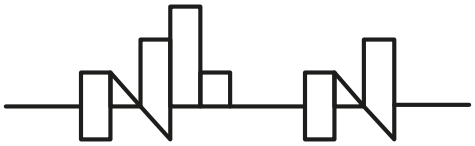



alfatec


UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA


EDBO
European Digital Bootcamps

JavaScript

Outline

- Overview of JavaScript (JS) Language
- Node.js
- The JavaScript event loop
- NPM (Node Package manager)
- V8 turbo-charges JavaScript by leveraging C++
- Events
- Bibliography

Overview of JavaScript (JS) Language

- Web Browsers are simple readers for web pages.
- Web browser were designed to display text with format capabilities.
- Virtually Web browsers does not have capabilities on their own (except for the ability to display text)
- As soon as Microsoft released its Internet Explorer browser, the browser wars were on, and the features started flying!
  - One browser introduced the ability to display images
  - Another introduced the capability to have different fonts, and then blinking text, moving text
  - All sorts of other impressive capabilities were introduced
- Someone got the idea that browsers could actually do useful things themselves, rather than just acting as fancy document display programs
- In 1995 (Netscape) Brendan Eich wrote JavaScript in record time by borrowing many of the best features from various other programming languages

Overview of JavaScript (JS) Language (cont.)

- Mocha was the original name of JavaScript
- It was renamed LiveScript with the first beta deployment of Netscape Navigator
- It was then changed to JavaScript when it was built into the Netscape 2 browser in 1995.
- Microsoft very quickly reverse-engineered JavaScript and introduced an exact clone of it in Internet Explorer, calling it Jscript in order to get around trademark issues.
- Netscape submitted JavaScript to the standards organization known as Ecma International, and it was adopted and standardized as EMCAScript in 1997.
- **Brandon Eich** commented about the name of the standardized language; stating that ECMAScript was an "unwanted trade name that sounds like a skin disease"
- When JavaScript debuted, it quickly became very popular as a way to make web pages more dynamic.
  - So-called Dynamic HTML (DHTML) was an early result of JavaScript being built into web browsers
  - It enabled all sorts of fun effects, like the falling snowflake effect , pop-up windows, and curling web page corners
  - But also more useful things like drop-down menus and form validation.

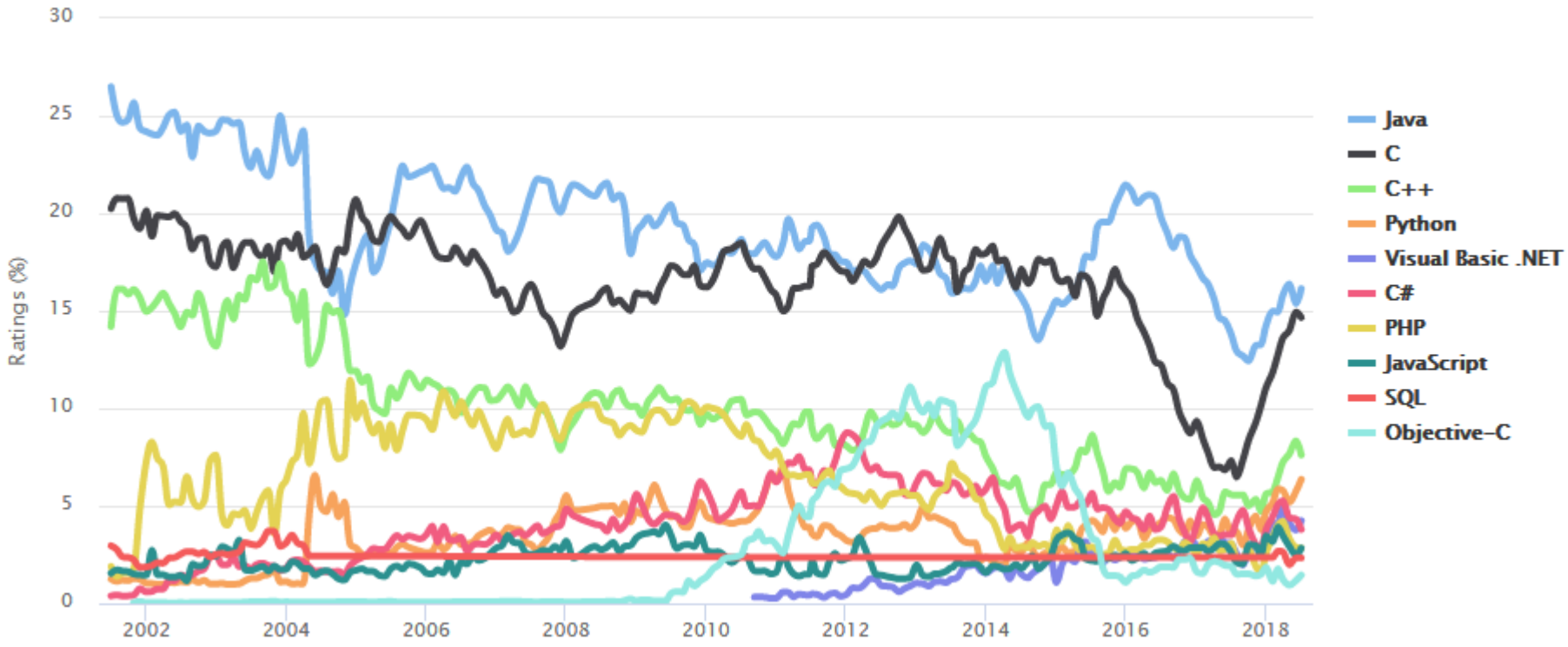
Overview of JavaScript (JS) Language (cont.)

