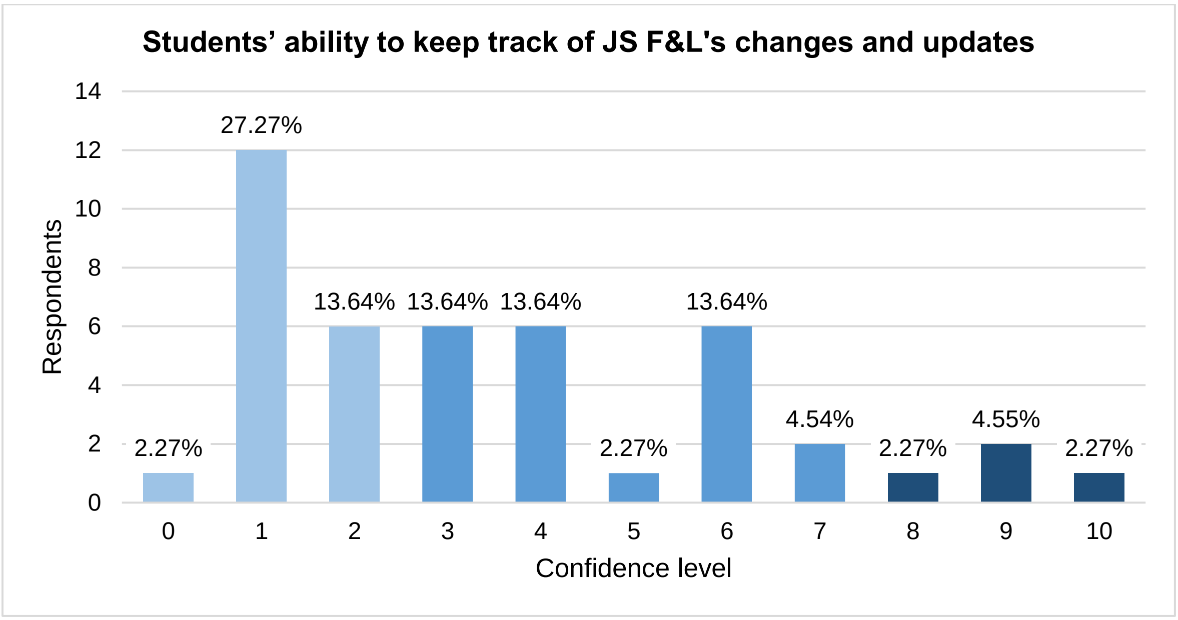


Figure 20. Students' ability to keep track of JS F&L's changes and updates.

Table 3, shows that fortunately the average of all responses had a bit higher level of confidence level of 3.57, even though some respondents had strong confidence in their ability to stay up to date with those JavaScript tools.

Table 3. Respondents ability to keep track of JS F&L's changes and updates.

Min value	Max value	Average	Median	Standard Deviation
0	10	3.57	3	2.6

The surveyed students were also asked how often they use these technologies and as shown on Figure 21, only 37% use them always, while another 37% only when necessary and 13% for both using them sometimes and not at all.

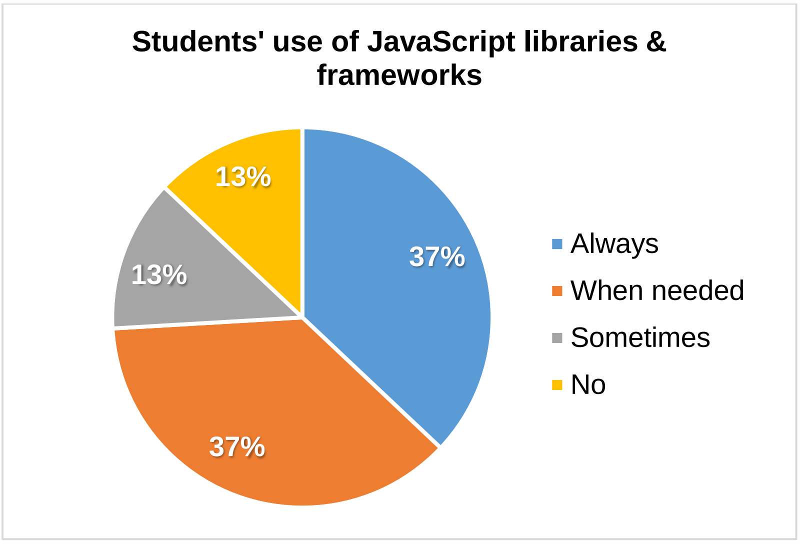


Figure 21. Students' use of JavaScript libraries & frameworks.

Figure 22 shows the results to the respondents' understanding of the popular JavaScript library React as it is used on important BITe programming courses. Figure 22 shows that the majority of their answers are on a positive nature, as even when 26% does not have a good grasp on React they do plan on becoming better at it.

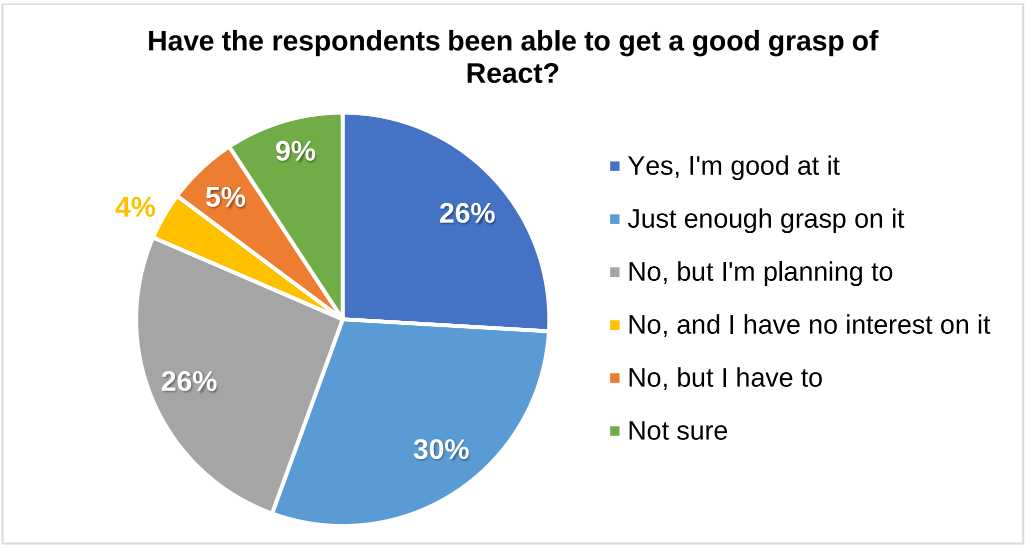


Figure 22. Have the respondents been able to get a good grasp of React?

As in BITe there are no courses were the students can learn other popular and emerging JavaScript frameworks like Angular or Vue, the students were asked about their interest in exploring such technologies. Figure 23 shows that half of them are interested, while 9% already have explored them, 24% not sure and only a few not knowing about them or not interested at all.

