ES2015/ES6

Basic of modern JavaScript

Alberto Giovanelli

digital GARAGE

TECHNOLOGIES

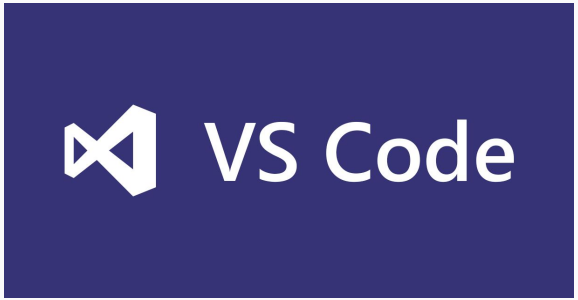
CONTENT

PRESENTATION

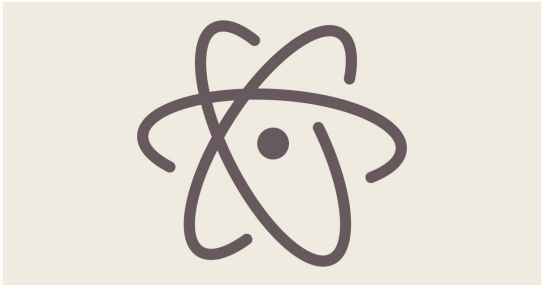
DYNAMIC
EFFECTS/PROGRAMMING

EDITORS

Webstorm



Visual Studio Code



Atom

HTML

- The HTML allow us to construct the visible part of a website.
- HTML is NOT a programming language, its a markup language, which means its purpose is to give structure to the content of the website.
- It is a series of nested tags that contain all the website information (like texts, images and videos). Here is an example of tags:
- `<title>This is a title</title>`

HTML : basic rules

- There are different HTML elements for different types of information and behaviour
- The information is stored in a tree-like structure (nodes that contain nodes inside) called DOM (Document Object Model).
- It gives the document some semantic structure (pe. this is a title, this is a section, this is a form).
- It must not contain information related to how it should be displayed (that information belongs to the CSS), so no color information, font size, position, etc.

HTML : syntax example

```
<div id="main">
    <!-- this is a comment -->
    This is text without a tag.
    <button class="mini">press me</button>
</div>
```

Alberto Giovanelli

HTML : main tags

Although there are lots of tags in the HTML specification, 99% of the webs use a subset of HTML tags with less that 10 tags, the most important are:

- : a container, usually represents a rectangular area with information inside.
- <img/>: an image
- <a>: a clickable link to go to another URL
- <p>: a text paragraph
- <h1>: a title (h2,h3,h4 are titles of less importance)
- <input>: a widget to let the user introduce information
- <style>: to insert CSS rules
- <script>: to execute Javascript
- : a null tag (doesn't do anything)

Alberto Giovanelli

HTML : tagging correctly

Try to avoid doing this:

```
<div>
Title

Here is some content
Here is more content
</div>
```

Do this instead

```
<div>
    <h1>Title</h1>
    <p>Here is content.</p>
    <p>Here is more content</p>
</div>
```

HTML : references

The 25 most used tags : a list with information of the more common tags.

HTML5 best practices : some tips for starters.

HTML reference

Alberto Giovanelli

CSS

Allows to specify how to present (render) the document info stored in the HTML.

Allows to controls all the aspects of the visualization and some other features:

- Colors: content, background, borders
- Margins: interior margin, exterior margin
- Position: where to put it
- Sizes: width, height
- Behaviour: changes on mouse over

CSS example

```
* {
    color: blue; /*a comment */
    margin: 10px;
    font: 14px Tahoma;
}
```

This will change all the tags in my web (‘*‘ means all) to look blue with font Tahoma with 14px, and leaving a margin of 10px around.

