Javascript

JavaScript was initially created to *“make web pages alive”*.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Statements</h2>

<p>A <b>JavaScript program</b> is a list of <b>statements</b> to be executed by a computer.</p>

<p id="demo"></p>

<script>
var x, y, z;  // Declare 3 variables
x = 5;    // Assign the value 5 to x
y = 6;    // Assign the value 6 to y
z = x + y;  // Assign the sum of x and y to z

document.getElementById("demo").innerHTML =
"The value of z is " + z + ".";
</script>

</body>
</html>
```

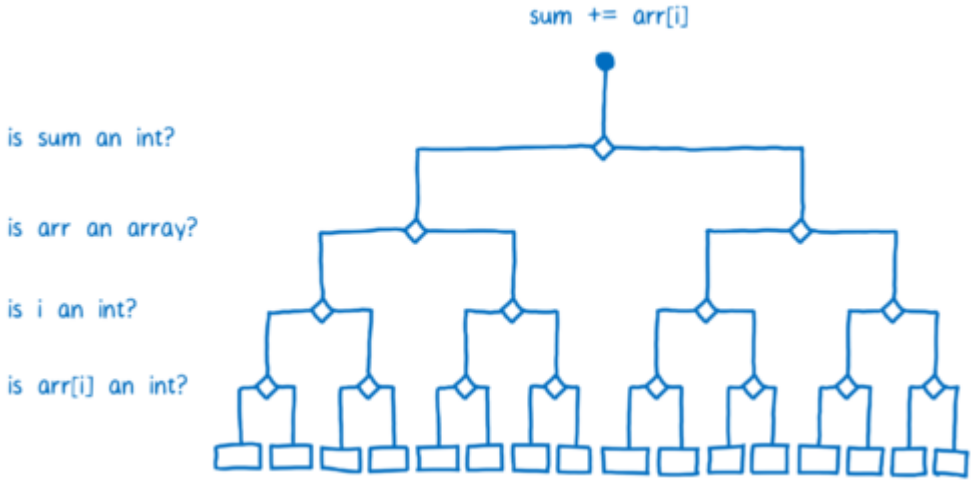
https://www.w3schools.com/js/js_examples.asp

Javascript

Javascript è un linguaggio debolmente tipizzato

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


