COMP6080
Web Front-End Programming

Week 2
The Javascript Ecosystem

What even is "Javascript"?

- Is it what Google Chrome has?
- Is it what NodeJS is?
- Is ReactJS different from Javascript?
- What version am I using?

Let's take a step back....

Language V Compiler/Interpreter

Language Definition (Standards)

- Describes how a language should function (rules, syntax)
- Typically defined as a globally recognised standard
- New features to a language mean new versions of the language

Compiler or Interpreter

- A program that takes source code (plain text) from you, and, following language definition rules, produces runnable code for execution
- E.G. Python3, Node, Gcc

Source Code (plain text)

- Programs that you write in .py, .js, .c, .cpp, .java files.
- Fundamentally just plain text (ascii) that compilers interpret based on a language definition

Language V Compiler (Interpreter)

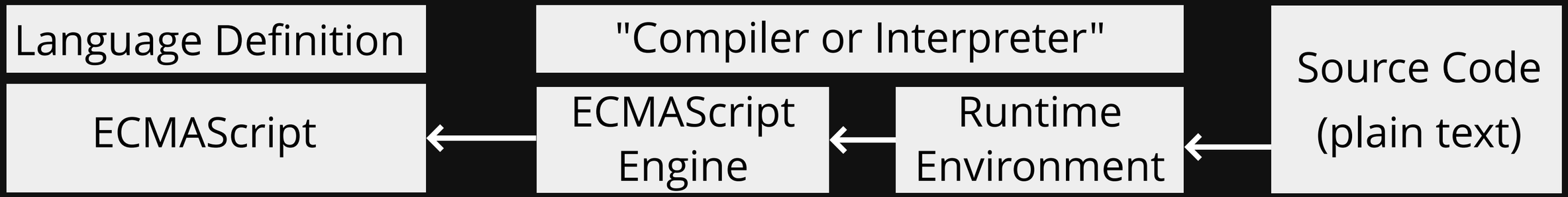
Language Definition (Standards)

Compiler or Interpreter

Source Code (plain text)

Compilers/Interpreters take source code (plain text) and produce executable programs. The way to interpret the source code into executable programs is provided in the language definition.

"Javascript"



- ECMAscript (ES)
- First appeared 1997
- Major releases are:
  - ES5 (ECMAScript 2009)
  - ES6 (ECMAScript 2015)
  - ECMAScript 2016
  - ... etc
  - ECMAScript 2019