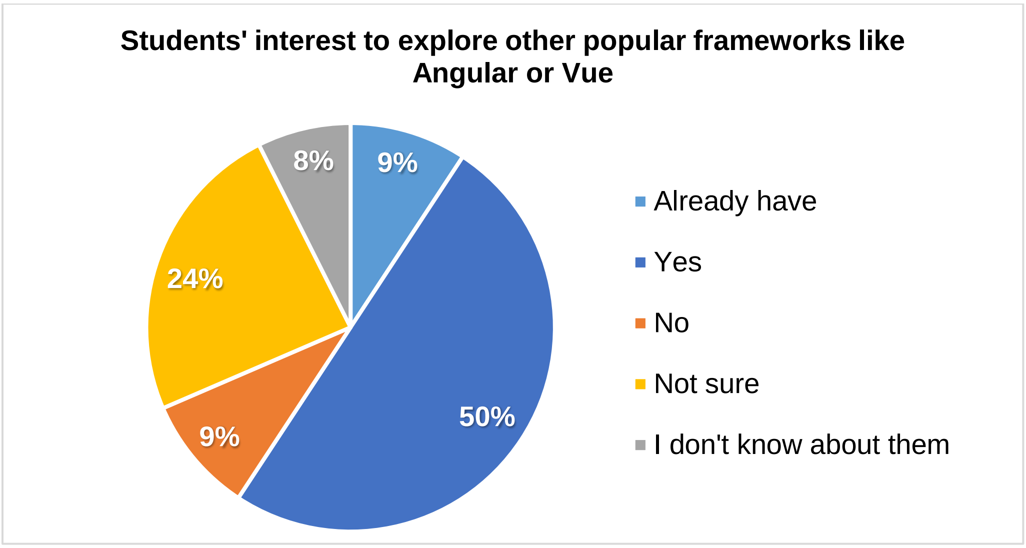


Figure 23. Students' interest to explore other popular frameworks like Angular or Vue.

Figure 24 shows the results regarding the students fear of spending too much of their free time to programming, with the great majority choosing a fear level of 1 out of 10.

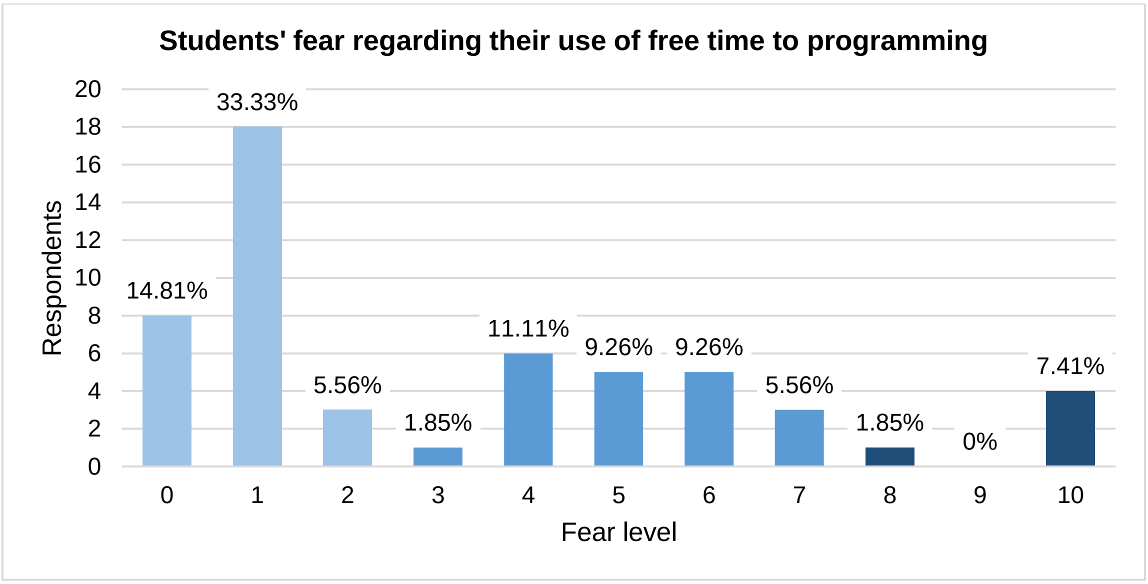


Figure 24. Students' fear regarding their use of free time to programming.

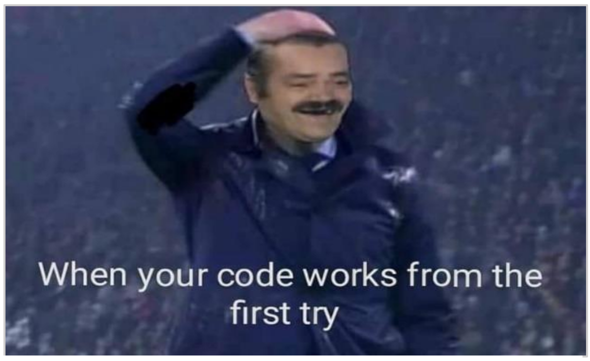
In Table 4 below the average and median, show a low level of fear on this matter, although for four people the fear was to the highest level, as seen above on Figure 24.

Table 4. Respondents' fear regarding their use of free time to programming.

Min value	Max value	Average	Median	Standard Deviation
0	10	3.24	2	3.02

4.1.3 Meme & closing sections' results

When presented with the following memes, respondents were asked, how much they agreed with the different situations they represent about programmers' experiences. These memes represented comically amusing and frustrating experiences of a developer's life and the data from these answers were further linked to the students' thoughts and feelings towards JavaScript technologies and their interest in the language.



Meme 1. Surprise if code works on first try.

Meme 1 represents how for many developers it is not expected to get your code working on first try to run it. This implies that it is part of the job to be constantly dealing with trial and error situations, which might be a frustrating way of working.

Figure 25 shows how the majority agree with the meme above with 27.78% completely agreeing with it.

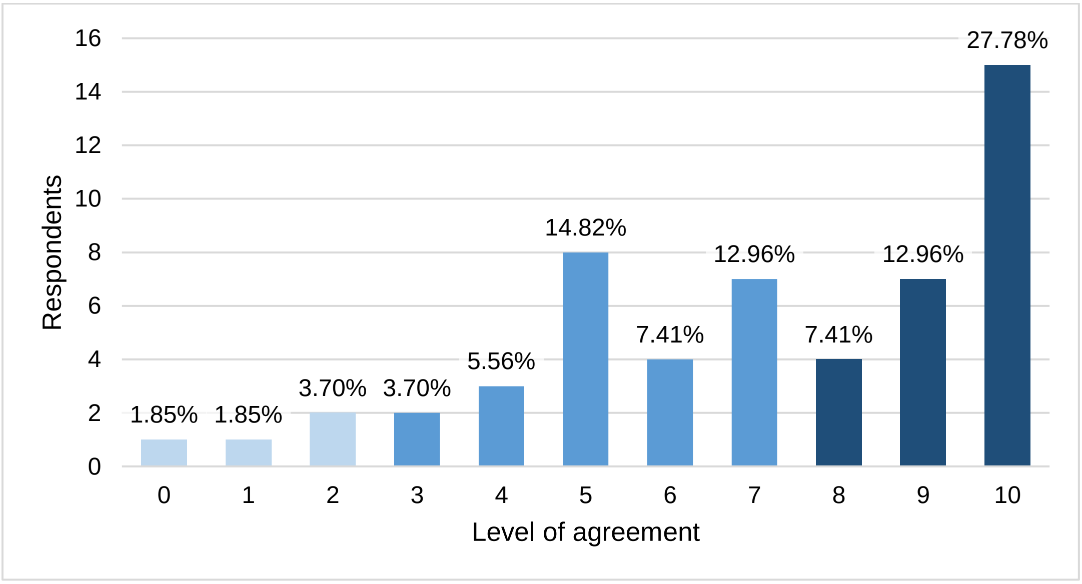
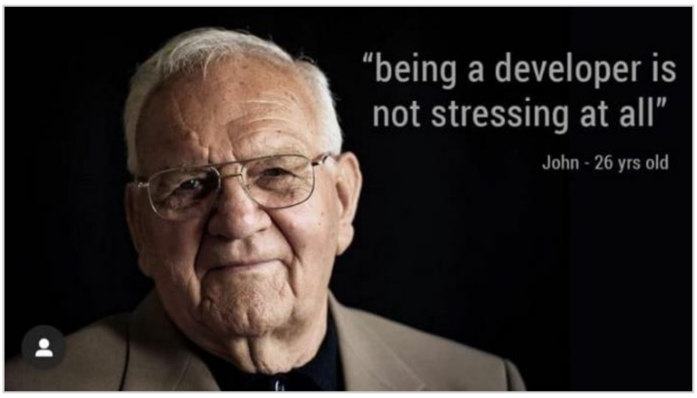


Figure 25. Answers for the 1st meme.