```
function arraySum(arr) {
  var sum = 0;
  for (var i = 0; i < arr.length; i++) {
    sum += arr[i];
  }
}
```



Javascript

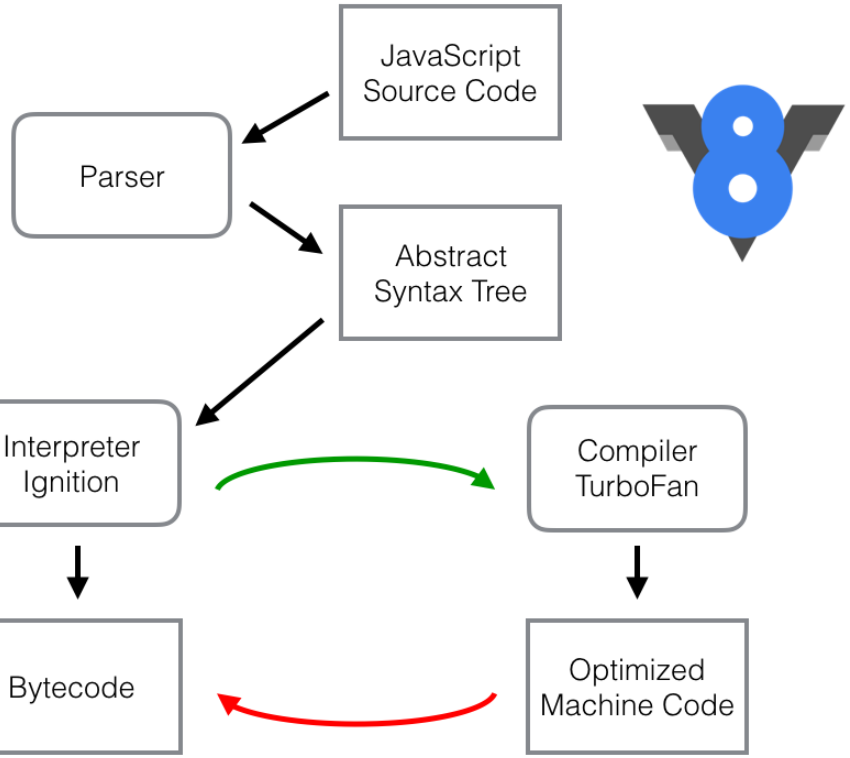
(SpiderMonkey)

(Nitro)

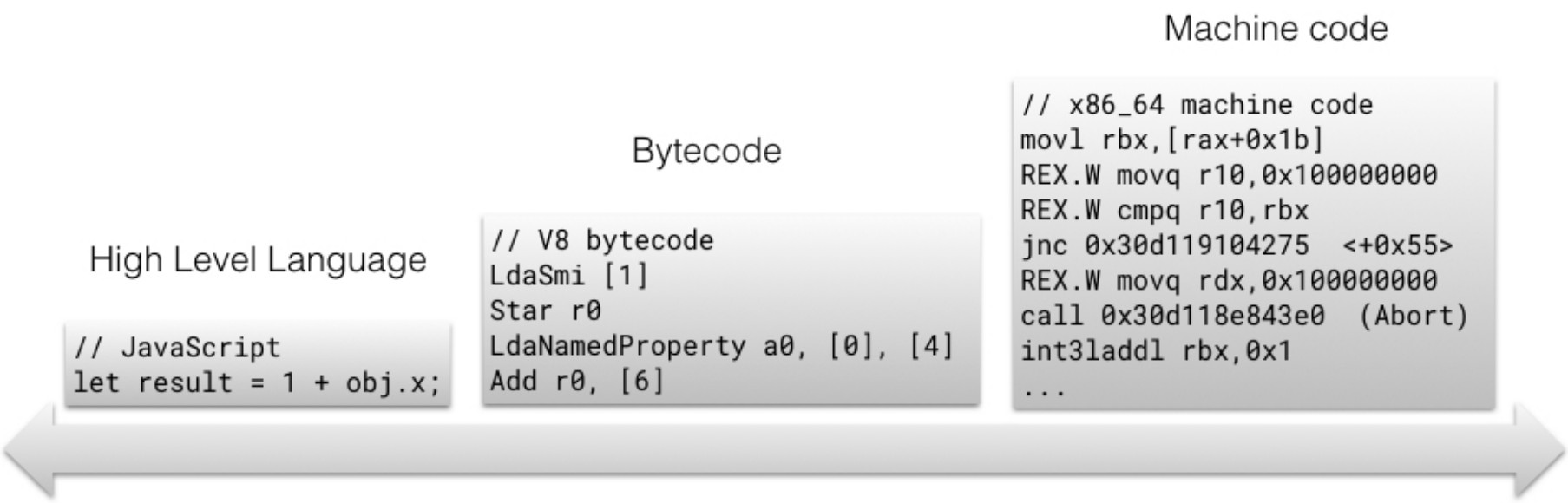
1. The engine (embedded if it's a browser) reads ("parses") the script.

2. Then it converts ("compiles") the script to the machine language.

3. And then the machine code runs, pretty fast.



Javascript

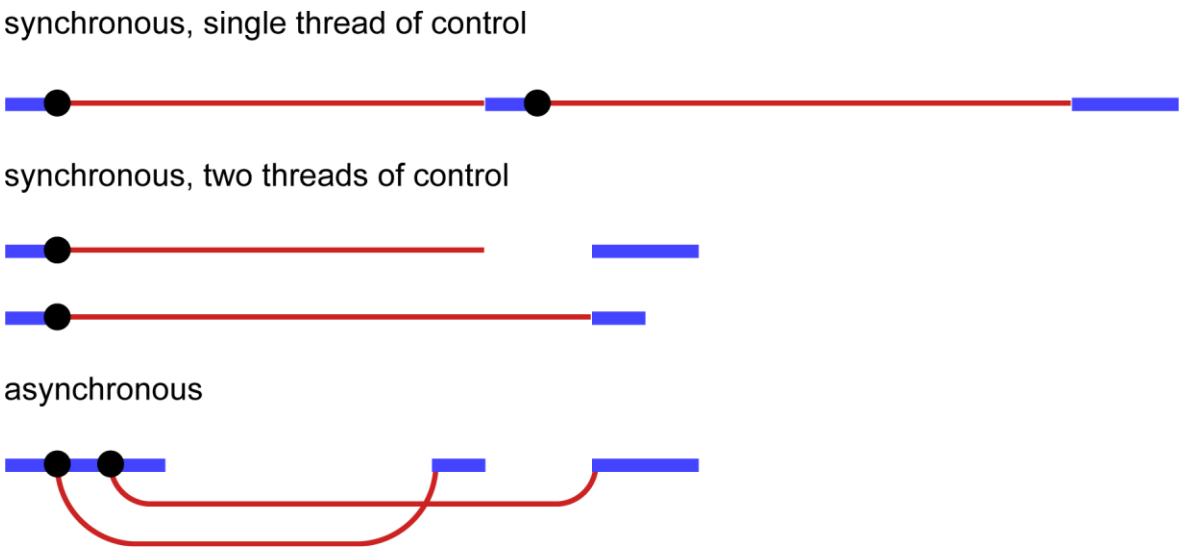


Google

@fhinkel

Javascript

JavaScript is always synchronous and single-threaded. If you're executing a JavaScript block of code on a page then no other JavaScript on that page will currently be executed.



Javascript – Callback and Promise

One approach to asynchronous programming is to make functions that perform a slow action take an extra argument, a *callback function*. The action is started, and when it finishes, the callback function is called with the result.

