Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 1. pp. 339–350
doi: 10.14794/ICAI.9.2014.1.339

Teaching introductory programming with JavaScript in higher education

Győző Horváth, László Menyhárt

Department of Media & Educational Informatics,
Eötvös Loránd University, Budapest, Hungary
gyozo.horvath@inf.elte.hu
menyhart@inf.elte.hu

Abstract

As the Internet penetration rate continuously increases and web browsers show a substantial development, the web becomes a more general and ubiquitous application runtime platform, where the programming language on the client side exclusively is JavaScript. This is the reason why recently JavaScript is more often considered as the lingua franca of the web, or, from a different point of view, the universal virtual machine of the web. In addition, the JavaScript programming language appears in many other areas of informatics due to the wider usage of the HTML-based technology, and the embedded nature of the language. Consequently, in these days it is quite difficult to program without getting in touch with JavaScript in some way.

In this article we are looking for answers to how the JavaScript language is suitable for being an introductory language in the programming related subjects of the higher education. First we revisit the different technologies that lead to and ensure the popularity of JavaScript. Following, current approaches using JavaScript as an introductory language are overviewed and analyzed. Next, a curriculum of an introductory programming course at the Eötvös Loránd University is presented, and a detailed investigation is given about how the JavaScript language would fit in the expectations and requirements of this programming course. Finally, the supported platforms and integrated development environments (IDEs) are also reviewed from the point of view of beginner programmers just started to programming.

Keywords: JavaScript, programming, higher education

MSC: 97Q60, 97B40

1. Introduction

IT knowledge is changing very fast, so we need to think over rather frequently the content and applied technology of computer science courses. On the one hand, this

is very important from the technological point of view, as the different technical tools in teaching (programming languages, development environments and platforms) may become out-of-date as the overall IT sector develops. On the other hand, education has to adapt to those cognitive, perceptual structures that typify and determine the students themselves and their environment they grew up in. IT development has a great influence on how the new generation of pupils is thinking, and what kind of expectation they raise. Education has to answer for these changes, and has to modify its ways to the students' interests from time to time.

What we see nowadays is that the web is becoming a general, ubiquitous application runtime platform, through which information and services are continuously available independently of place, time and device. Not only the number of web applications but their size and complexity are also increasing, while they need to satisfy the needs of the wider and wider user basis which require easy-to-use, clear, fast-operable user interfaces. In this process the rapidly developing web technologies have a great role. Web standards try to transplant many aspects of desktop environments to the web platform in the form of different kind of programming interfaces (APIs). Web browsers, being the runtime environment of web applications, show a substantial development, incorporating these standards and APIs, and executing client side code faster and faster. Finally, the programming language, which operates these client side technologies, is exclusively JavaScript in this viral client side environment. That is the reason why JavaScript is considered as the lingua franca, or the universal virtual machine of the web [1,2].

JavaScript, however, does not live only in web browsers, but tries to conquer the traditional components of the server side environment. For example, Node.js [3] – a platform built on Google Chrome's V8 JavaScript engine, basically an asynchronous runtime environment for command line JavaScript – is a very competitive alternative for serving out HTTP requests, replacing the traditional web servers, such as Apache, in some specific scenarios, e.g. as the server side of real time web applications. The traditional relational database management systems also have their own challengers, as alternative databases are showing up under the NoSQL flag. One of the most popular alternative database engine is the document-based MongoDB database ecosystem, which uses JavaScript internally on the server-side for certain options, and has a shell that is based on JavaScript [4].

Furthermore, web technologies, including JavaScript, are leaking into more and more areas of application development. Newer operating systems and desktop environments, e.g. Windows 8 [5] or Gnome 3 [6], give the opportunity to write applications using HTML, CSS and JavaScript. Mobile and browser based operating systems, like FirefoxOS and ChromeOS, have this options as well. JavaScript finds its way to the embedded environments and home automations, as the programming language of these small devices and microcontrollers [7,8,9].