May 2019	May 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.005%	-0.38%
2	2		C	14.243%	+0.24%
3	3		C++	8.095%	+0.43%
4	4		Python	7.830%	+2.64%
5	6		Visual Basic .NET	5.193%	+1.07%
6	5		C#	3.984%	-0.42%
7	8		JavaScript	2.690%	-0.23%
8	9		SQL	2.555%	+0.57%
9	7		PHP	2.489%	-0.83%
10	13		Assembly language	1.816%	+0.82%
11	15		Objective-C	1.626%	+0.69%
12	12		Delphi/Object Pascal	1.406%	+0.39%
13	18		Perl	1.394%	+0.48%
14	16		MATLAB	1.366%	+0.44%
15	10		Ruby	1.343%	+0.16%
16	17		Visual Basic	1.317%	+0.40%
17	91		Groovy	1.173%	+1.06%
18	19		Swift	1.150%	+0.24%
19	**14**		**Go**	**1.114%**	**+0.14%**
20	22		PL/SQL	1.017%	+0.12%

https://www.tiobe.com/tiobe-index/

Overview of JavaScript (JS) Language (cont.)

https://www.tiobe.com/tiobe-index/

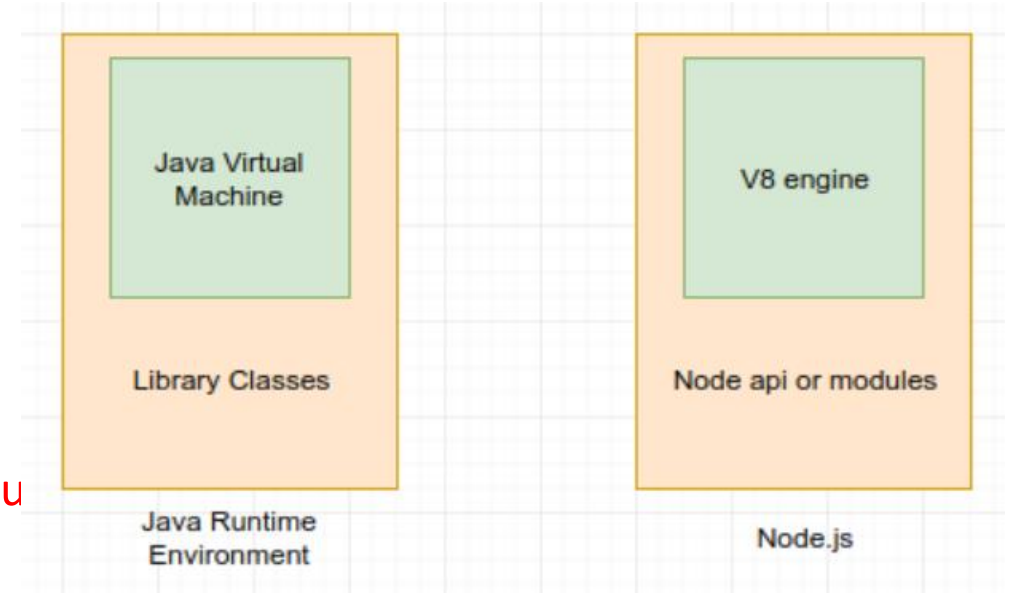


Overview of JavaScript (JS) Language (cont.)