Table 5 below shows the average of student's responses are on a 7.06 level of agreement with the statement of being surprised when their code works on the first try.

Table 5. Extra data for the 1st meme's answers.

Min value	Max value	Average	Median	Standard Deviation
0	10	7.06	7	2.74



Meme 2 on the other hand, while being a sarcastic exaggeration, shows how for many being a developer is a stressing venture. This can lead to some students to reconsider perusing this career if the most they get from it is stress.

Meme 2. The stress of being a developer.

Unlike the first meme, Figure 26 shows that more people disagree with this second meme although these answers show a greater tendency towards being neutral regarding the idea that being a developer has the intrinsic consequence of it being a stressful undertaking.

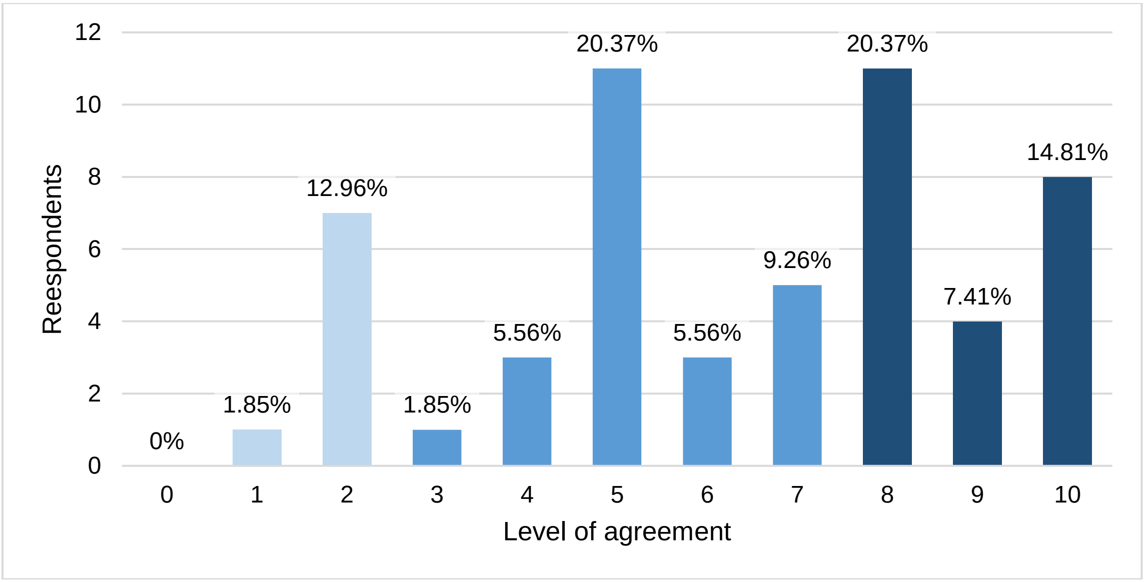
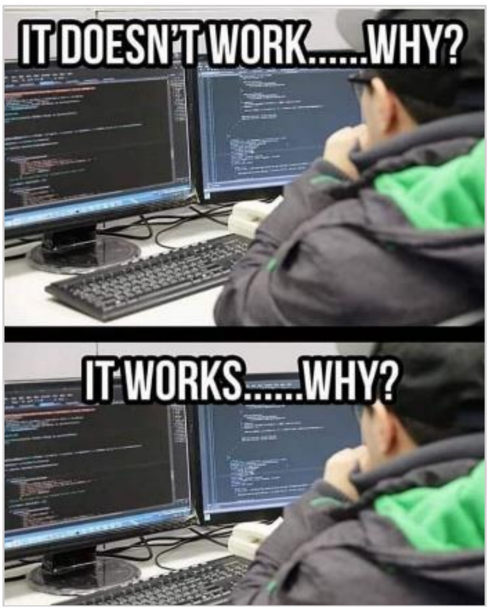


Figure 26. Answers for the 2nd meme.

Table 6 shows a bit lower average from the responses regarding the stress of being a developer when compared to the previous meme.

Table 6. Extra data for the 2nd meme's answers.

Min value	Max value	Average	Median	Standard Deviation
1	10	6.33	7	2.66



Meme 3. Confusion when code works or doesn't work.

While somewhat related to Meme 1, Meme 3 shows the recurrent *confusion* of many developers whenever their code works or not, instead of focusing on the *surprise* when it works. This meme shows how often developers code without complete certainty on how their code will behave, which can be due to not knowing the language, using code snippets found on the internet, reviewing someone else's code or simply coding in an improvised manner without much planning or structure beforehand.

When asked how much the students felt represented by this meme, as it happened with the first meme, the highest level of representation (level 10) was the most popular answer for 29.63% as seen on Figure 27, with the other 2 most popular answers being for levels 9 and 8.

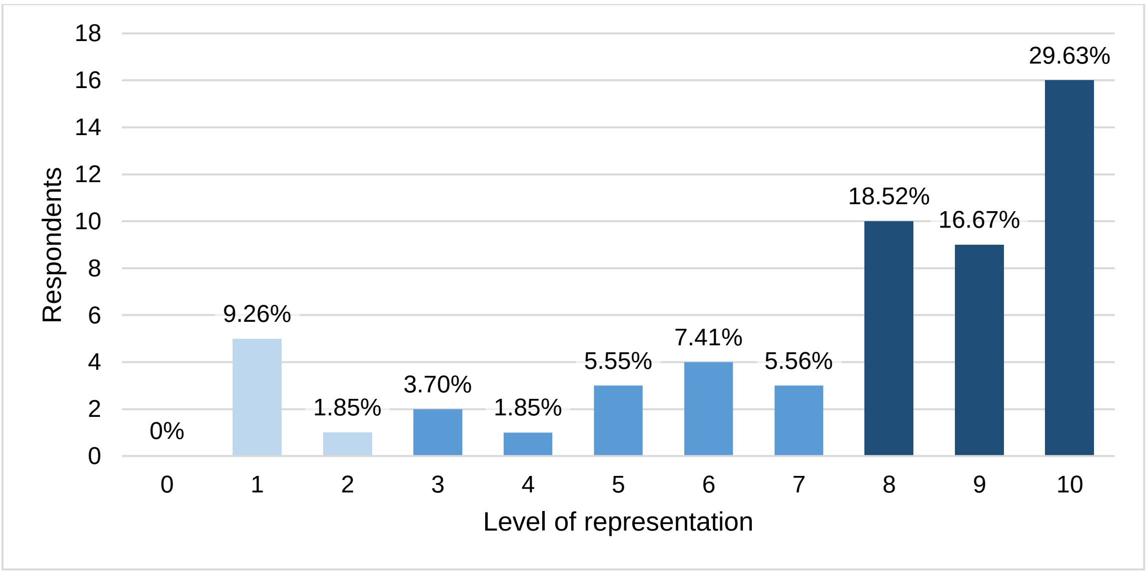
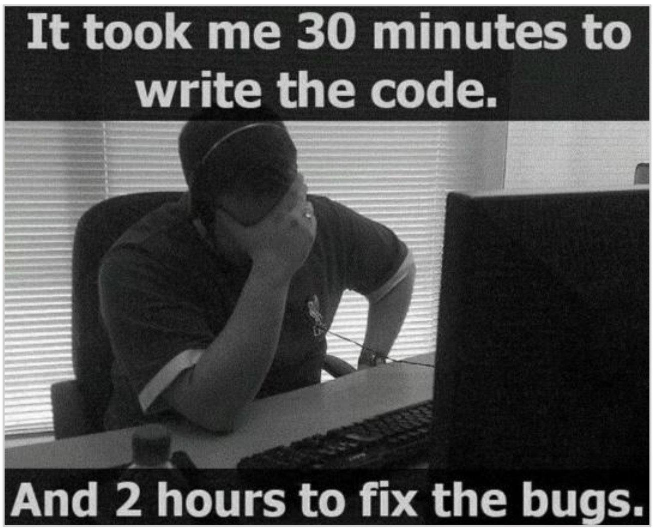


Figure 27. Answers for the 3rd meme.