CSS : How to add it

There are three ways to add CSS rules to your website:

- Inserting the code inside a style tag

```
<style>
    p { color: blue }
</style>
```

- Referencing an external CSS file

```
<link href="style.css" rel="stylesheet" />
```

- Using the attribute style on a tag

```
<p style="color: blue; margin: 10px ">
```

CSS Selectors

The main selectors are :

- tag name: just the name of the tag
  - `p { ... }  //affects to all <p> tags`
- dot (.): affects to the tags of that class
  - `p.highlight { ... } //affects all <p> tags with class="highlight"`
- sharp character (#): specifies tags with that id.
  - `p#intro { ... } //affects to the <p> tag with the id="intro"`
- two dots (:): behaviour states (mouse on top)
  - `p:hover { ... } //affects to <p> tags with the mouse over`

CSS Selectors

You can also specify tags by its context, for example: tags that are inside of tags matching a selector.

Just separate the selectos by a space:

```
div#main p.intro { ... }
```

This will affect to the p tags of class intro that are inside the tag div of id main

CSS Selectors

If you want to select only elements that are direct child of one element (not that have an ancestor with that rule), use the `>` character:

```
ul.menu > li { ... }
```

Finally, if you want to use the same CSS actions to different rules, you can use the comma `,` character:

```
div, p { … }
```

CSS future reading

Understanding the Box Model : a good explanation of how to position the information on your document.

CSS Selectors : the CSS selectors specification page.

CSS Reference

JAVASCRIPT

- Allows to give some interactivity to the elements on the web.
- You can change the content of the HTML or the CSS applied to an element.
- You can even send or retrieve information from the internet to update the content of the web without reloading the page.
- Today, JavaScript can be used in different places :
    - Client-side : JavaScript was traditionally only used in the browser.
    - Server-side : Thank to node.js, we can use JavaScript on the server as well.

JAVASCRIPT : insert code

There is three ways to execute javascript code in a website:

- Embed the code in the HTML using the <script> tag.

```
<script> /* some code */ </script>
```

- Include a Javascript file using the <script> tag:

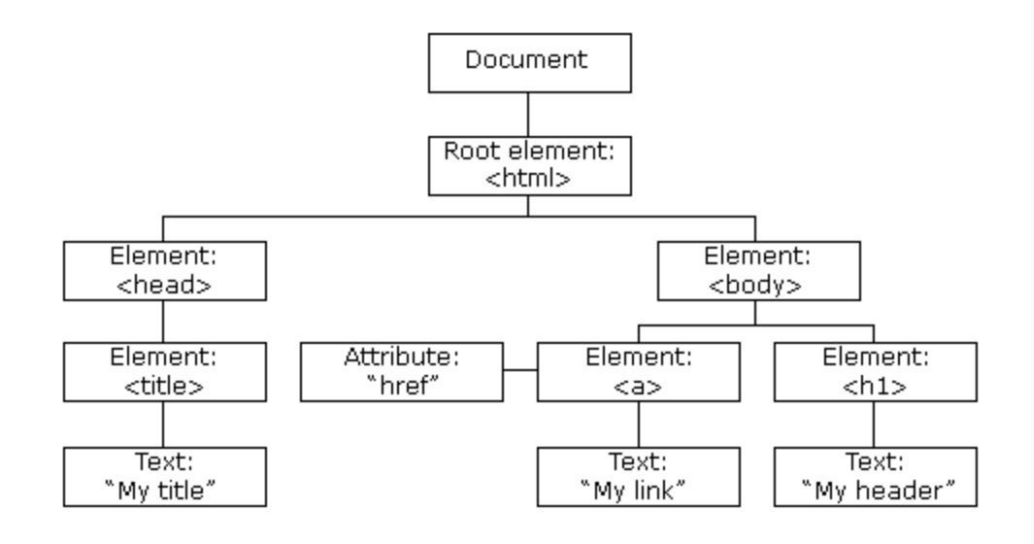
```
<script src="file.js" />
```

- Embed the code on an event inside a tag:

```
<button onclick="javascript:  /*code*/">press me</button>
```

JAVASCRIPT : DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- https://www.w3schools.com/js/



digital GARAGE

JAVASCRIPT : retrieving elements

You can get elements from the DOM (HTML tree) using different approaches.