```
setTimeout(() => console.log("Tick"), 500);
```

A *promise* is an asynchronous action that may complete at some point and produce a value. It is able to notify anyone who is interested when its value is available.

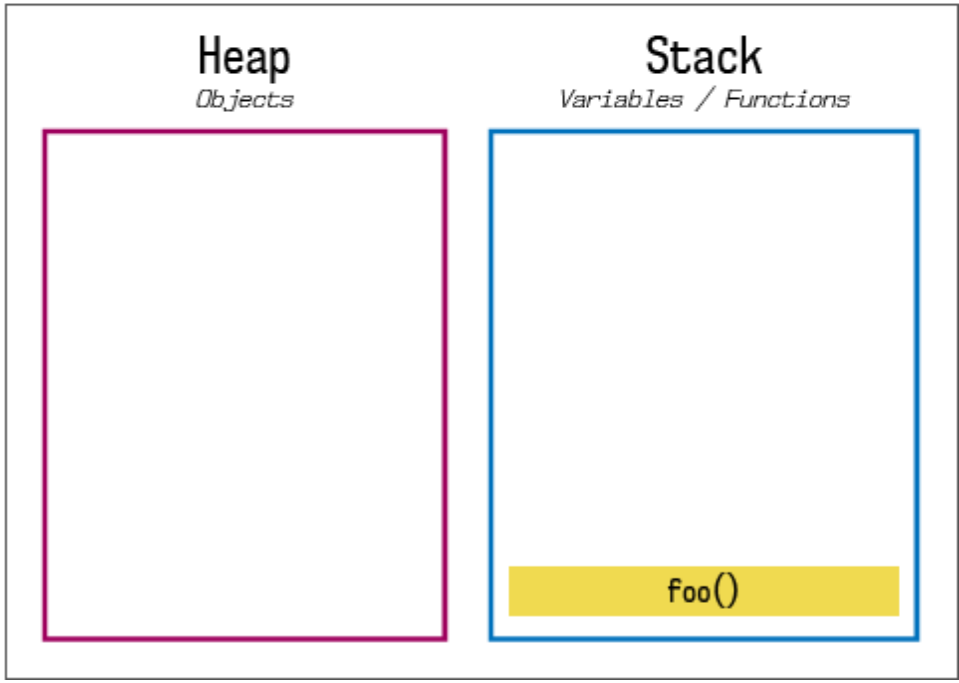
```
let fifteen = Promise.resolve(15);
fifteen.then(value => console.log(`Got ${value}`));
```

Javascript – Callback and Promise

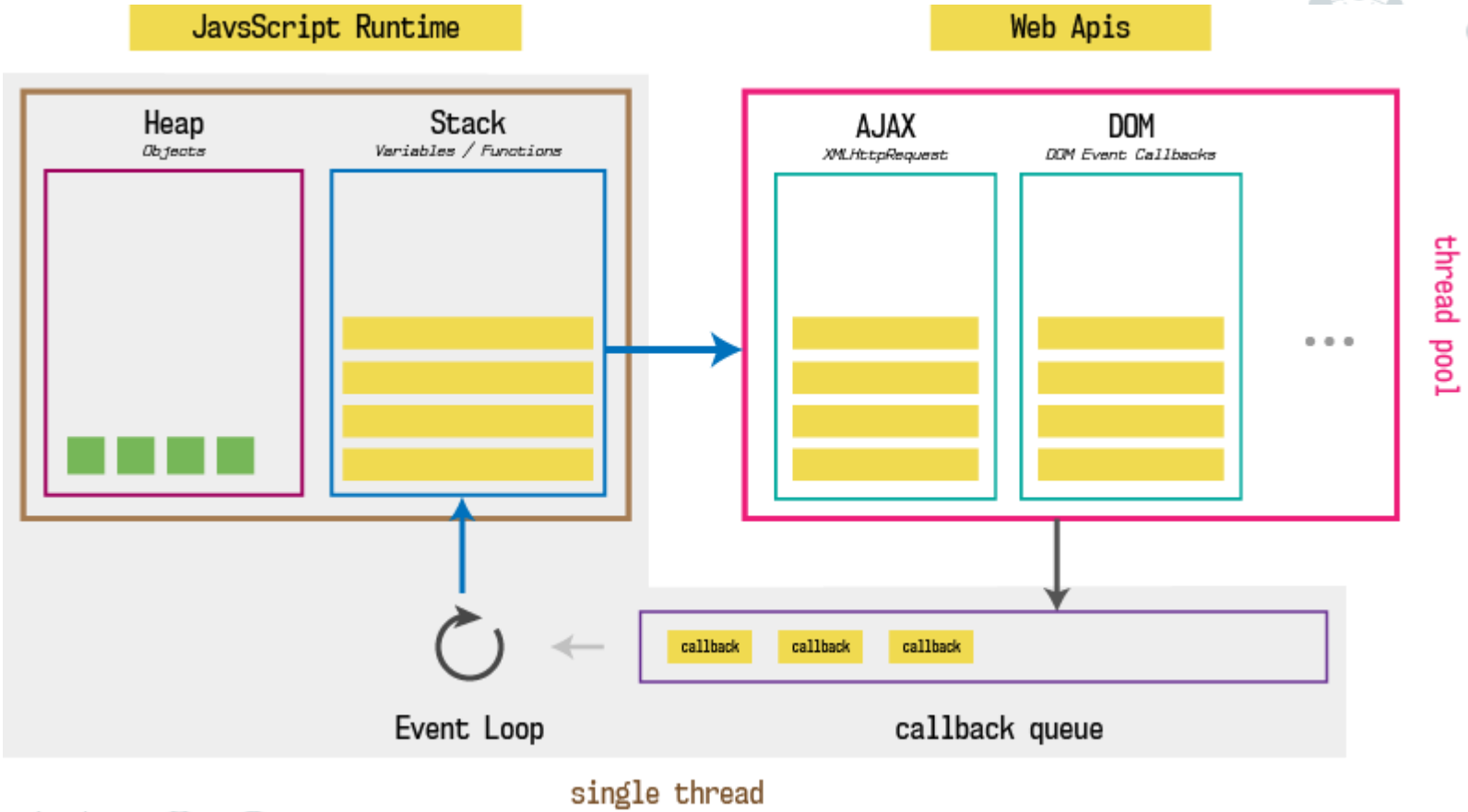
JavsScript Program