Table 7 shows that most respondents felt quite represented by the situation depicted on the meme

Table 7. Extra data for 3rd meme's answers.

Min value	Max value	Average	Median	Standard Deviation
1	10	7.37	8	2.9


It took me 30 minutes to write the code.
And 2 hours to fix the bugs.

Meme 4 helps to test the idea of the great effort it can take to have code to work as intended and the notion that developers can spend more time looking at the screen trying to find the errors in the code than actually writing code.

Meme 4. Greater time and effort to fix bugs than to write code.

Once again, from Figure 28 is easy to see how little the students disagree with the meme and the results continue to show agreement level 10 as the student's most popular answer.

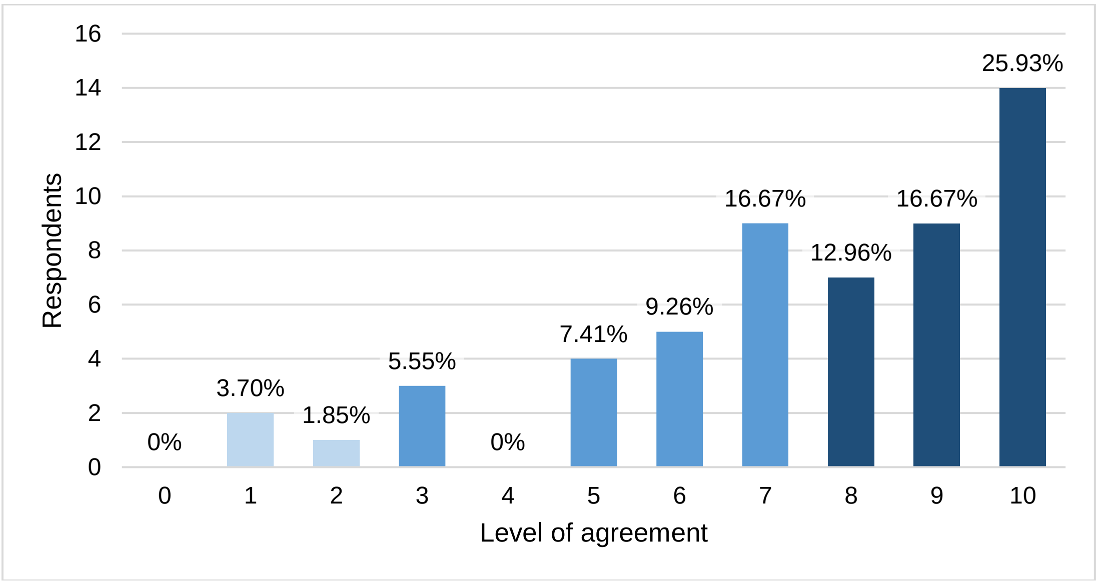


Figure 28. Answers for the 4th meme.

Table 8 shows that the average of responses for this meme highly agree with the situation depicted on it.

Table 8. Extra data for 4th meme's answers.

Min value	Max value	Average	Median	Standard Deviation
1	10	7.46	8	2.47

when you write 10 lines of code without searching on Google
It ain't much, but it's honest work

Meme 5. Googling when coding.

Meme 5 represents the idea of constantly having to search extra information in Google, in order to accomplish a desired goal when coding. This situation can be either due to being stuck with no results for too long, or simply being in a rush to deliver those results. Regardless of the reason, knowing how to Google and adjusting the findings to your needs, is an extremely valuable skill to have, as it can be difficult to achieve it efficiently and without breaking other parts of your code.

It is perhaps for that reason when the respondents were asked if they Google often while coding 40.48% chose a level 10 as Figure 29. However, this question in particular, unlike all previous ones, had only 42 answers, out of the 54 that answered the survey. The reason for such situation is unknown, but although these results not being the greatest for comparison purposes 17 picks of level 10 remains a high choice concentration.

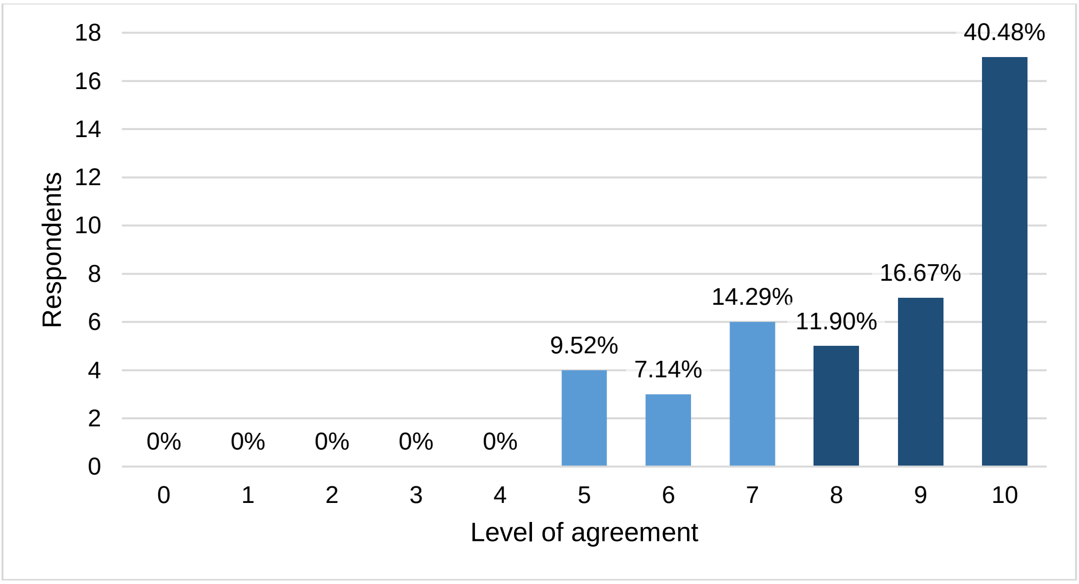
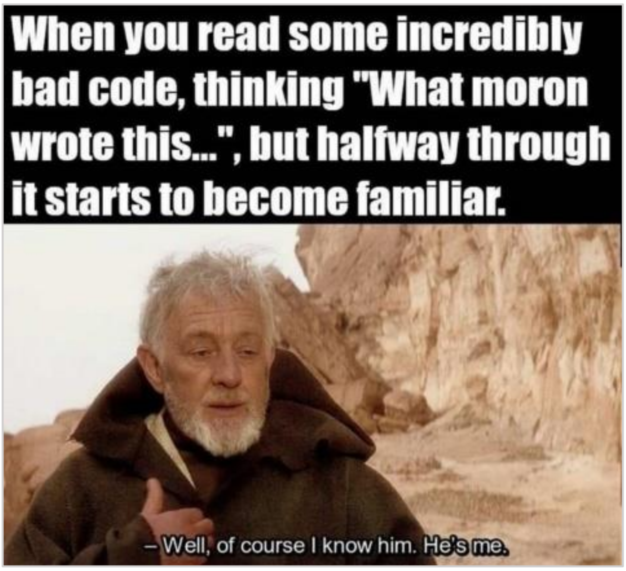


Figure 29. Answers for the 5th meme.

Table 9 shows that while not being able to represent the maximum possible of respondents the answers for this meme got the highest average and median of agreement level of all the presented memes.

Table 9. Extra data for 5th meme's answers.