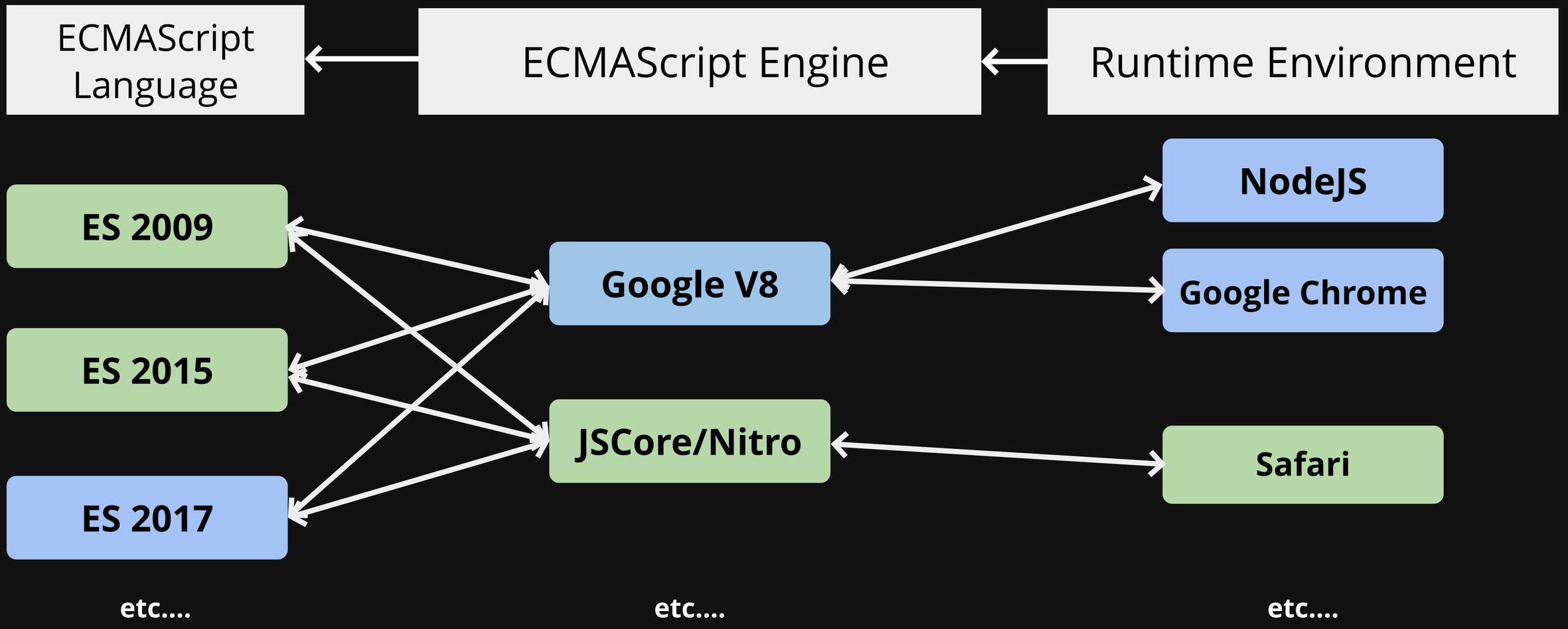
An article about language features

- Javascript compilers or interpreters are known as **runtime environments.**
  - Examples of runtime environments include NodeJS, Google Chrome
- Runtime environments are built on top of **ECMAScript Engines** which are the engines that interpret the ES language and produce runnable code. They do not have I/O nor do they have APIs
  - Examples of engines include V8, Nitro
- Let's chat about this more...

- .js files you write

Javascript refers to a runtime environment that is built on top of an ECMAScript engine

"Javascript"



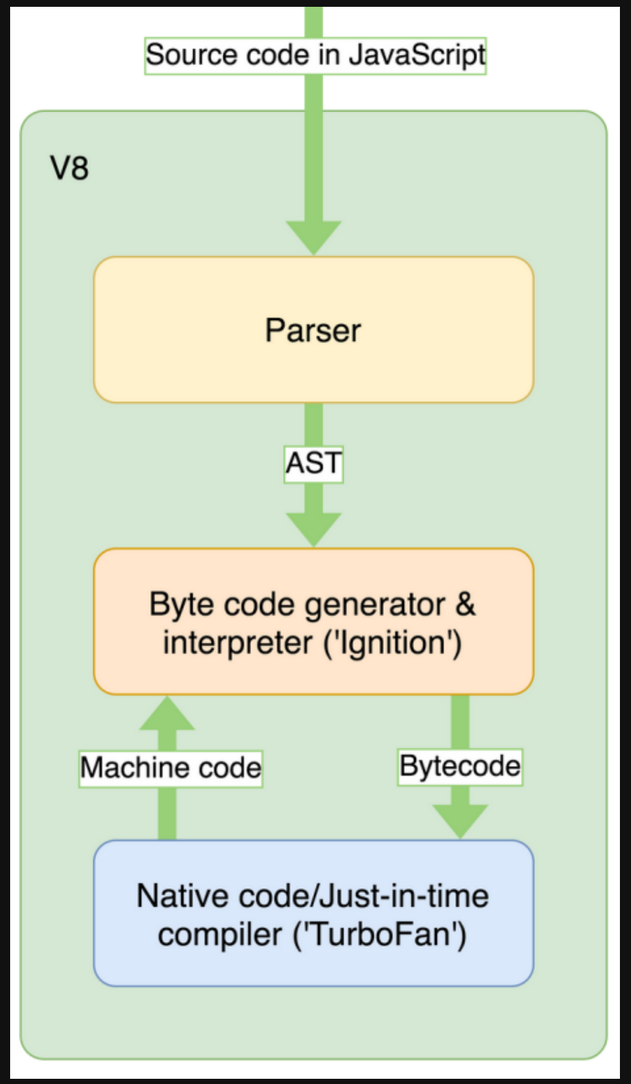
Each version of a runtime environment is built off a particular version of an ECMAScript engine.