- Crawling the HTML tree (starting from the body, and traversing its children)
- Using a selector (like in CSS)
- Attaching events listeners (calling functions when some actions are performed)

JAVASCRIPT : Crawling the DOM

From javascript you have different variables that you can access to get information about the website:

- document: the DOM information (HTML)
- window: the browser window

The document variable allows to crawl the tree:

```
document.body.children[0] // returns the first node
inside body tag
```

Alberto Giovanelli

JAVASCRIPT : Using selector

You can retrieve elements using selectors:

```
var nodes = document.querySelectorAll("p.intro");
```

will return an array with all <p class="intro"> nodes in the web.

Or if we have already a node and we want to search inside:

```
var node = mynode.querySelectorAll("p.intro")
```

JAVASCRIPT : Modify node

From JS you can change the attributes

```
mynode.id = "intro"; //sets an id
mynode.className = "important"; //adds a class
mynode.classList.add("good"); //to add to the current ones
```

the content

```
mynode.innerHTML = "<p>text to show</p>"; //change content
```

the style

```
mynode.style.color = "red"; //change css properties
```

or the behaviour of a node

```
mynode.addEventListener("click", function(e) {
    //do something
});
```

JAVASCRIPT : create nodes

Create elements:

```
var element = document.createElement( "div");
```

And attach them to the DOM:

```
document.querySelector( "#main").appendChild( element);
```

Or remove it from its parent:

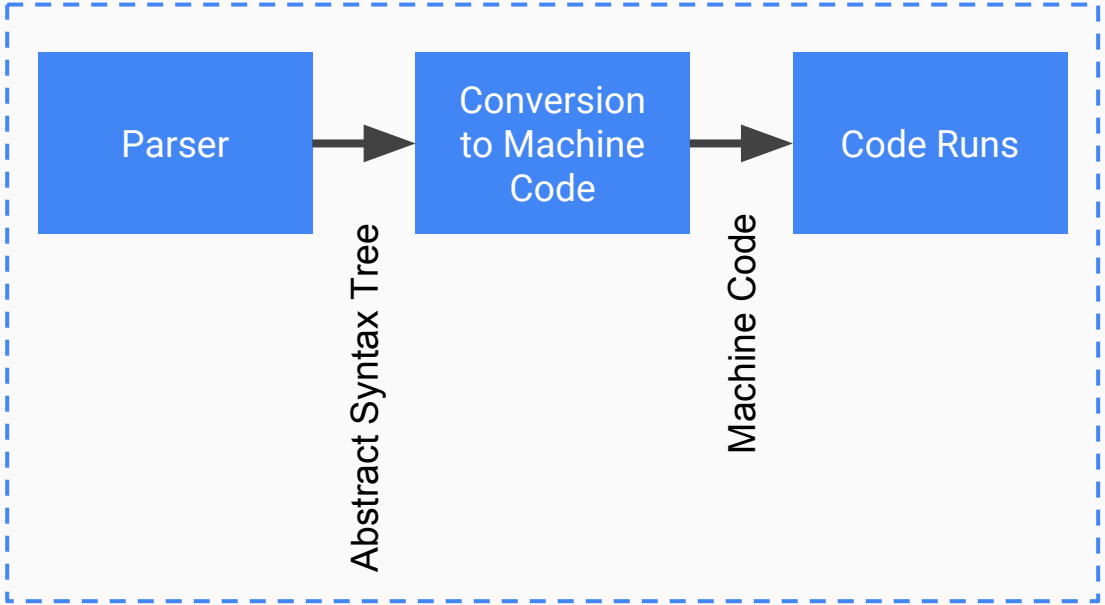
```
var element = document.querySelector( "foo");

element.parentNode.removeChild( element );
```

WHAT HAPPENS TO OUR CODE?