Min value	Max value	Average	Median	Standard Deviation
5	10	8.4	9	1.73


When you read some incredibly bad code, thinking "What moron wrote this...", but halfway through it starts to become familiar.
– Well, of course I know him. He's me.
Meme 6. Recognizing your progress when reading your old code.

Finally, Meme 6 tested the idea of how the students' evolution as programmers can be strikingly surprising and being able to recognize your own progress, can be a great encouragement and motivation in times of professional hardship.

When students were asked if they are able to realise how much their skills have improved when reviewing old code, most of them replied positively with answers between level 5 and 10 as seen on Figure 30. Even though this question didn't receive the maximum possible of answers either, it did get more than the previous one, with 47 answers out of 54.

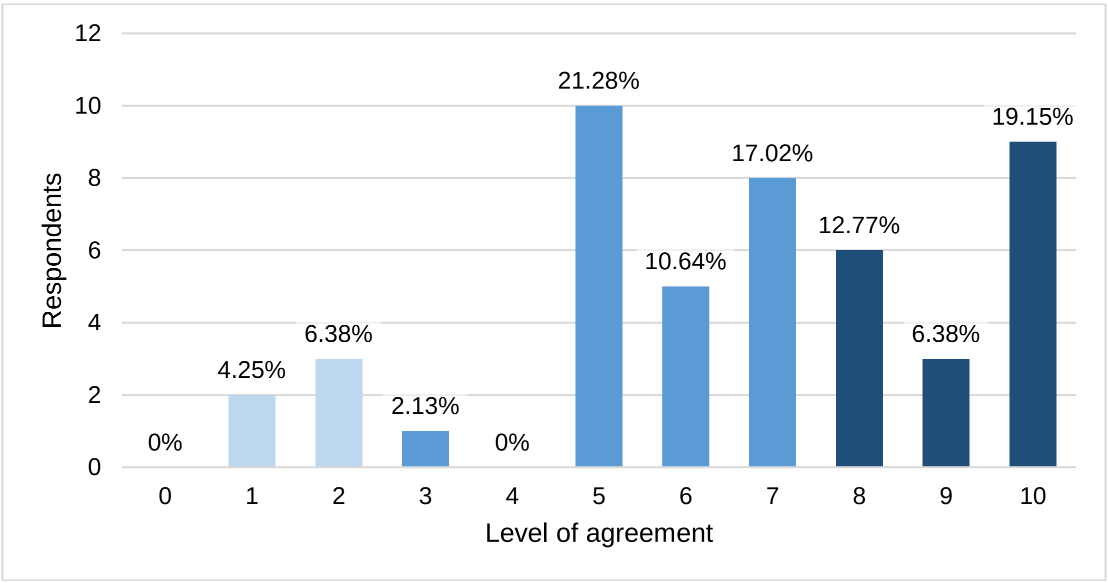


Figure 30. Answers for the 6th meme.

Table 10 shows the average of responses passes the middle point with 6.64 and a median of 7 on the level of agreement.

Table 10. Extra data for 6th meme's answers.

Min value	Max value	Average	Median	Standard Deviation
1	10	6.64	7	2.56

To finalize the survey, the students were asked three more questions in order to strengthen the results and allow the respondents to rethink their previous postures after being exposed to all the previous questions and ideas on the survey.

The first one of these questions concerned the respondents' opinions about the memes shown on the survey being able to represent the life of developers in general. The students were given five answers to choose from and they were able to pick as many as they wished. Figure 31 shows the results based on the amount of selected options, where out of all the selections made by respondents 47% were for the option *Yes* with 33 picks. Overall, option *Yes* was was chosen by 61% of all 54 respondents.

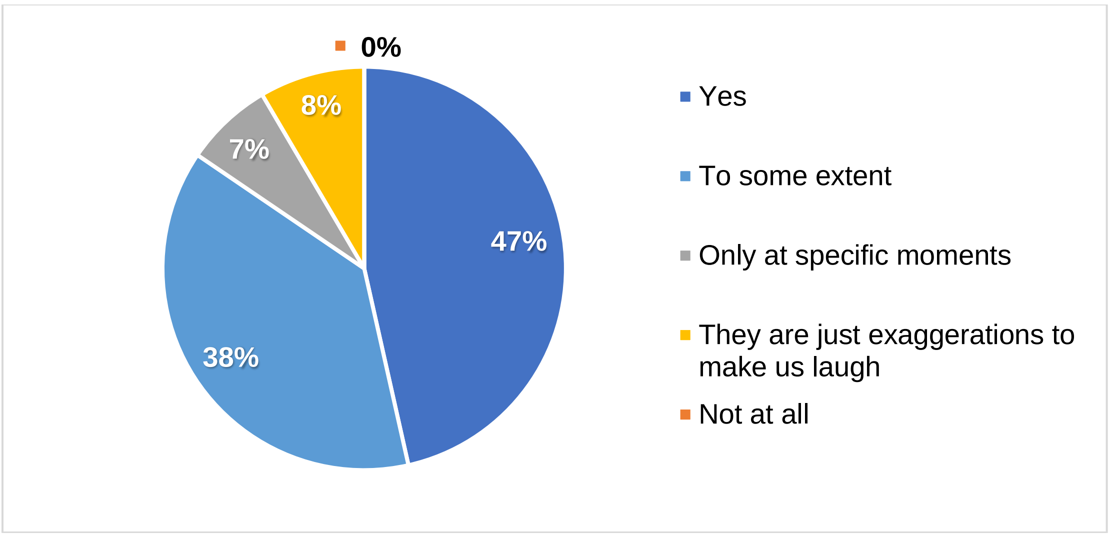


Figure 31. Percentage of selected opinions regarding memes representing the life of developers or not.

Then the students were asked if JavaScript frameworks & libraries have ever made them feel like the memes seen on the survey. In a similar fashion as the previous question, the respondents could choose one or more of the options given to them and although the great majority thought memes represented the life of developers, when put in contrast to their own experience with JavaScript, they felt slightly less represented less than the previous question with 43.3 % of the respondents selecting the *Yes* option as seen on Figure 32.

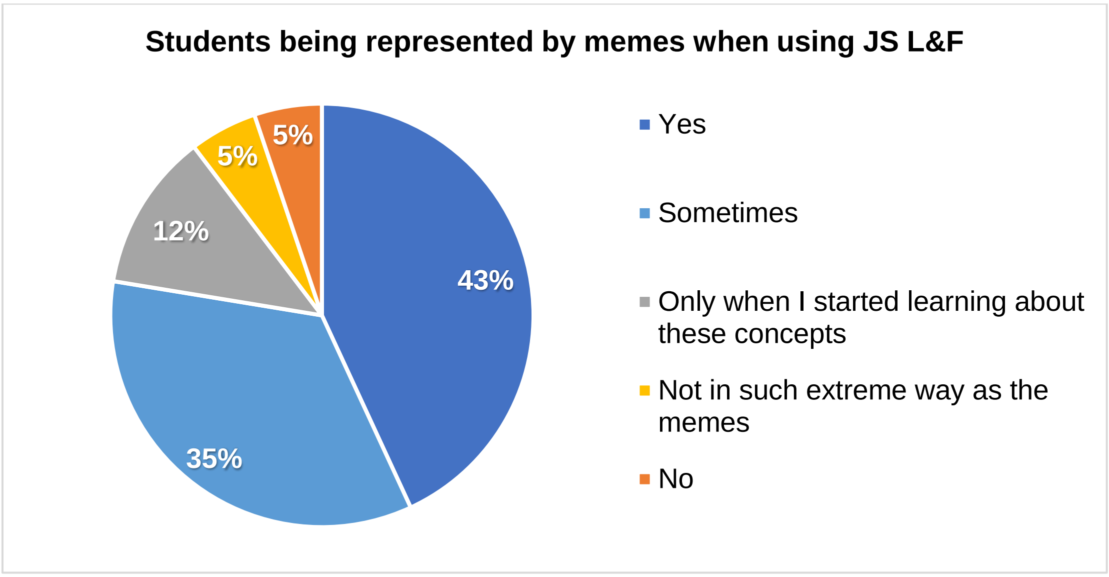


Figure 32. Students being represented by memes when using JavaScript L&F.

