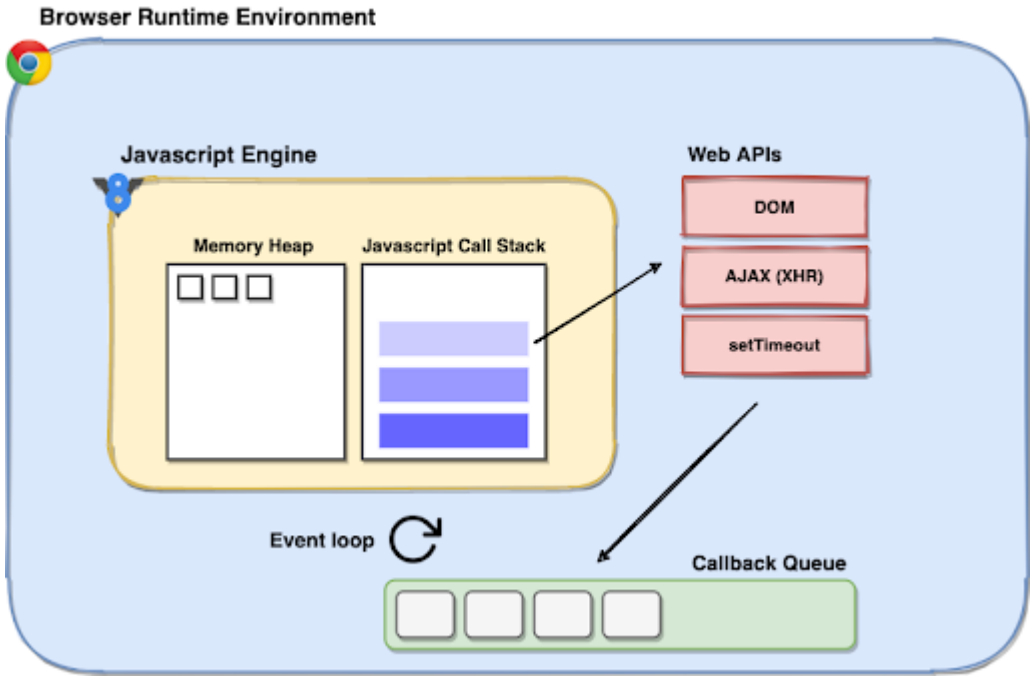
Event Loop - Browser

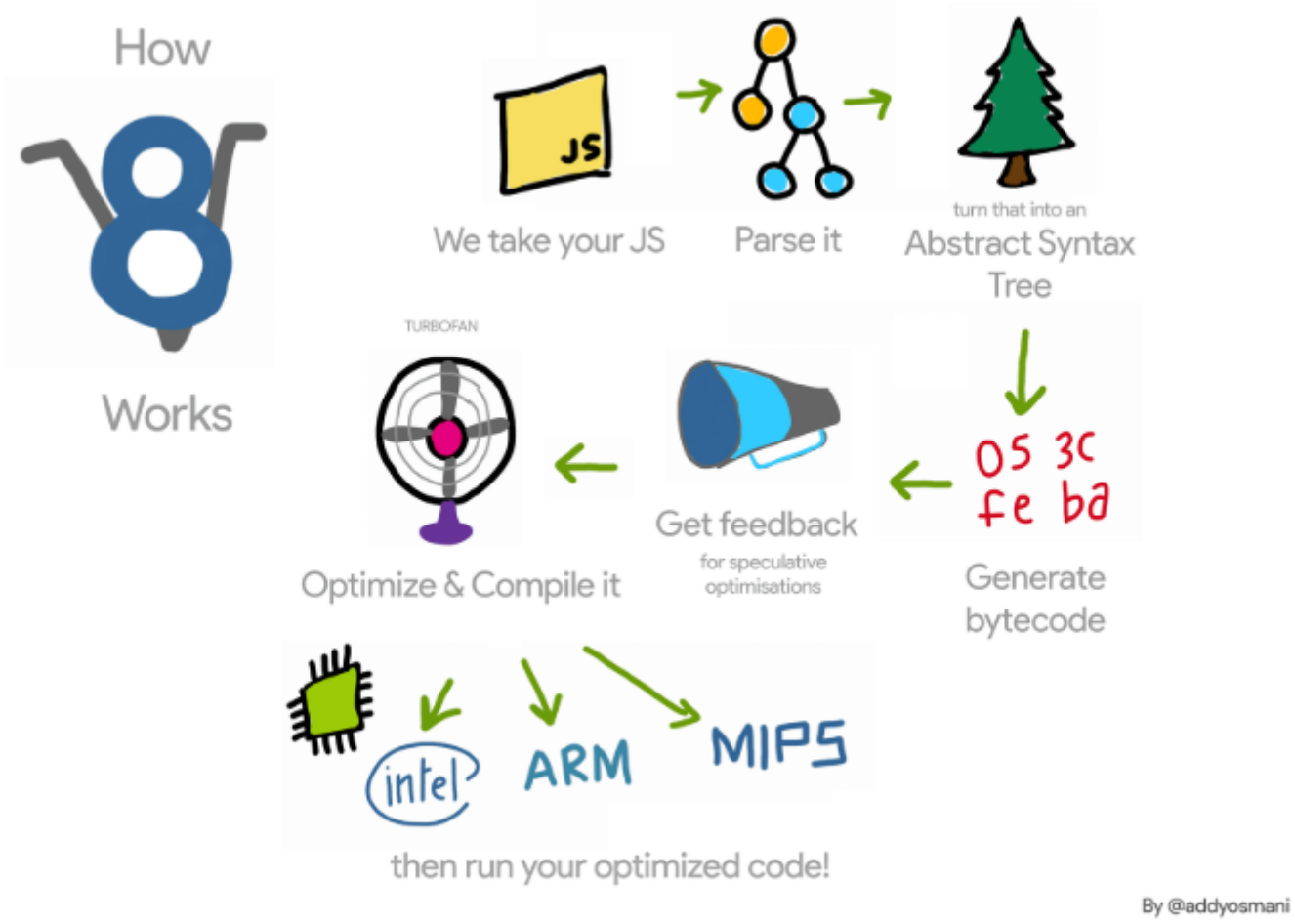
IMPORTANT CONCEPT that every javascript developer shoud know.