Consequently, JavaScript appears on almost every platform, and hardly could avoid getting touch with it in some ways. But questions arise. Does the curriculum of related courses in higher education reflect the renaissance and hegemony of JavaScript? Is it important to reflect it at all? Web technology classes necessarily

deal with this programming language, but the main priorities are the technologies themselves, not the operating programming language. Subsequently, are undergraduate students prepared for meeting with JavaScript? Could this meeting be facilitated by teaching introductory programming in JavaScript? Is the language itself applicable for introductory programming language at all? Are there proper IDEs and tools to support the process of education?

In this article we are looking for answers whether JavaScript can be used as a programming language in introductory programming courses, and what the advantages and drawbacks of this application are. To achieve this, similar approaches are investigated in different universities and among online courses first, next the methodology of introductory programming in Eötvös Loránd University is presented, and finally we explore how JavaScript fits in this curriculum as an introductory language.

2. Current approaches using JavaScript as an introductory language

Teaching programming for beginners with JavaScript is not without precedence. There are many good examples which use JavaScript to introduce computer programming to novice programmers. Two main categories can be distinguished according to the form of these courses. The first group contains the different kind of online courses [10,11,12]. These online courses can be used by anybody surfing the web interested in programming, but the targeted age group is primarily the youths from secondary schools to endear programming. This aim is reflected on the overall layout and operation of these sites. Design is youthful with high colours and cartoonish graphical elements, the tasks are mainly games, and the whole learning process is gamified to maintain interest (e.g. collecting points, badges, completing levels). The solving process is divided into smaller steps, and the user interface has to be configured in a way that promotes self-supporting problem solving without any human intervention.

These online courses lean on the ubiquitous feature of the web platform through which anybody anywhere and anytime can use their services. The web platform offers interesting enough components in the web browser to work with, such as HTML elements, images, text, canvas, and so on. And these components can be manipulated only with JavaScript because it is the exclusive programming language of client side web applications. Due to the relatively high level of the tasks, frameworks or some kind of abstract interface are provided to the user.

The other main category contains the different introductory computer science classes in universities [13,14]. They also use some kind of high-level framework to work with, but their user interfaces do not need to be over-sophisticated because they are used in contact lessons with the help of the teachers. A common characteristic with the courses from the first group is that they use the interesting nature of the web platform to motivate students in their progression, and this choice leads

to JavaScript as the programming language of this platform.

Summarizing, these environments providing introduction to computer-aided problem solving chose JavaScript as a programming language because their tasks are being realized on the web platform. This platform has several advantages. From the technological point of view, it is always online, ubiquitous, universal, and it can be used in any web browsers independently from the host environment or device, and is always ready, there is no need to setup. From the students' aspects, the web browser is a modern, trendy environment which they are familiar with and fun to work with through different HTML elements (texts, tables, images, canvas, etc.), and in a web application there is the chance to give the learning process a social manner (e.g. sharing). From the teachers' aspects, there is no need for compilation and build processes, the results can be seen immediately, and arbitrary level of abstraction can be prepared.

3. Curriculum and methodology of an introductory programming course

There are many different ways to teach programming for beginners [15,16,17]. At the Eötvös Loránd University an algorithm-first (or structure-first) approach is applied, analogue programming and specifically programming theorems being in the focus. The main topic of this class is not about giving introduction for coding or teaching everything about a specific programming language. Instead, it tries to give tools and techniques for systematic problem solving in the field of programming tasks. This systematic approach first specifies the problem itself in an abstract way with mathematical tools (specification), discovering the input and output data structures behind the textual information of the task, and establishing connection between the input and output data. The next step is finding out how to solve the problem, defining the steps in an abstract language that produce the right output data from the input ones. The result of this section is an algorithm. And finally, only the last step deals with the coding itself, namely finding a tool, a programming language, which helps to translate the steps of the abstract algorithm to machine language. In this systematic approach the implementation and the programming language is just a tool, not the aim itself. The main focus is kept on data modelling (specification part) and breaking down solution into elementary steps (algorithm part).