- JavaScript is flexible enough that it can be used and learned by nonprogrammers
- But powerful enough that it is used by professional programmers to enable not only front-end programming but also for back-end programming

Node.js

- Node.js is a JavaScript runtime environment
- Node run-time environment includes everything you need to execute a program written in JavaScript
- Node.js came into existence when the original developers of JavaScript extended it from something you could only run in the browser to something you could run on your machine as a standalone application
- Programmers can do much more with JavaScript than just making websites interactive.
- Both your browser JavaScript and Node.js run on the V8 JavaScript runtime engine
- V8 engine takes your JavaScript code and converts it into a faster machine code
  - Machine code is low-level code which the computer can run without needing to first interpret it



Formal definition as given on the official Node.js website:
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.
Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Node.js (cont.)

- I/O refers to input/output and it takes time and hence blocks other functions
  - It can be anything ranging from reading/writing local files to making an HTTP request to an API.

Example:

- Consider a scenario where we request a backend database for the details of user1 and user2 and then print them on the screen/console. The response to this request takes time, but both of the user data requests can be carried out independently and at the same time. Blocking I/O (left) vs Non-Blocking I/O (right)

Node.js (cont.)

Blocking I/O

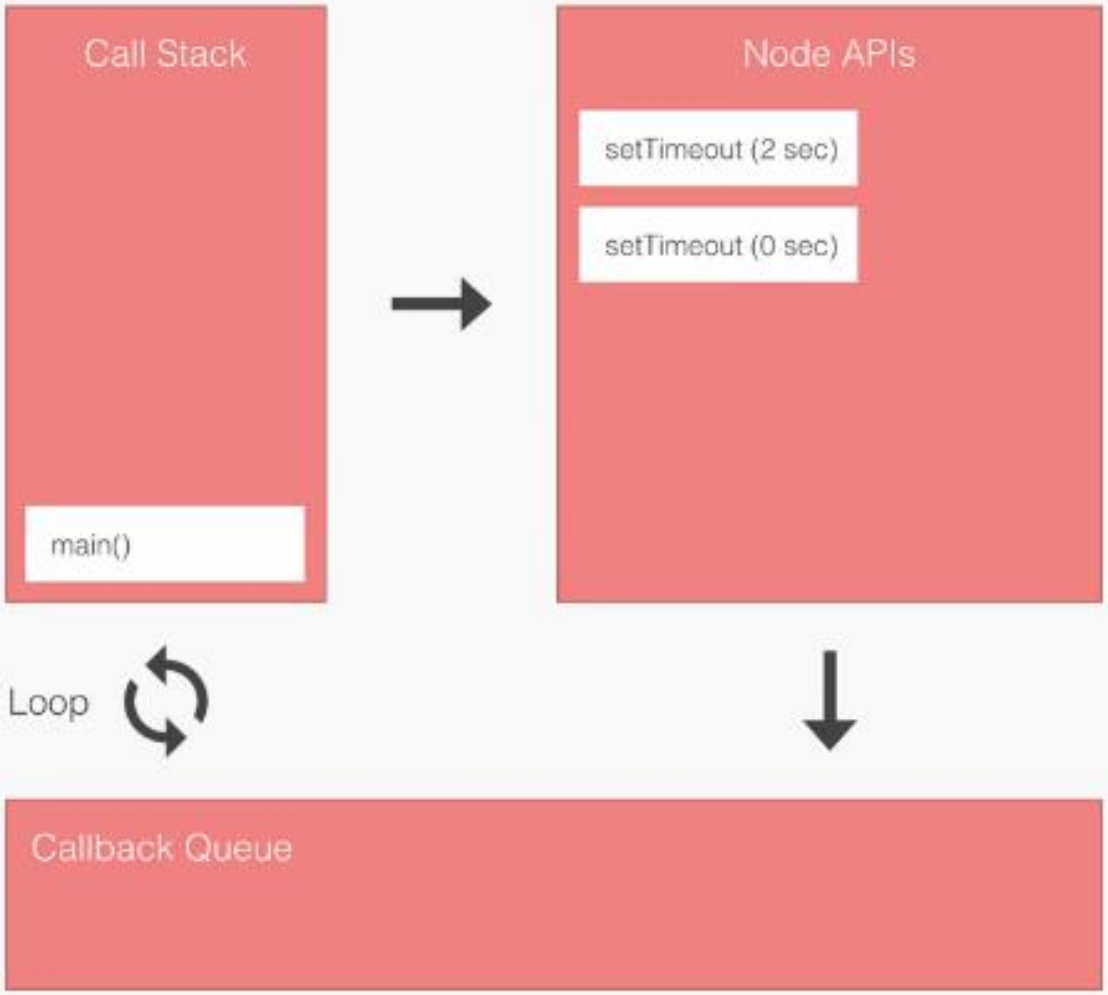
- In the blocking method, user2's data request is not initiated until user1's data is printed to the screen.
- If this was a web server, we would have to start a new thread for every new user. But JavaScript is single-threaded (not really, but it has a single-threaded event loop, which we'll discuss a bit later).
  - This would make JavaScript not very well suited for multi-threaded tasks.
- That's where the non-blocking part comes in