To conclude the survey the respondents were asked to describe in one word if possible, how using JavaScript libraries and frameworks makes them feel. The answers were varied and not all in one word, but all 54 students gave their answer as seen on the list below this paragraph.

Lost | Assistant | Exciting 😂 | Anxious | A learning process and a systematic approach | Comfortable | Chaotic | Makes it easier | JS frameworks make life easier in large projects but for small static sites its an overkill | Smooth | Intimidating! | Uncertain | Productive | Magic | Suicidal | annoyed? :D i dunno i dont do much of js anymore | I dont use them | Convenient | Confused | Ok | Frustrated | Not used | I don't know about them yet... | Important | Haven't used them yet as far as I know | Confused | lost | Weird | Nice | Incompetent | Good | Effective | Professional | Sick | Empty | Easier and faster in making web application | Efficient | Efficient | Alright | Well , I will learn all this things next semester, I am studding Programming 2 now | Useless | Happy | good | Confident | Stressed | Sad | Smart | Dead inside | comfortable | They simplify things. | coding cat on a planet where the language is the same in a hardly recognizable dialect | Challenged | comme ci comme ça | Like a noob.

List 1. Respondents feelings towards JavaScript libraries and frameworks

4.2 Semi-structured interview results

For this research semi-structured interviews were carried out on two BITe students who were close to graduate at the time of the interviews and for simplicity and keep the interviewees' anonymity, they will be referred as A and B.

At the time of each interview, both persons had been working as consultant developers for about 8 months on their respective companies. While A has been working with JavaScript technologies an estimated 80% of the time while working as developer, B had not been working much with JavaScript at his workplace. When asked about their use of libraries and frameworks at work, A was excitedly mentioning *React*, *React Native* and *Loadash* as the *"three biggest ones"*, while B had mostly worked with JavaScript on testing frameworks at the time of the interview.

With both interviewees having a considerable amount of working experience in the IT field, they could have a better understanding of which things would have been helpful to know before starting their careers. When asked about this in relation to JavaScript, libraries and frameworks, A replied with great ease *TypeScript*, because it was something learned at work and which was expected to be known from the start. In B's opinion, developers are constantly needing to learn new things and for that reason that B wished having known better methods for constant learning and particularly learning a *"framework for learning frameworks".* In B's experience, that issue has been one of the most difficult tasks to deal with on the path to become a developer.

Afterwards the interviewees were asked to share something learned during their BITe studies that they initially undervalued and learn to appreciate while working as professional developers. For interviewee A one thing taught at school and that they appreciated more during working life was the use of Git, as A is constantly involved on projects were the use of Git within a team is an essential daily tool. For B, debugging is something wished that had been taken more seriously at school, because it is also something developers have to constantly deal with.

When asked about their thoughts on how JavaScript, libraries & frameworks are taught during the BITe programme, both had several observations and recognized the value of the practical and theoretical teaching they received at BITe. A also said that most things taught at Haaga-Helia were quite useful, specially learning modern technologies, but would have liked having a course dedicated to JavaScript after being introduced to it in *Orientation to Programming* course, as it would be very useful and allow easier an easier transition to frontend and mobile development courses.

While replying the same question, B confessed not being personally satisfied with the way those technologies were taught during at BITe. While slightly hinting difficulties to learn with from some teachers, B also recognizes that *"some teachers are better than others for* *some people"* and that the material provided by some teachers is of great quality which can be revisited to this date. In addition, B mentioned about the difficulty of learning newer EcmaScript concepts while starting to learn React at the same time, which is perhaps the reason of B desiring having had a second follow-up React course at BITe.

To finalize the interview A and B were asked to suggest possible improvements to the teaching of such concepts at Haaga-Helia, to which A made recommends to keep practising the combination of learning theory and applying the theory in practice, along with giving more focus use of numbers in JavaScript as it is in fact not that simple and has been *"tricky"* to deal with in A's professional experience. Regarding this matter B put great emphasis on the importance of improving the quality of the learning material, provided during the studies. B suggests funnelling more resources to material creation and getting it to be peered reviewed (and perhaps even the teaching itself), that way there can be more consistency and higher quality of the learning experience.

5 Discussion

5.1 Comparison to initial assumptions and research questions

5.1.1 Question 1: Are BITe students discouraged in any way to continue learning and developing professionally in JavaScript related fields?

This thesis started with the assumption that BITe students might experience some level of discouragement from further learning of JavaScript and pursuing a programming career as they progressed in their BITe studies and were faced with more advanced and complex concepts of JavaScript. The data gathered on the survey showed that although there are reasons to believe that students could be discouraged from continuing their professional growth in JavaScript, those factors have not been enough to deter their interest in it. The students ability to keep track of the changes within JavaScript, libraries and frameworks scored a low average level of 3.57, but regardless only one person was not interested in BITe's JavaScript courses, in fact, 82.7% of the respondents were still interested in them or had already completed them all, with only few unsure about their interest in such courses.

Also, even with the increasing complexity of the language as more advanced topics are uncovered or the frequency of changes that JavaScript or its libraries and frameworks have faced in the last years, the survey respondents' confidence on their skills to work with such technologies remains on good levels as seen on Figures 33 and 34.

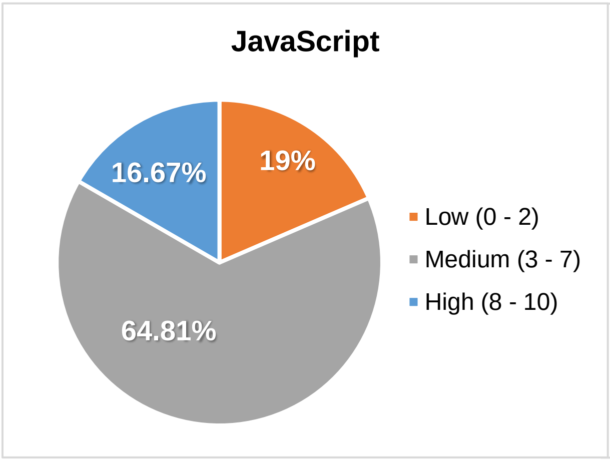


Figure 33. Respondents level of confidence on their JavaScript skills.

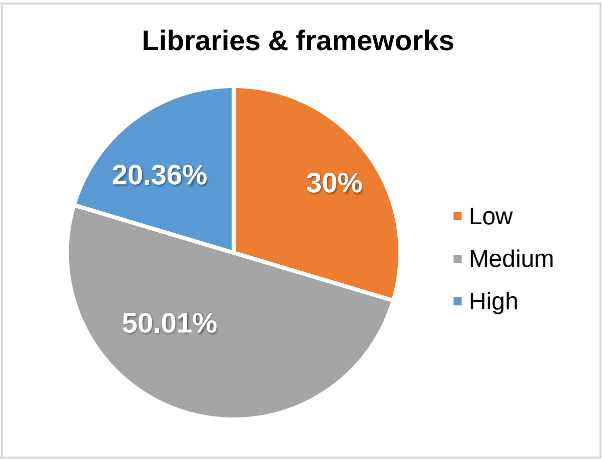


Figure 34. Respondents confidence on their libraries and frameworks skills.

The high levels of agreement with memes 3 and 4, representing some of the hurdles of being a developer, show that students recognise the complexity and effort involved in the work of developers. However, the mixed responses for meme 2 regarding the intense stress of being a developer, show that the challenges involved on this line of work, are not enough to discourage them. This idea is further supported by how little respondents fear dedicating too much of their free time to programming, with an average of 3.24 points out of ten. Even if that low average was due to their good coding or organizational skills or even deciding not sacrificing their free time at the expense of their progress, the data shows that using free time for programming is not an issue the respondents.