If we investigate this problem solving procedure from the implementation side, the following topics need to be covered in a programming language (as a corresponding implementation of an algorithm):

- Input/output (console, file)
- Data types and data structures (primitive types and composite types, such as arrays and records)
- Control structures

- Programming theorems
- Decomposition to sub-programs (functions)

As the programming language is just a tool not the final goal in this systematic approach, we do not have to stick to a specific one, but rather we can set up expectations a programming language must fulfil, leaving a big flexibility in choosing the proper one. An introductory programming language must suit the following requirements [18]:

- Simplicity (easy to learn)
- Consistency
- Typicality (similar to the algorithmic language)
- Usability (advanced programming style: top-down design, control structures, enumeration types, free of dangerous possibilities, modularity)
- Forgiving syntax
- Flexibility
- Easy to setup
- Good documentation
- Good base for subsequent languages
- Good IDE

4. JavaScript as an introductory programming language

In this section the JavaScript programming language is investigated in order to decide whether it is an applicable programming language for introductory programming courses and fulfils the corresponding requirements. For this, we take a specific example of summing values in an array which helps us examining the above requirements. The example is implemented in three different programming language: C++, FreePascal and JavaScript.

4.1. Specification, algorithm and final implementation

In this specific example a sequence of numbers are given, and we have to calculate the sum of these numbers. Figure 1 shows the corresponding specification and the solving algorithm can be seen in Figure 2. Both of them reflect the summing programming theorem. The final implementations in the examined programming languages are given side by side in Table 1. For the sake of comparison, there is no reading of input variables, instead prefilled arrays are used.

```
Input:      n: Integer
            x: Array[1..n: Real]
Output:     s: Real
Pre-cond:   n>=0
Post-cond:  s = sum(i=1..n) x[i]
```

Figure 1: The specification of summing values

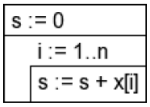

Figure 2: The Nassi–Shneiderman diagram of summing values

FreePascal

```
program sumup;
//declaration
const n=5;
type TX=array[1..n] of real;
var x:TX=(1,3,5,7,9);
var s:real;
var i:integer;
begin
  //process
  s:=0;
  for i:=1 to n do begin
    s:=s+x[i];
  end;

  //write output
  writeln('Sum: ', s:5:2);
end.
```

C++

```
#include <iostream>
using namespace std;
int main() {
  //declaration
  const int n=5;
  double x[]={1,3,5,7,9};
  double s;

  //process
  s=0;
  for(int i=0; i<n; i++) {
    s=s+x[i];
  }

  //write output
  cout<<"Sum: "<<s<<endl;
  return 0;
}
```

JavaScript

```
//declaration
var x=[1,3,5,7,9],
    s, i;

//process
s=0;
for(i=0; i<x.length; i+=1) {
    s=s+x[i];
}

//write output
console.log('Sum: ', s);
```

Table 1: The implementations of the example in FreePascal, C++ and JavaScript

4.2. Syntax

JavaScript has a bad reputation according to some of the programmers [19]. One component of this opinion is undoubtedly the fact that the JavaScript language has some very bad parts due to language design errors [20]. In an introductory class, however, not every aspects of the language are covered, but only those parts which are necessary to implement the algorithmic structures. These are the variable declarations, assignments, basic operators, conditionals, loops and functions. Consequently, only a subset of the language is used during implementation. This subset has many things in common with the C-like languages, and does not contain any conspicuous bad part. This statement can be verified in Table 2, where the implementation of the core algorithm is presented in the investigated languages. This table shows that a limited set of language tools is needed to express the algorithm in code. The three implementations are very similar to each other, only the

language-specific syntax is different as expected.
The safety of this language subset can be increased with further tools in JavaScript. In the 5th version of ECMAScript [21], the standard behind JavaScript, a strict mode was introduced to remedy some shortcomings of the language. Furthermore, different tools can help to avoid the erroneous part of the language, such as linters [22,23].