```
function baz(){
   console.log('Hello from baz');
}

function bar() {
   baz();
}

function foo() {
   bar();
}

foo();
```

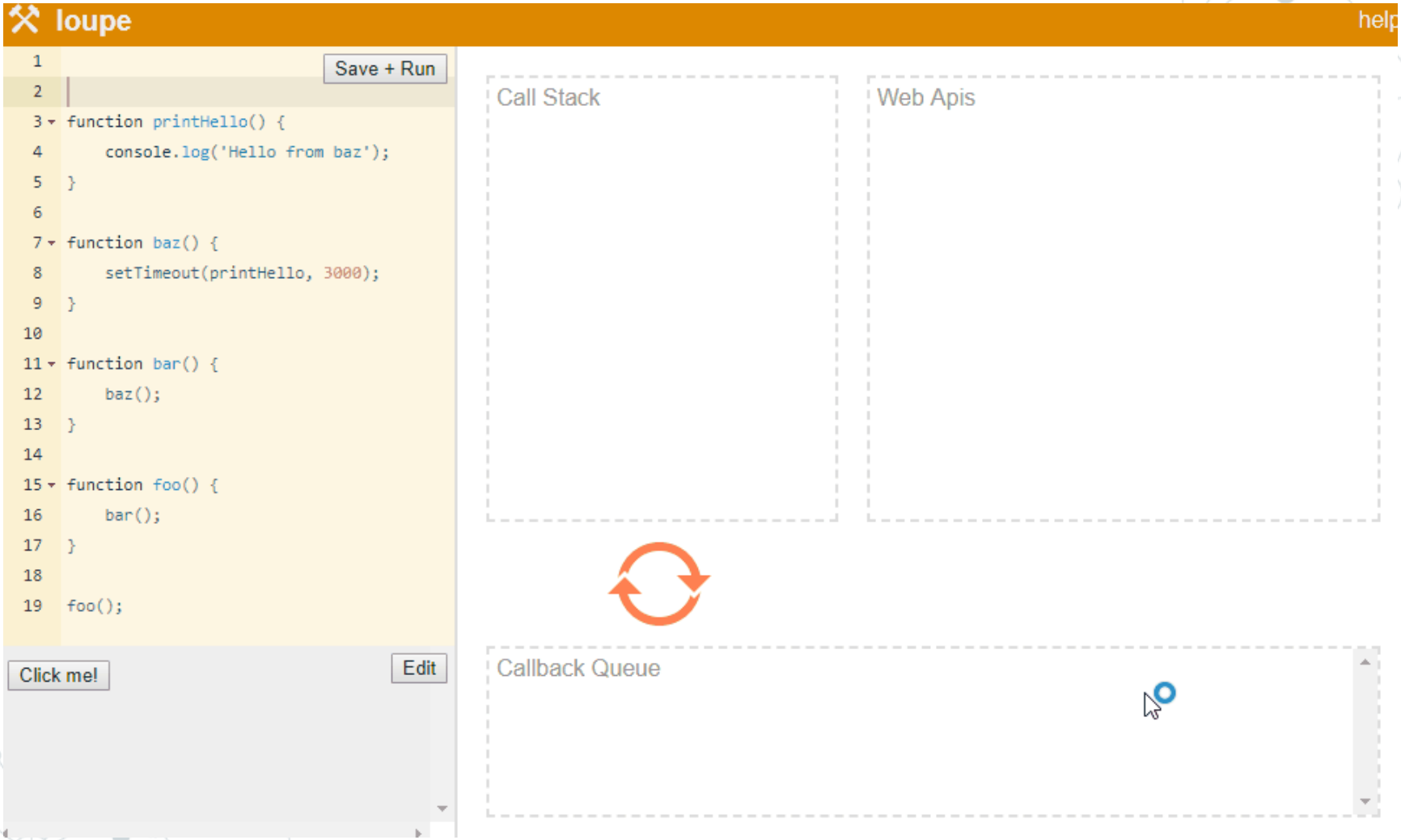
JavsScript Runtime



Javascript – Callback and Promise



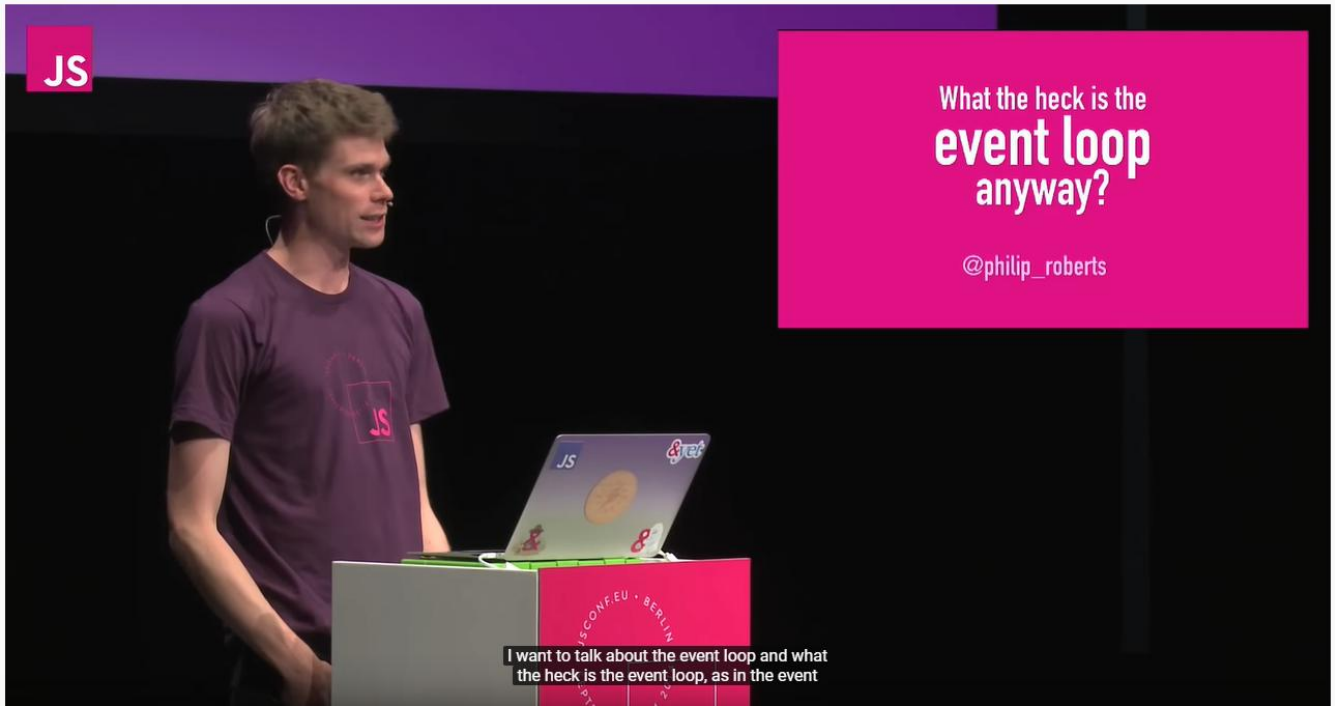
Javascript – Callback and Promise