Non-blocking I/O

- Using a non-blocking request, you can initiate a data request for user2 without waiting for the response to the request for user1.
- You can initiate both requests in parallel.
- This non-blocking I/O eliminates the need for multi-threading since the server can handle multiple requests at the same time

The JavaScript event loop

```
1  console.log('Starting app');
2
3  setTimeout(() => {
4    console.log('Inside of callback');
5  }, 2000);
6
7  setTimeout(() => {
8    console.log('Second setTimeout');
9  }, 0);
10
11 console.log('Finishing up');
12
```



The JavaScript event loop (cont.)

1. Push main() onto the call stack.
2. Push console.log() onto the call stack. This then runs right away and gets popped.
3. Push setTimeout(2000) onto the stack. setTimeout(2000) is a Node API. When we call it, we register the event-callback pair. The event will wait 2000 milliseconds, then callback is the function.
4. After registering it in the APIs, setTimeout(2000) gets popped from the call stack.
5. Now the second setTimeout(0) gets registered in the same way. We now have two Node APIs waiting to execute.
6. After waiting for 0 seconds, setTimeout(0) gets moved to the callback queue, and the same thing happens with setTimeout(2000)
7. In the callback queue, the functions wait for the call stack to be empty, because only one statement can execute a time. This is taken care of by the event loop
8. The last console.log() runs, and the main() gets popped from the call stack
9. The event loop sees that the call stack is empty and the callback queue is not empty. So it moves the callbacks (in a first-in-first-out order) to the call stack for execution.

NPM (Node Package manager)

NPM

- These are libraries built by the awesome community which will solve most of your generic problems
- npm has packages you can use in your apps to make your development faster and efficient
- It is used the reserved word require to load packages
- Require is a function, and it accepts a parameter “path” and returns module.exports

Require

1. It loads modules that come bundled with Node.js like file system and HTTP from the Node.js API
2. It loads third-party libraries like Express and Mongoose that you install from npm
3. It lets you require your own files and modularize the project

NPM (Node Package manager) (cont.)