FreePascal	C++	JavaScript

`s:=0;`
`for i:=1 to n do begin`
`  s:=s+x[i];`
`end;`
	`s=0;`
	`for(int i=0; i<n; i++) {`
	`  s=s+x[i];`
	`}`
		`s=0;`
		`for(i=0; i<x.length; i+=1) {`
		`  s=s+x[i];`
		`}`

Table 2: The implementations of the core algorithm in FreePascal, C++ and JavaScript

4.3. Types

One of the biggest drawbacks of using JavaScript in an introductory course is its dynamic-typed nature. During data modelling in the specification variables with type and structure are distinguished which hold the input and output data. Types are very important aspects in modelling the real world entities. In JavaScript, however, there is no way to indicate the data type expectations during variable declaration (see the last column in Table 3), as not the variables, but their values have a type.

There are different efforts to give JavaScript a statically-typed nature. TypeScript language [24] is one example of this efforts. This language is a typed superset of JavaScript which compiles to plain JavaScript. In this language the specific type can be indicated optionally at variable declaration, function parameters and return values. Running the program requires two steps: a TypeScript-to-JavaScript compilation and interpreting the compiled JavaScript. Table 3 shows the data description part of the specification, the corresponding TypeScript and the compiled JavaScript code.

Type coercion is also a mess in JavaScript which can confuse beginner programmers. To avoid such confusions, the strict `===` operator should be used instead of the more permissive `==` operator, and falsy values must be replaced with explicit comparisons in logical expressions.

4.4. Data structure literals

Comparing the array definition of the FreePascal, C++ and JavaScript versions, it is apparent that the JavaScript array literal is very compact and expressive contrary to the others. Generally speaking, JavaScript object and array literals are very expressive. That is the reason of the high popularity of the JSON (JavaScript

Data description	TypeScript declaration	JavaScript declaration
Input: n: Integer x: Array[1..n: Real] Output: s: Real	var x : number[]; var s : number; var i : number;	var x; var s; var i;

Table 3: Comparing the data description in the specification with the TypeScript and JavaScript variable declarations

Object Notation) data format [25], which is no more than the standardized version of these literal formats. The JavaScript array has dynamic length which can be useful for the beginner programmers. Unfortunately, the elements contained in the array are also dynamics, but this can be restricted with a statically-typed addition, e.g. TypeScript.

```
var x : number[] = [1, 3, 5, 7, 9];
```

4.5. Input and output operations

It is possible to write programs with built-in data, but it is a legitimate demand to determine the program input by the user. It makes the programs much more interactive, flexible and testable. JavaScript, being an embedded language, has no input/output commands. The runtime environment has to ensure the interfaces through which input and output operations become available. The runtime environment can be anything, but the two major typical environments are the web browsers and the Node.js command line ecosystem. With Node.js it is possible to read input from the command line or file. In Table 4 the reading mechanisms are compared in the three investigated languages. Here we introduced a module (cio) which makes it possible to read different types of data into variables through higher-level methods (readInt, readDouble, etc.).

FreePascal	C++	JavaScript (Node.js)

//read input
write('N = ');
readln(n);
for i:=1 to n do begin
  write(i, '. elem: ');
  readln(x[i]);
end;
	//read input
	cout<<"N = ";
	cin>>n;
	for (int i=0; i<n; i++) {
	  cout<<i+1<<". elem: ";
	  cin>>x[i];
	}
		var cio=require('cio');
		//declaration
		var n;
		var x = [];
		var e;
		//...
		//read input
		n=cio.readInt('N = ');
		for (i=0; i<n; i++) {
		  e = cio.readDouble((i+1) +
		  '. elem: ');
		  x.push(e);
		}

Table 4: Reading from console in FreePascal, C++ and JavaScript in Node.js runtime environment

In web browsers input can be carried out through HTML elements (like form fields, e.g. textarea) or with the global prompt() method. Output can be written to the console with the console.log() command, which works in both environments, and to HTML elements in browsers.