https://latentflip.com/loupe/

Javascript – Callback and Promise

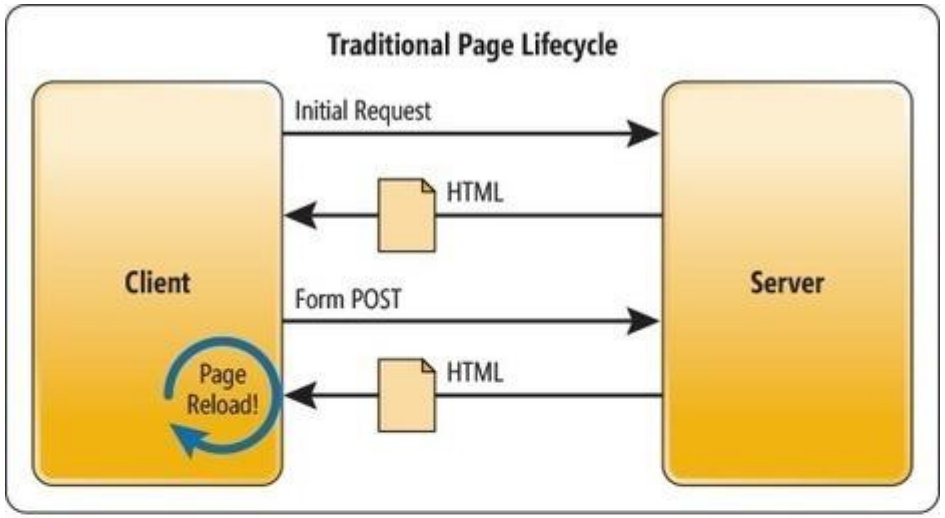
https://www.youtube.com/watch?v=8aGhZQkoFbQ



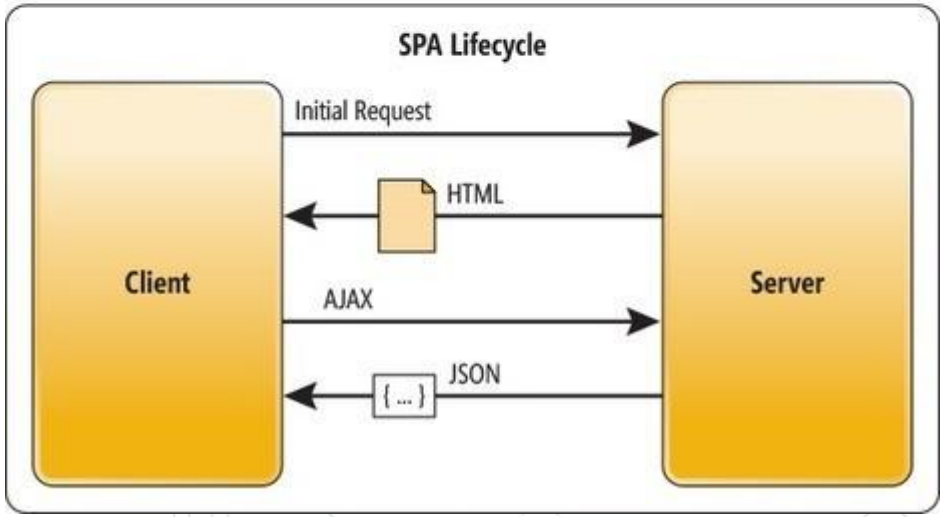
Web e pattern architetturali

Pattern architetturali

Multi-Page Application



Single-Page Application