Each version of an ECMAScript engine is built to a particular version of ECMAScript

Google V8 Engine

Google's V8 Engine is an open-source Javascript execution engine, a part of the Chromium project. It can run standalone, or can be embedded and extended into any C++ application as a library. V8 is just a compiler+vm toolset, it does not have I/O and APIs built in.

V8 parses, interprets, executes and compiles Javascript code. It is shipped ONLY with the APIs that the ECMAScript Standard specifies.

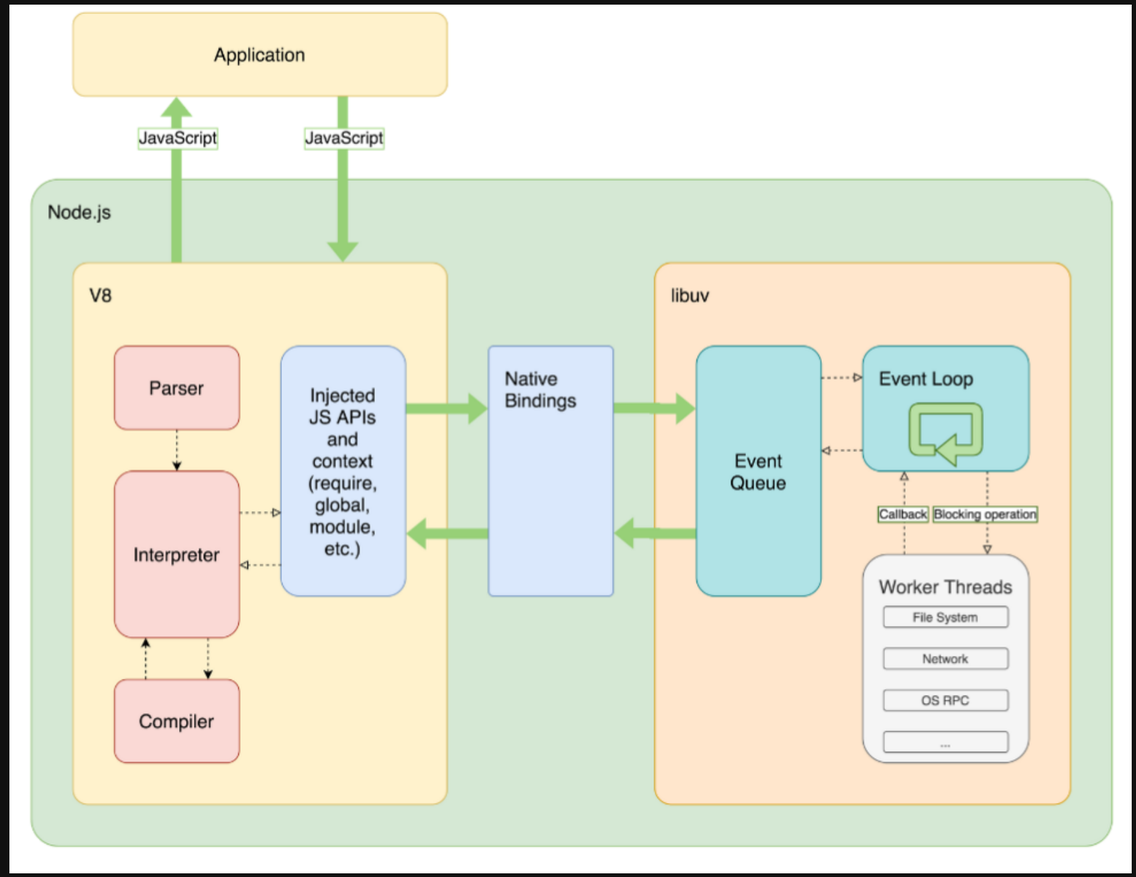


Node.JS

NodeJS is a javascript runtime, with easy to use **command-line** capabilities, that is built on Chrome's V8 Javascript engine.

V8 only provide the core parsing and compiling, but features such as the async event loop/queue are built on top as part of NodeJS.

NodeJS also ships with I/O APIs for network, file system operations, and the concept of modules.