OUR CODE

```
function calculateAge(yearOfBirth) {
    return 2016 - yearOfBirth;
}

var johnsAge = calculateAge(1990);

function yearsUntilRetirement(name, yearOfBirth) {
    var age = calculateAge(yearOfBirth);
    var retirement = 65 - age;
    if (retirement >= 0) {
        console.log(name + ' retires in ' + retirement + ' years.');
    } else {
        console.log(name + ' is already retired.');
    }
}

yearsUntilRetirement('John', 1990);
```

JAVASCRIPT ENGINE



Alberto Giovanelli

digital GARAGE

EXAMPLE OF A WEBSITE

HTML in index.html

```
<link href="style.css" rel="stylesheet"/>
<h1>Welcome</h1>
<p>
     <button>Click me</button>
</p>
<script src="code.js"/>
```

CSS in style.css

```
h1 { color: #333; }
button {
     border: 2px solid #AAA;
     background-color: #555;
}
```

Javascript in code.js

```
//fetch the button from the DOM
var button = document.querySelector("button");

//attach and event when the user clicks it
button.addEventListener("click", myfunction);

//create the function that will be called when
the button is pressed
function myfunction()
{
     //this shows a popup window
     alert("button clicked!");
}
```

Alberto Giovanelli

digital GARAGE

INTRODUCTION TO ES6 - DICTIONARY

- ECMA International
  - An international non-profit standards organization for information and comunication systems. It acquired its current name in 1994, when the Eurpean Computer Manufacturing Association (ECMA) changed its name to reflect the organization's global reach and activities.

- ECMAScript(ES)
  - scripting-language specification standardized by ECMA International in ECMA-262 and ISO/IEC 16262. Well-known implementations of the language, such as Javascript, JScript and ActionScript have come into wide use for client-side scripting on the Web.

- ES2015/ES6
  - the newest version of ECMAScript

JAVASCRIPT : Primitive Data Types

- JavaScript is un **untyped** language, but...what does it mean?
    - **untyped** means not type declaration.

VARIABLES

VAR

```
var obj = {par:3};
obj = 4; //Fine
```

CONST

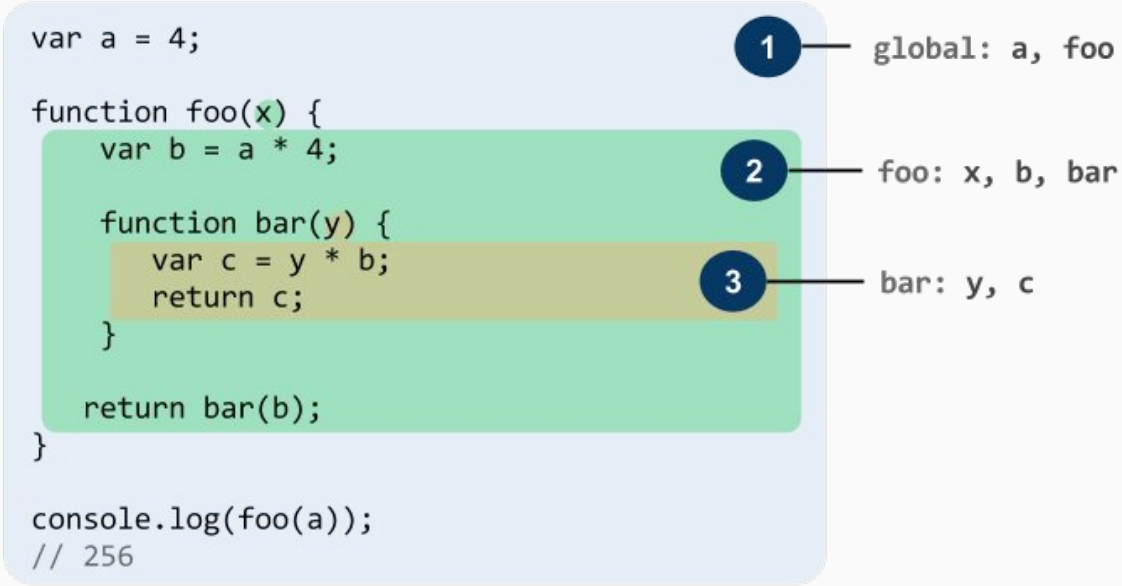
```
const obj = {par : 3};
obj = 4; //TypeError
```

LET

```
let obj = {par : 3};
obj = 4; //Fine
```

const - makes variables Immutable.