AJAX – L'inizio delle SPA

https://plnkr.co/edit/KoWor0SZdn1WihEU

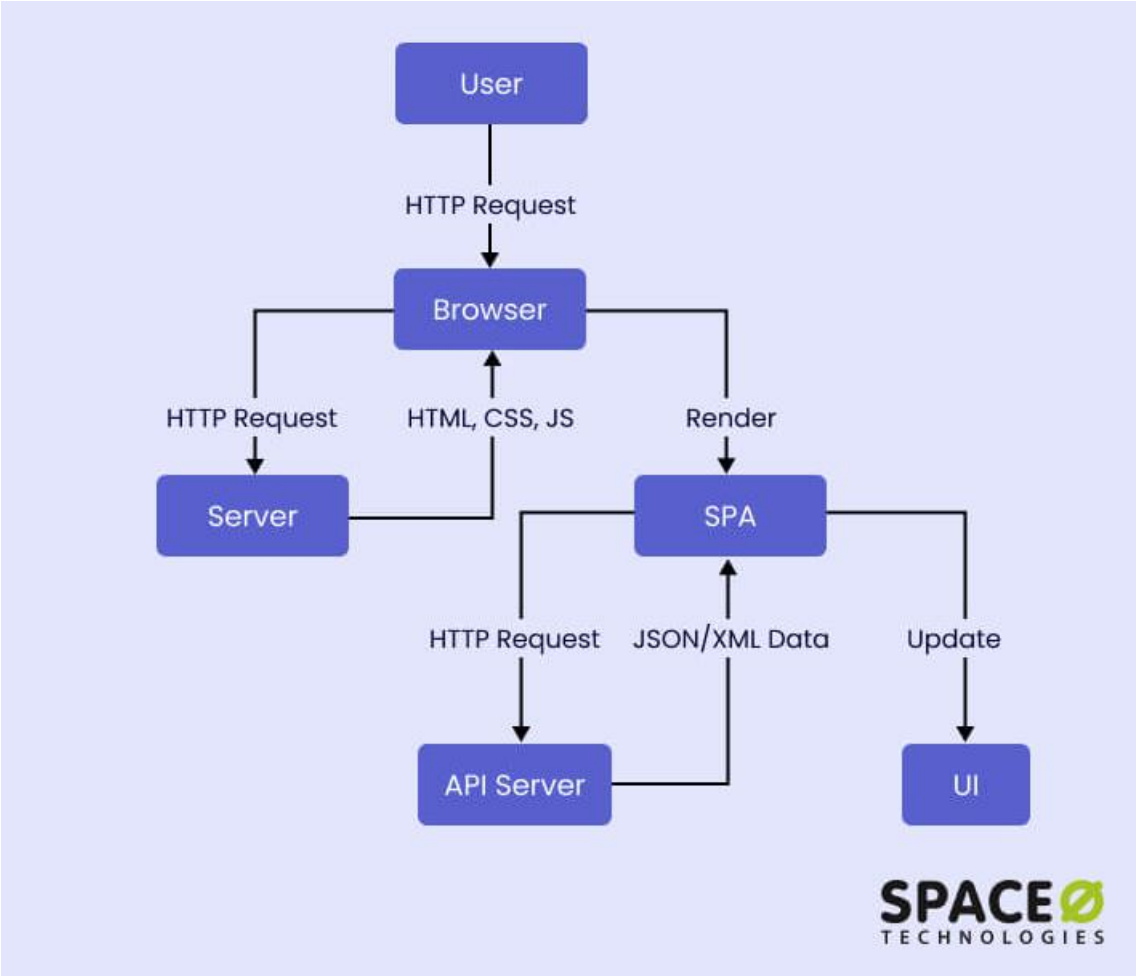
```html
<body>
  <h1>Hello <span id="firstname"></span> <span id="lastname"></span>!</h1>
  <button onclick="reload_user();">Reload!</button>
  <p id="loadingtext">LOADING....</p>
</body>