5.1.2 Question 2: What are BITe students' feelings regarding JavaScript, its frameworks and libraries?

The results regarding respondents' feelings towards JavaScript, libraries and frameworks, varied in length and not all represented a feeling as it was asked. While most answers represented feelings, nine of the answer were adjectives to describe the technology in question, while one other answer was not possible be categorized.

Figure 35 shows the results to the above mentioned respondents' feelings when categorized as positive, neutral, negative or unclear. The results show a clear mix of positive and negative responses, with a slight preference for positive ones. Among the neutral replies four of them were categorized as such, because of students answering not having used use those technologies yet, which has a potential for becoming a positive or negative feeling once they get to experience working with such JavaScript technologies.

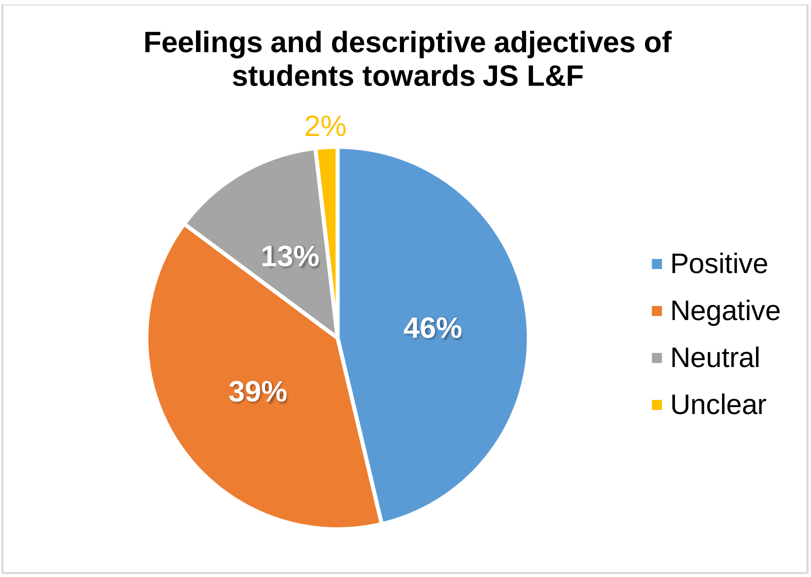


Figure 35. Feelings and descriptive adjectives of students towards JavaScript L&F.

The mix of feeling JavaScript libraries and frameworks have on the surveyed students can be further backed up considering that the level of agreement to all memes shown to the students had averages and medians above 6 and 7 respectively, therefore recognizing the feelings of surprise, confusion or a sense of growth as depicted on some of the memes.

5.1.3 Question 3: What possible improvements there could be on the JavaScript teaching methods in Haaga-Helia for BITe students?

Although the data gathered in this research showed that the students still have interest in developing further in JavaScript, regardless of the difficulties they might encounter and mix feelings they have towards this language and its environment, there are ways to improve the students' learning and engagement.

Figure 17 showed that the students have a mix of prior knowledge of JavaScript at the start of their studies in Business Information Technology programme at Haaga-Helia, but the great majority of them got to properly learn the language at BITe. While the electronic survey had no question asking students to give suggestions to improve their learning, Haaga-Helia asks student feedback at the end of each course they provide in order to improve and monitor the teaching and learning at the school. Nonetheless survey data showed that students have difficulties keeping track of JavaScript F&L changes (Figure 20), while at the same time a great portion of them considers using those tools in their projects (Figure 21) and even exploring other popular frameworks that have not been taught at school (Figure 23).

The interviews made during this study provided some light on possible improvements, as their experience of learning at BITe was still fresh and they had been putting their learnings and experiences from Haaga-Helia in a professional environment for long enough time, to provide rich insights. While the mix of theory and practical learning exercised at BITe has been a great asset, the inclusion of extra courses for JavaScript and React, can allow the students to better absorb deeper concepts within these technologies, allowing a better implementation of them on more complicated projects or across different frameworks or languages. As it was mentioned by interviewee B, providing stronger theory and methods to learn new concepts and technologies can greatly benefit students' progress and ability to adapt to each new subject even after graduation. Something that can also greatly benefit such goals, is putting more effort and resources into the quality of learning material, which in the long run can also benefit teachers as their teaching will become more effective and engaging.

5.2 New information and final conclusions

The data gathered in this study revealed that although students can easily identify themselves with the hardships and situations depicted in coding memes, they are not discouraged enough to not pursue JavaScript related career. Based on the data gathered, the students do have varied and sometimes conflicting feelings towards JavaScript and its libraries and frameworks, but they remain engaged and intrigued with its technology.

It could be said that even when participating in several JavaScript related courses, it doesn’t necessarily mean that the students will absorb enough knowledge to feel greatly confident in their skills or properly understand the greater depths of the learnt concepts. A higher standard in learning material can greatly influence students’ development but also adding advanced level courses for important subjects like JavaScript and React, can further help students absorb and apply deeper and richer knowledge. Even if the presence of mixed feelings towards the JavaScript environment persist, the skill level of the students can be increased through greater quality in their learning processes and tools.

The data showed a considerable interest of students towards different JavaScript technologies, that could be better taken advantage of. As time passes some technologies suddenly become obsolete or replaced and offering an extra course on emergent technologies can help decrease the backlash of sudden changes in the IT field.

As with any study, there are things that could have improved the quality of results if included or been presented differently. In an attempt to not guide the survey responses to specific answers, respondents were given freedom to write as much as they wished regarding their feelings towards JavaScript libraries and frameworks, but as shown previously, the replies varied in length and clarity. Perhaps a better approach for future investigations can be to give a pool of options, where the respondents could choose the three words that best represent their feelings towards such technologies. Additionally, something similar could be done in regard to what they think about those technologies, which can serve to show their appreciation towards them.

Also, since the research questions of this thesis were directed towards BITe students, it could have been convenient to ask all *research questions* directly on the survey and back up those answers with the additional questions on it. With that in mind, it is also important to note how the use of memes could facilitate could help the students to reflect about the issues surrounding the life of developers and their own experience towards becoming professionals on the field. The memes could also help give students a much needed comic and mental relief after replying several possibly serious and tiring questions, through stimulating visual humour that could also serve a practical investigation purposes.

Perhaps conducting more interviews to students that have had the opportunity to be working as developers for some months, could help better trace which factors remain constant for different interviewees (such as being satisfied with the mix of theoretical and practical learning during studies) and which factors are completely different for each person (such as undervaluing something learned at school or learning to appreciate what they have learned in professional life).

This research is focused on JavaScript libraries and frameworks, but similar studies could be done with other subjects or technologies, such as Java, Linux, Cloud services, mobile development, etc. Similar kinds of research can help evaluate the students' perceptions of such subjects and help improve their learning and confidence in their skills, as well as further improve the quality of teaching provided by Haaga-Helia, even if it is already in good shape. This study has shown that even if sometimes students have negative feelings towards a subject, facing its challenges and conquering them can be more rewarding than the frustrations in between. It is for that reason that it's important to keep supporting and improving the students' learning process and maximize their potential as future professionals.

References

Angular. (2020). AngularJS Releases. Retrieved January 2020, from https://github.com/angular/angular.js/releases?after=v0.9.4

Banks, A. & Porcello E . 2017. Learning React. 1st ed. O’Reilly, pp. 1. Sebastopol, CA.

Boduch, A. 2017. React and React Native. Packt Publishing, pp. 210. Birmingham, UK.

Brown, E. 2016. Learning JavaScript. 3rd ed. O’Reilly, pp. 265. Sebastopol, CA.

Cerruzi, P. 2012. Computing: A Concise History. The MIT Press Essential Knowledge Series. Cambridge.