DECLARATION AND SCOPING



In Javascript there are 2 type of scope:

1) Local Scope

2) Global Scope

DECLARATION AND SCOPING

```
if (true) {
    var x = 3;
}
console.log(x); //3

if (true) {
    let x = 3;
}
console.log(x); //ReferenceError
```

- Scopes determines the accessibility of these variables.
- Variables defined inside a function are not accessible from outside the function.
- var : is function scope, each function create a new scope.
- ES6 : let & const are block scope.

OPERATORS

Operator	Description	Example	Result
+	Addition	3 + 11	14
-	Subtraction	9 – 4	5
*	Multiplication	3 * 4	12
/	Division	21 / 7	3
%	Modulus (*remainder after division*)	21 % 8	5
++	Increment	a = 5; ++a	(a *equals*) 6
--	Decrement	a = 5; --a	(a *equals*) 4

Operators precedence :

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

if / else Statement

```
const temperature = 105;
const rain = false;

if ((temperature >= 100) && (rain === false)){
    console.log("It's pretty hot");
} else {
    console.log("It's gonna be cold!");
}
```

TERNARY OPERATOR

```
const temperature = 105;

(temperature >= 100) ? console.log("It's pretty hot") : console.log("It's gonna be cold!");
```

Switch Statement

```
const dayOfTheWeek = 2;

switch (dayOfTheWeek){
    case 1 :
        console.log("Monday");
        break;
    case 2 :
        console.log("Tuesday");
        break;
    case 3 :
        console.log("Wednesday");
        break;
    case 4 :
        console.log("Thursday");
        break;
    case 5 :
        console.log("finally Friday");
        break;
    case 6 :
        console.log("Saturday");
        break;
    case 7 :
        console.log("Sunday");
        break;
    default:
        console.log("Not a number representing a day of the week!");
}
```

Alberto Giovanelli

FUNCTIONS

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).

```
function sum (x,y) {
    return x + y;
};

sum(x1,y1);
```

ARROW FUNCTIONS

There are two benefits to arrow functions.

First, they are less verbose than traditional function expressions:

```
function inc(x) {
    return x + 1;
}

//is equivalent to :
let inc = x => x + 1;

//2 parameters :
let inc = (x,y) => x + y;

//no parameters :
let inc = () => 7;
```

```
//more than 1 statement :
let inc = (x) => {
    console.log(x);
    return 7;
}
```

ARROW FUNCTIONS

Second, their this is picked up from surroundings (*lexical*). Therefore, you don't need bind() anymore.

Arrow functions reference

Understanding this keyword

Understanding bind(), call() and apply()

Array

```
const names = ['John', 'Jane', 'Mark'];

const years = new Array(1990, 1969, 1948);

console.log(names); // ['John', 'Jane', 'Mark']
console.log(names[1]) //Jane

names.push('Eliot');
console.log(names); // ['John', 'Jane', 'Mark', 'Eliot']

names.unshift('Luke');
console.log(names); // ['Luke', John', 'Jane', 'Mark', 'Eliot']

names.pop();
console.log(names); // ['Luke', John', 'Jane', 'Mark']

names.shift();
console.log(names); // ['John', 'Jane', 'Mark']
```

Classes

```
class Person {
    constructor(name, yearOfBirth, job) {
        this.name = name;
        this.yearOfBirth = yearOfBirth;
        this.job = job;
    }
    calculateAge(){
        const age = new Date().getFullYear - this.yearOfBirth;
        console.log(age);
    }
}
```

```
class Athlete extends Person {
    constructor(name, yearOfBirth, job, olympicGames, medals) {
        super(name, yearOfBirth, job);
        this.olympicGames = olympicGames;
        this.medals = medals;
    }

    wonMedal() {
        this.medals++;
    }
}
```