</html>
```

script.js

```js
document.addEventListener("DOMContentLoaded", () => {
  reload_user();
});

function reload_user(){
    document.getElementById("loadingtext").style.display = 'block';

    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
      const result = JSON.parse(xhttp.responseText);
      document.getElementById("firstname").innerHTML = result.results[0].name.
        first;
      document.getElementById("lastname").innerHTML = result.results[0].name.
        last;
      document.getElementById("loadingtext").style.display = 'none';
  }
  xhttp.open("GET", "https://randomuser.me/api");
  xhttp.send();
}
```

Asynchronous Javascript And XML

Flusso di una SPA

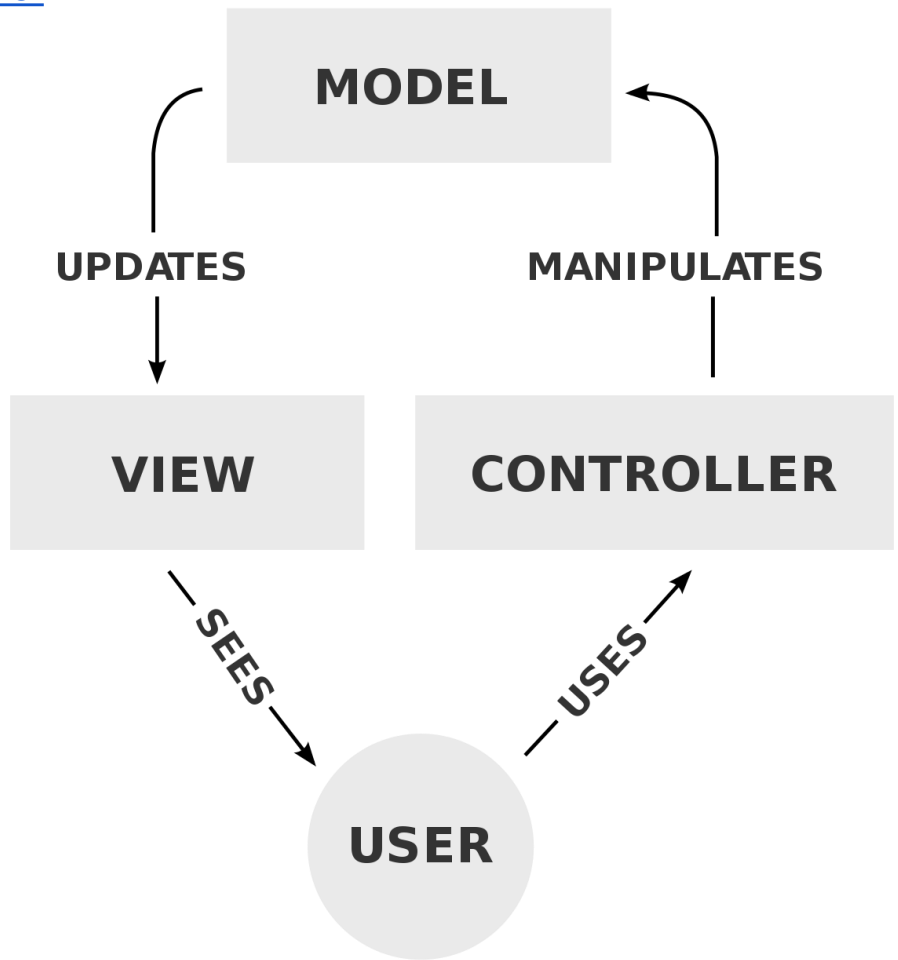


Pattern MVC

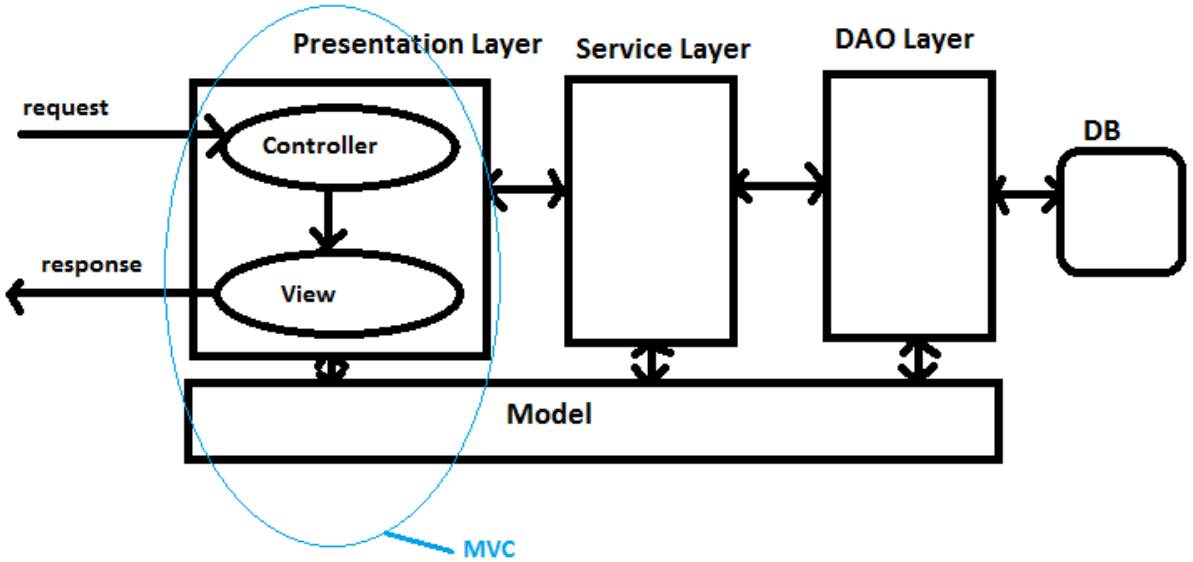
https://it.wikipedia.org/wiki/Model-view-controller

Vantaggi:

1) Disaccoppiare
2) Responsabilità certe
3) View multiple



Architettura generica



The presentation layer is where the data is formatted and presented to the user.

The service layer is wherethe business logic of the application is implemented.

The persistence layer is where the data is simply saved or retrieve.

Il mondo reale è composto da soluzioni ibride

Frontend

Backend

Approcci ibridi
Mix di soluzioni
Evoluzione continua

Trend:
- PWA vs Mobile
- Low Code
- Serverless
- Static site generators
- MicroService

UI Bakery
https://www.youtube.com/watch?v=xbB3MrEi5bo

Less servers for your Angular app
https://www.youtube.com/watch?v=WEYtDYBkalI

Mastering Chaos - A Netflix Guide to Microservices
https://www.youtube.com/watch?v=CZ3wIuvmHeM

Security

Malicious code on library

Un buon inizio da leggere

> If an attacker successfully injects any code at all, it's pretty much game over