Node modules

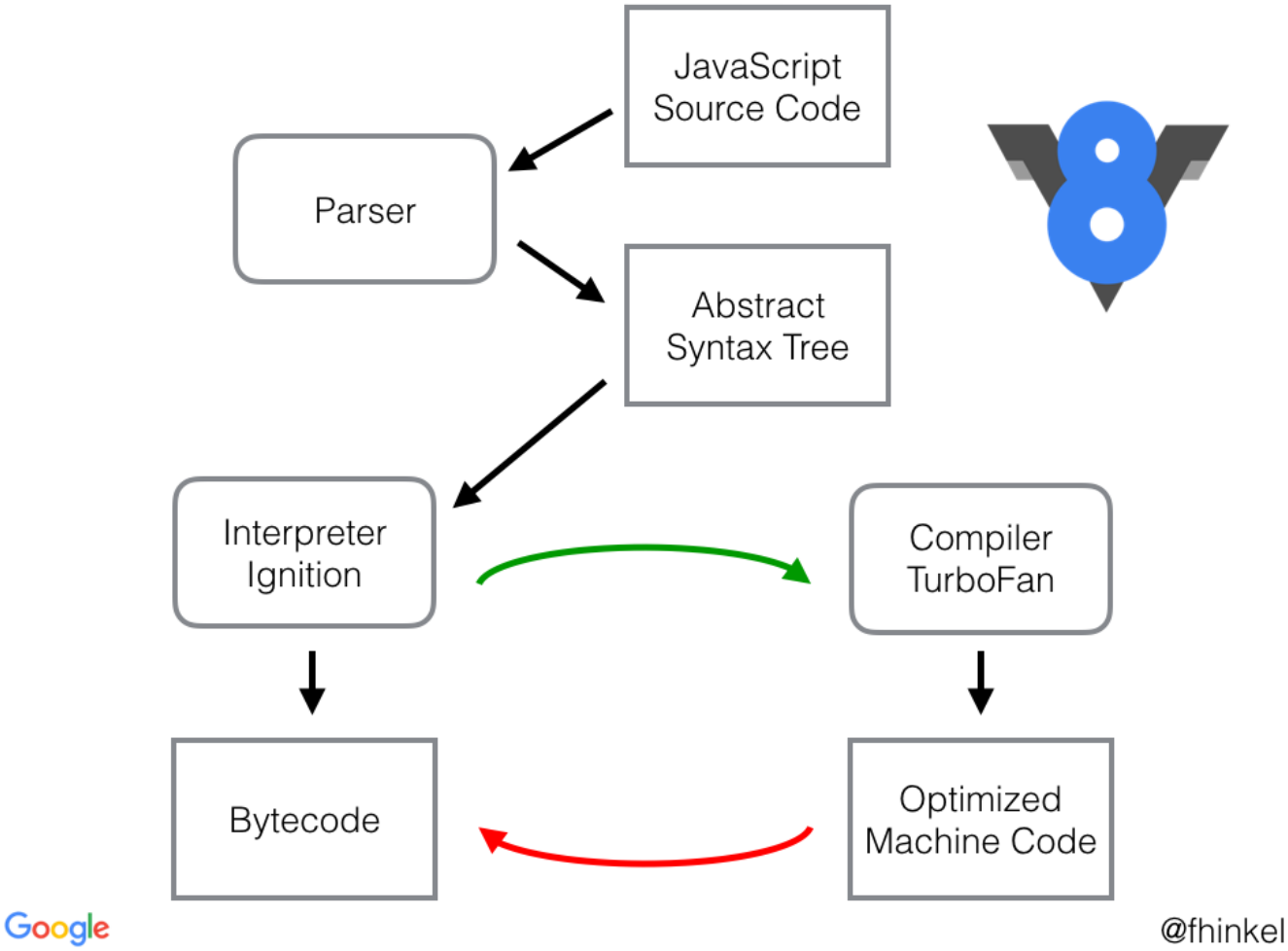
- A Node module is a reusable block of code whose existence does not accidentally impact other code.
- You can write your own modules and use it in various application.
- Node.js has a set of built-in modules which you can use without any further installation.

V8 turbo-charges JavaScript by leveraging C++

- V8 is an open source runtime engine written in C++ developed by Google for its Chrome Browser
- JavaScript -> V8(C++) -> Machine Code
- V8 implements a script called ECMAScript as specified in ECMA-262
  - ECMAScript was created by Ecma International to standardize JavaScript
- V8 can run standalone or can be embedded into any C++ application
  - It has hooks that allow you to write your own C++ code that you can make available to JavaScript
- This essentially lets you add features to JavaScript by embedding V8 into your C++ code so that your C++ code understands more than what the ECMAScript standard otherwise specifies
- There are many different JavaScript runtime engines apart from V8 by Chrome like SpiderMonkey by Mozilla, Chakra by Microsoft, etc. Details of the same can be found on this page.

- A **JavaScript engine** is a computer program that executes JavaScript (JS) code.
- The first JS engines were mere interpreters, but all relevant modern engines utilize just-in-time compilation for improved performance
- JS engines are developed by web browser vendors, and every major browser has one. In a browser, the JS engine runs in concert with the rendering engine via the Document Object Model (DOM).
- The use of JS engines is not limited to browsers.
  - For example, the Chrome V8 engine is a core component of the popular Node.js runtime system.

Chrome's V8 JavaScript Engine



Events

- There are two types of events in Node.
- System Events: C++ core from a library called libuv.
  - For example, finished reading a file
- Custom Events: JavaScript core

Hello World in node.js

Bibliography

1. https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5
2. https://www.monterail.com/blog/nodejs-development-enterprises
3. *Coding with JavaScript for Dummies*. Chris Minnick and Eva Holland. Wiley Brand. John Wiley and Sons, Inc. Hoboken, New Jersey. 2015. ISBN 978-1-119-05607-2 (pbk); ISBN 978-1-119-05605-8 (ePDF); ISBN 978-1-119-05606-5 (ePub)