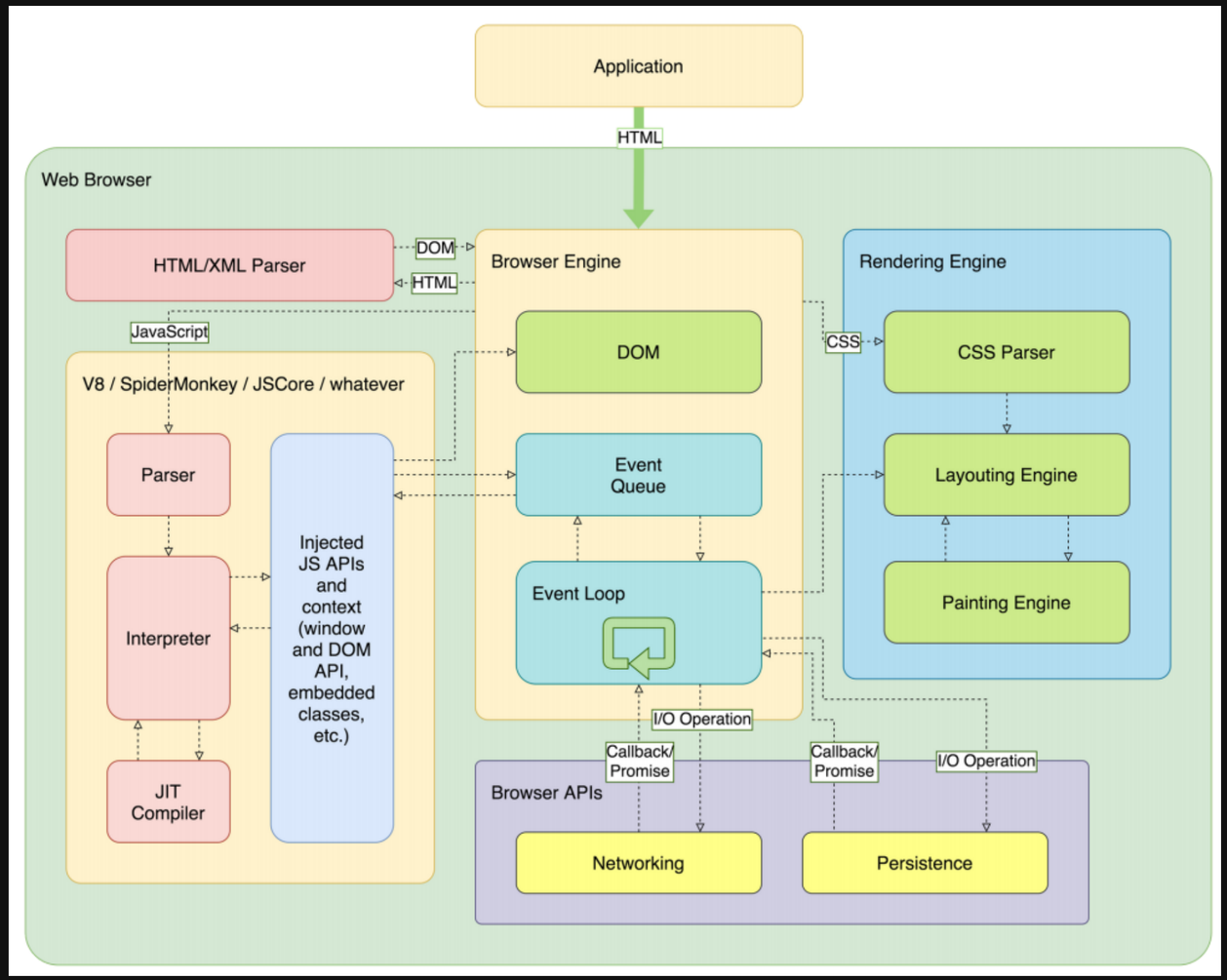


Google Chrome (any web browser)

A web browser is a HTML & CSS document renderer for client-side user interfaces.

The Javascript V8 engine is a small but critical part of a web browser that allows for the execution of Javascript. The primary purpose of Javascript execution in web browsers is to:

- Mutate the DOM
- Make network requests
- Persist data client-side



Feedback