4.6. Program structures

Table 5 shows that JavaScript has a very simple program structure. This may be understood better for a beginner than accepting a more verbose start-up program.

FreePascal	C++	JavaScript

`program foo;`
`//declaration`
`//...`
`begin`
`  //read input`
`  //...`
`  //process`
`  //...`
`  //write output`
`  //...`
`end.`
	`#include <iostream>`
	`using namespace std;`
	`int main() {`
	`  //declaration`
	`  //...`
	`  //read input`
	`  //...`
	`  //process`
	`  //...`
	`  //write output`
	`  //...`
	`  return 0;`
	`}`
		`//declaration`
		`//...`
		`//read input`
		`//...`
		`//process`
		`//...`
		`//write output`
		`//...`

Table 5: Basic program structures in FreePascal, C++ and JavaScript

4.7. Functions

Considering the target group, the drawbacks of function declaration in JavaScript comes from the dynamic type system. In JavaScript neither the parameters, nor the return value are not marked with type indication. This feature can be improved with the help of TypeScript as it can be seen in Table 6.

JavaScript	TypeScript

`function sumup(x) {`
`  //...`
`  return s;`
`}`
	`function sumup(x : number[]) : number {`
	`  //...`
	`  return s;`
	`}`

Table 6: Defining function in JavaScript and TypeScript

4.8. Modules

Separating the program code into different units and making reusable program modules can be a requirement in an introductory class as well. In contrary to the

FreePascal unit or the C++ includes, JavaScript has no built-in module language construct. The demand from web developer communities resulted in some temporary solutions. In Node.js the CommonJS format [26], while in web browsers the RequireJS format [27] became popular. The next version of ECMAScript contains a module definition format which brings solution for these diverse procedures [28].

4.9. Development environments

Beside the programming language, the development environments have also a big impact on the overall experiences in introductory classes. A bad, inconsistent, out-of-date IDE can frighten away beginner students. This is the reason why the proper development environment is as important as the proper programming language. Beyond the general requirements (e.g. debugger) [29], a special expectation can be set up for a JavaScript IDE: it has to support the different kind of additional tools which make JavaScript acceptable for an introductory language. These tools contain the TypeScript analyzer and compiler, and different linters, or additional add-ons, such as code beautifiers.

Two different IDEs can be distinguished according to the runtime environments. Command line programs can be written in Node.js and usually coding is realized in light- or heavy-weighed text editors. The basic functions of these editors can be extended with additional plug-ins, like linter and beautifier tools. TypeScript support is very weak; it can be reached from the command line. Node.js debugging is also available as an external tool (node-inspector), but it is using the debugging console of Google Chrome browser. There are promising projects that integrate all these features into one IDE, e.g. nodeclipse.

The other possible type of IDEs sits inside the web browser as a web application. It is based on the runtime environment of the web browser. There are many JavaScript sandboxes which make it possible to write HTML, CSS and JavaScript, and show the HTML output or the console, e.g. JSFiddle or JSBin. The debugging feature can be utilized from the console of the browser, and there are initiatives which give TypeScript support for these online playgrounds. However, the target groups of these environments are not students, but rather developers.

5. Conclusions

JavaScript is one of the most widely used programming languages nowadays. It gains its popularity due to the big success of the web platform and web technologies. These web technologies represent the modern computer environment, and they are interesting enough to motivate students. This is the reason why it is actual considering the possibilities to get students familiar with the basics of programming through the JavaScript programming language. Working in a browser environment has several advantages: it is modern and familiar, it is available everywhere with minimal or no setup platform-independently, gives immediate results thus more

satisfying for the students, and with the help of the internet a social aspect can be involved in the learning process.