> https://scoutapm.com/blog/async-javascript

1. Javascript Engine
2. Event loop components
3. Event loop visualizer

1. Javascript Engine



Là 1 chương trình máy tính thực thi được code javascript

#	Name	Desc
1	Javascript Engine	chương trình máy tính thực thi mã js
2	Ecmascript Engine	chương trình máy tính thực thi mã ECMAScript
3	Compiler	AoT (Ahead of Time), biên dịch tất cả -> thực thi
4	Interpreter	đọc từng dòng và thực thi
5	Just-in-time (JIT) compiler	Compiler + Interpreter (modern browsers). JIT compiler biên dịch mã nguồn thành mã máy trực tiếp trong quá trình thực thi.

> Source: https://hacks.mozilla.org/2017/02/a-crash-course-in-just-in-time-jit-compilers/

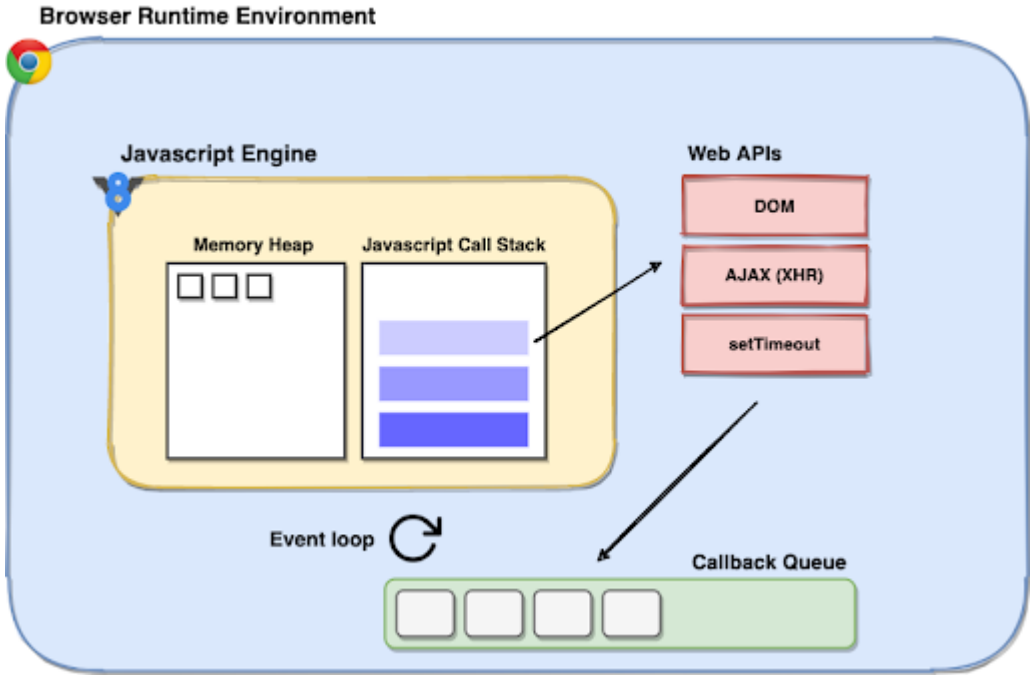
ECMAScript Engines

#	Environment	Engine
1	Chrome	V8 - Google open source
2	NodeJS	V8
3	Microsoft Edge	V8 (v79)

#	Environment	Engine
4	Firefox	Spider Monkey
5	Safari	JavascriptCore

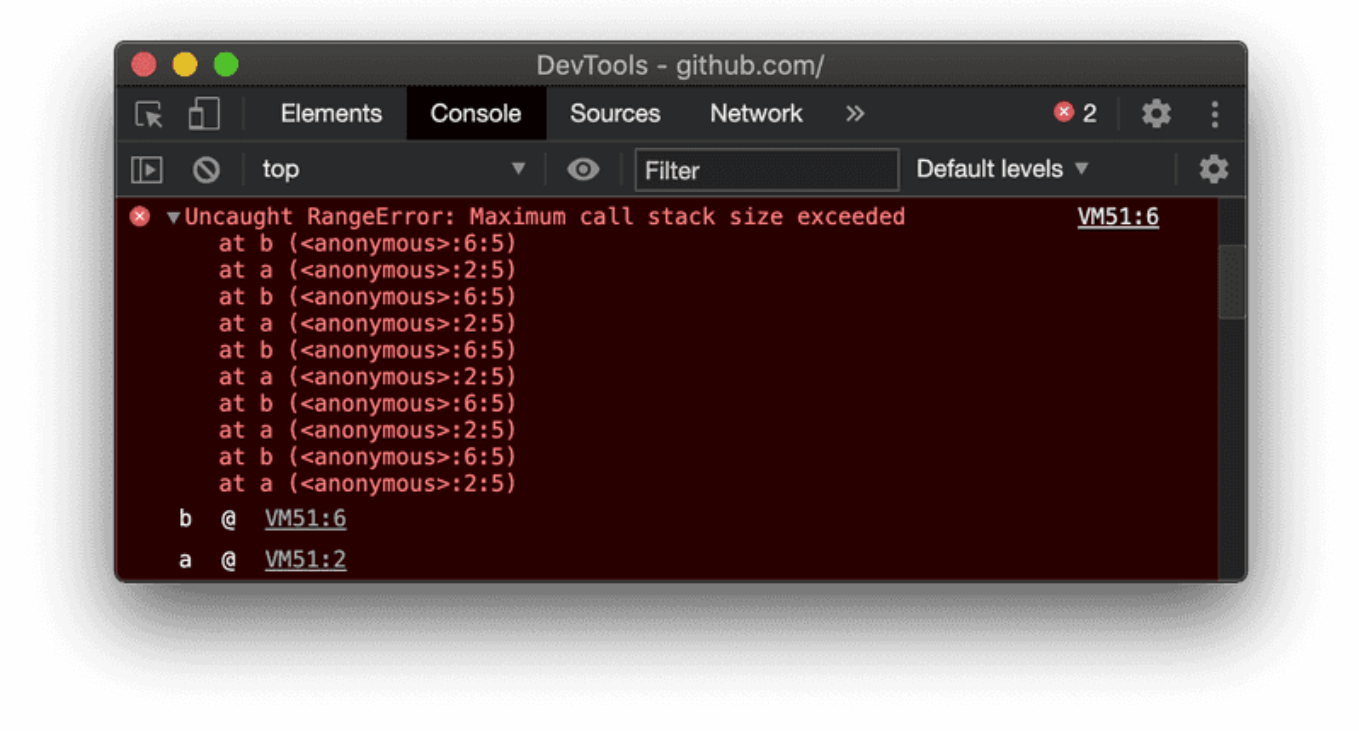
> Source: https://en.wikipedia.org/wiki/List_of_ECMAScript_engines

2. Event loop components



#	Name	Desc
1	Heap	where to store objects and functions read more
2	Call Stack	keep track of the functions that a script calls (last in last out)
3	Web API	APIs of web browsers to help you make AJAX request, DOM manipulation, do things concurrently, ...
4	Callback Queue	run code after the execution of the Web API call has finished (first in first out)
5	Event Loop	run-to-complete, add call from Queue when the call stack is empty.

> Maximum call stack size exceeded - 10k -> 50k. Hay gặp ở đệ quy



> Source: https://felixgerschau.com/javascript-event-loop-call-stack/

> Source: https://2ality.com/2014/04/call-stack-size.html

Callback Queue vs Promise Queue (Macrotask vs Microtask)

- Promise Queue has higher priority than callback queue
- Or we can say: Microtask has higher priority than macrotask.

```
console.log('a');
setTimeout(() => console.log('b'), 0);
new Promise((resolve, reject) => {
 resolve();
})
.then(() => {
    console.log('c');
});
console.log('d');
// a -> d -> c -> b
```

3. Event loop visualizer

> Source: http://latentflip.com/loupe/

> Source: https://www.jsv9000.app/

Tham khảo

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop
- https://scoutapm.com/blog/async-javascript