ECMA International. History of Ecma, pp 2-3. Retrieved January 2020, from https://www.ecma-international.org/memento/history.htm

GitHub. (2019). The State of the Octoverse. Retrieved January 2020, from https://octoverse.github.com/

Guzdial, M. (2015). What's the best way to teach computer science to beginners? *Communications of the ACM, 58*(2), pp. 12-13. doi:10.1145/2714488

Haber, J. 2014. MOOCS. The MIT Press Essential Knowledge Series. Cambridge.

Henry, K. (2017). The Rise & Fall of jQuery. Retrieved February 2020, from https://keithhenry.github.io/jsmeetup-jquery-rise-fall/

Honeypot. (2020). Vue.js: The Documentary. Retrieved May 2020, from https://www.youtube.com/watch?v=OrxmtDw4pVI

Hoy, W. K. (2010). *Quantitative research in education: A primer, pp. 2, 14*. Los Angeles, [Calif.] ; London.

Hsu, T., Chang, S. & Hung, Y. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, pp. 296-310. doi:10.1016/j.compedu.2018.07.004

Macrae, C. 2018. Vue.js: Up and Running. 1st ed. O'Reilly, pp. 136. Sebastopol, CA.

Merriam-Webster Online. (2019). Algorithm. Retrieved April 2019, from https://www.merriam-webster.com/dictionary/algorithm

Merriam-Webster Online. (2019). Computer. Retrieved April 2019, from https://www.merriam-webster.com/dictionary/computer

Nørmark, K. (2011). Overview of the four main programming paradigms. http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

Payne, B. 2015. Teach Your Kids to Code. 1st ed. No Starch Press, p. xxi. San Francisco, CA.

Piva, M., & Vivarelli, M. 2018. Is innovation destroying jobs? firm-level evidence from the EU. Sustainability, 10(4), 1279, pp. 2. doi:http://dx.doi.org.ezproxy.haaga-helia.fi:2048/10.3390/su10041279

Rascia, T. (2018). Getting Started with React - An Overview and Walkthrough Tutorial. Retrieved January 2020, from https://www.taniarascia.com/getting-started-with-react/

Rauschmayer, A. (2014). Speaking JavaScript, chapter 4, 5 . Retrieved January 2020, from http://speakingjs.com/es5

React. (2020). A JavaScript library for building user interfaces. Retrieved January 2020, from https://reactjs.org/

Robertson, J. (2014). Rethinking how to teach programming to newcomers. *Communications of the ACM, 57*(5), pp. 18-19. doi:10.1145/2591203

Saunders, M. N. K. k., Lewis, P. & Thornhill, A. (2019). *Research methods for business students* (Eighth edition.), pp. 175. Harlow, England: Pearson.

Seshadri, S. 2018. Angular: Up and Running. 1st ed. O'Reilly. Sebastopol, CA.

Shifman, L. 2014. MEMES in digital culture. The MIT Press Essential Knowledge Series, pp. 9, 15. Cambridge.

Spinello, R. (2005). Competing Fairly in the New Economy: Lessons from the Browser Wars. Journal Of Business Ethics, 57(4), pp. 343-361. doi:10.1007/s10551-005-1832-6

Stack Overflow. (2019). Stack Overflow Developer Survey Results 2019. Retrieved May 2019, from https://insights.stackoverflow.com/survey/2019#technology

Sue, V. M. & Ritter, L. A. (2016). *Conducting online surveys* (Second edition.). Los Angeles: SAGE.

Terlson B., Farias B., Harband J. (2019). ECMAScript 2019 Language Specification, Retrieved January 2020, from https://www.ecma-international.org/ecma-262/10.0/index.html#Title

TypeScript. (2020). TypeScript - JavaScript that scales. Retrieved April 2020, from https://www.typescriptlang.org/

Wakil, K., Khdir, S., Sabir, L. & Nawzad, L. (2019). Student Ability for Learning Computer Programming Languages in Primary Schools. *International e-Journal of Educational Studies, 3*(6), pp. 109-115. doi:10.31458/iejes.591938

Wikipedia. (2020). Programming paradigm. Retrieved in May 2020, from https://en.wikipedia.org/wiki/Programming_paradigm

Appendices

Appendix 1. Survey Questions

BACKGROUND INFORMATION			
n°	Question	Answer Type	Additional Information
1	Which option identifies you?	Selection	Gender information
2	Nationality	Text Field	
3	Age Range	Selection	
4	Start of studies at BITe (Business Information Technology at Haaga-Helia)	Text Field	(Ex. Fall 2017)
5	Was an IT degree your priority before coming to Haaga-Helia's BITe?	Selection	Yes | No | Not sure
6	Do you pay for your studies at Haaga-Helia?	Selection	Yes | No
7	Have you always been interested of participating in BITe's programming courses?	Selection	Yes | No
8	Are you still interested in BITe's programming courses?	Selection	Yes | No | Not sure | Already finish them

JAVASCRIPT			
n°	Question	Answer Type	Additional Information
9	Did you know anything about JavaScript, before studying at Haaga-Helia?	Selection	Yes (a lot) | Yes (a bit) | The name was familiar | Nothing at all
10	Have you started or completed any JS related course at Haaga-Helia?	Selection	Yes | No
11	How confident do you feel about your JS skills?	Slider Range	0 – 10 (Not confident - Very confident)
12	How confident do you feel about using JS libraries & frameworks?	Slider Range	0 – 10 (Not confident - Very confident)
13	When working with JS libraries & frameworks, are you able to keep track of the changes or lack of update of them? (such as security updates or changes in their usage)	Slider Range	0 – 10 (No – Yes)
14	Do you use or consider using libraries on your projects?	Selection	Always | When needed | Sometimes | No
15	Have you been able to get a good grasp of React?	Selection	Yes | No, but I'm planning to | No and I have no interest in it | Not sure

16	Are you planning to explore frame-works such as Angular or Vue?	Selection	Yes | No | Not sure
17	Do you fear needing to dedicate too much of your free time to program-ming?	Slider Range	0 – 10 (No – Yes)
MEMES			
n°	**Question**	**Answer Type**	**Additional Information**
18	Do you usually feel surprised if your code works on the first try? as de-picted on this meme	Slider Range	No – Yes
19	Do you often feel stressed coding?	Slider Range	No – Yes
20	Do you often feel represented by this meme?	Slider Range	Not knowing why your code works or does not work
21	Does this happen to you?	Slider Range	Meme depicting using 4 times the amount of time to fix bugs than to write initial the code
22	Do you google often when coding?	Slider Range	Never – When necessary – All the time
23	Do you ever realise how much your skills have improved (or how bad they were) after reviewing your old code?	Slider Range	No – Sometimes – Yes
CLOSING QUESTIONS			
n°	**Question**	**Answer Type**	**Additional Information**
24	Do the previous memes represent life of developers in general?	Multi-selection	Yes | To some extent | Only at specific moments | They are just exaggerations to make us laugh | Not at all
25	Have the use JS frameworks & li-braries ever made you feel like what those memes depicted?	Multi-selection	Yes | Sometimes | Only when I started learning about these concepts | Not in such extreme way as the memes | No
26	Please tell (in one word if possible), how using JavaScript frameworks & libraries make you feel.	Text field	

Appendix 2. Prize winner selection (Code snippet)

```javascript
function randomizer() {
  return Math.floor(Math.random()*51) + 1;
}

winner_1 = randomizer();
winner_2 = randomizer();

do {
    winner_2 = randomizer();
} while (winner_1 == winner_2);

console.log(`\nOut of 51 participants, the winners are:
%ccontestant n°${winner_1} with 2 movie tickets
contestant n°${winner_2} with 2 movie tickets`,
"color: coral; font-size: 20px;");
```

Appendix 3. Survey transcripts (Confidential)

Appendix 4. Semi-structured interview scripts (Confidential)