XSS is too small scale, and really well protected against.

Chrome Extensions are too locked down.

Lucky for me, we live in an age where people install npm packages like they're popping pain killers.

I was excited at this point — I had a compelling package — but I didn't want to wait around while people slowly discovered it and spread the word. So I set about making PRs to existing packages that added my colourful package to their dependencies.

I've now made several hundred PRs (various user accounts, no, none of them as "David Gilbertson") to various frontend packages and their dependencies. "Hey, I've fixed issue x and also added some logging."

Look ma, I'm contributing to open source!

There are a *lot* of sensible people out there that tell me they don't want a new dependency, but that was to be expected, it's a numbers game.

https://medium.com/hackernoon/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5

Un caso famoso

Malicious code found in npm package event-stream downloaded 8 million times in the past 2.5 months

NOVEMBER 26, 2018 | IN VULNERABILITIES | BY DANNY GRANDER

A widely used npm package, `event-stream` , has been found to contain a malicious package named `flatmap-stream` . This was disclosed via a GitHub issue raised against the source repo.

https://snyk.io/blog/malicious-code-found-in-npm-package-event-stream/

`flatmap-stream` is a malicious package which was used in order to steal bitcoins from wallets. The malicious code was able to check if the `copay-dash` package was installed, and then attempt to steal the bitcoins stored in it. It was distributed by hijacking the popular `event-stream` package and adding `flatmap-stream` as a dependency.

https://snyk.io/vuln/SNYK-JS-FLATMAPSTREAM-72637