In this article we investigated how the JavaScript language itself is applicable for introductory programming courses in higher education. It turned out that JavaScript is suitable for this task. It does not have much difference from the traditional educational languages (like FreePascal or languages from the C-family). In fact, in the most common scenarios the implementation of the main algorithm is very similar to the others. Among the differences there are advantages and disadvantages as well. Its syntax is forgiving, the program structure is simple, and the data structure literal formats are very expressive and compact. The main drawbacks come from the lack of specific language features. Dynamic typing makes it difficult to implement and work with the data description model, implicit conversions can lead to undesirable results. A module system is also missing from the language. Fortunately, these shortcomings can be eliminated with additional tools, like TypeScript, linters, strict mode and module systems.

References

[1] ATWOOD, J., JavaScript: The Lingua Franca of the Web, Online (2007), `http://blog.codinghorror.com/javascript-the-lingua-franca-of-the-web/`

[2] MAYER, C., JAXconf Community Night - Crockford lauds JavaScript, Raspberry Pi coolness, Online (2012), `http://jaxenter.com/jaxconf-community-night-crockford-lauds-javascript-raspberry-pi--coolness-43599.html`

[3] Node.js, Online, `http://nodejs.org/`

[4] MongoDB - JavaScript Language Center, Online, `http://docs.mongodb.org/ecosystem/drivers/javascript/`

[5] Windows API reference for Windows Runtime apps, Online, `http://msdn.microsoft.com/en-us/library/windows/apps/br211377.aspx`

[6] Answering the question: "How do I develop an app for GNOME?", Online (2013), `http://treitter.livejournal.com/14871.html`

[7] Johnny-Five - JavaScript Arduino programming framework, Online, `https://github.com/rwaldron/johnny-five`

[8] WILLIAMS, G., Espruino: JavaScript for Things, Online (2013), `https://www.kickstarter.com/projects/gfw/espruino-javascript-for-things`

[9] THIEL, W., heimcontrol.js, Online, `http://ni-c.github.io/heimcontrol.js/`

[10] CodeAvengers, Online, `http://www.codeavengers.com/`

[11] CodeHs, Online, `http://codehs.com/`

[12] Khan Academy, Online, `https://www.khanacademy.org/computing/cs`

[13] CS101 - Introduction to Computing Principles, Online, `http://www.stanford.edu/class/cs101/`

[14] Introduction to Computer Science, Online, https://www.cs.drexel.edu/~introcs/Fa13/

[15] GRIFFIN, J., FICKETT, M., POWELL, R., MENEZES, R., and CHEN, L., Objects-First, Algorithms-Early with BotWorld, University of Pennsylvania (2009)

[16] BENNEDSEN, J., CASPERSEN, M. E., Programming in context: a model-first approach to CS1, In SIGCSE '04: Proceedings of the 35th SIGCSE technical symposium on Computer science education (2004), 477–481.

[17] SAJANIEMI, J., HU, C., Teaching Programming: Going beyond "Objects First", University of JOENSUU (2006)

[18] HEIZLERNÉ BAKONYI, V., SZLÁVI, P., Evaluation of programming languages from educational perspective, Proceedings of the 9th International Conference on Applied Informatics (2014, to be published)

[19] CROCKFORD, D., JavaScript: The World's Most Misunderstood Programming Language, Online (2001), http://javascript.crockford.com/javascript.html

[20] CROCKFORD, D., JavaScript: The Good Parts, O'Reilly Media (2008)

[21] Ecma International, ECMAScript Language Specification, 5.1 (2011)

[22] JSLint, Online (2002), http://jslint.com/

[23] JSHint, Online (2011), http://jshint.com/

[24] TypeScript, Online (2012), http://www.typescriptlang.org/

[25] Ecma International, The JSON Data Interchange Format, Online (2013), http://www.json.org/

[26] CommonJS, Online (2011), http://wiki.commonjs.org/wiki/CommonJS

[27] RequireJS, Online (2011), http://requirejs.org/

[28] EcmaScript Harmony modules, Online (2014), http://wiki.ecmascript.org/doku.php?id=harmony:modules

[29] HEIZLERNÉ BAKONYI, V., SZLÁVI, P., Evaluation of developing environments from educational perspective, Proceedings of the 9th International Conference on Applied Informatics (2014, to be published)