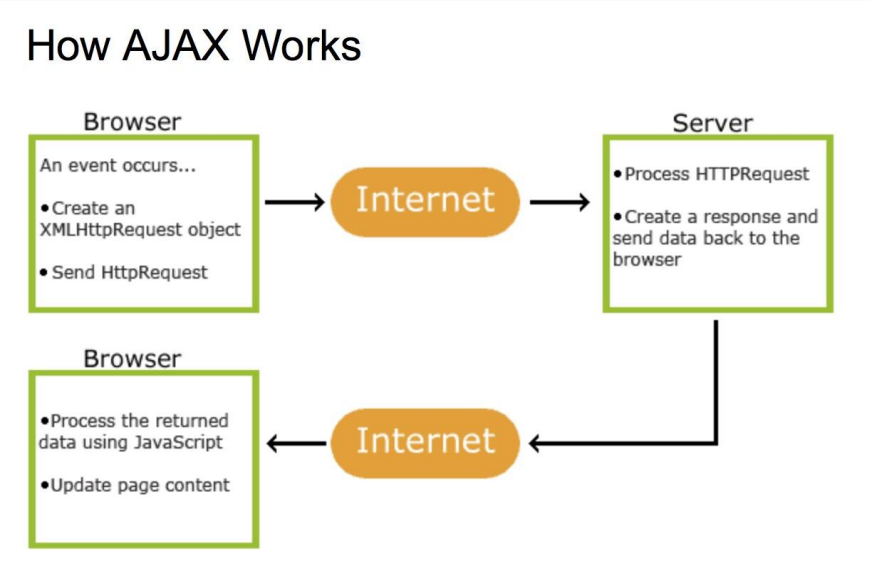
```
const johnAthlete = new Athlete('John', 1990, 'swimmer', 3, 10);

johnAthlete.wonMedal();
```

Alberto Giovanelli

digital GARAGE

ASYNC PROGRAMMING - AJAX

- The development of HTML apps with AJAX is based on data exchange in background, between web browser and server, which allows the dynamic refresh of the web page without forcing the user to refresh the page.
- AJAX is asynchronous; the extra data required from the server are loaded in background without interfering with the behaviour of the existing web page.



Alberto Giovanelli

digital GARAGE

ASYNC PROGRAMMING ES5

- Classic solution : **callback**.

```
1  const update = function(callback) {
2      setTimeout(()=> callback('slow data'), 5000)
3  }
4
5  update(slowData => {
6    //process slowData
7  })
```

ASYNC PROGRAMMING ES6 PROMISES

- Promises are a clean way to implement async programming in JavaScript

```
const update = function() {
    let promise = new Promise((resolve,reject) => {
      setTimeout(()=> resolve('slow data'), 5000)
    })
    return promise
}

update().then(
  slowData => {
    //process slowData
  },
  error => {
    //handle error
  })
```

Introduction to Promises

ES6 Promises - basics

ASYNC PROGRAMMING ES6 PROMISES

```
1  // fetchOrder() returns Promise
2  // fetchUser() returns Promise
3  // fetchCompany() returns Promise
4
5  const getCompanyFromOrder = function(orderId) {
6
7    let promise = fetchOrder(orderId)
8      .then(order => fetchUser(order.userId))
9      .then(user  => fetchCompany(user.companyId))
10
11   return promise
12 }
13
14 getCompanyFromOrder().then(company => {
15   //zrób coś z firmą
16 })
17
```

Alberto Giovanelli

digital GARAGE

MODULES (ES6)

- native ES6 modules are not implemented yet.
- use tool such as Browserify/**Webpack**.

employee.js ● example.js

```
1  export class Employee {
2      constructor(name){
3          this._name = name;
4      }
5
6      getName() {
7          return this._name;
8      }
9
10     work() {
11         return `${this._name} is working`;
12     }
13 }
```

employee.js example.js ●

```
1  import {Employee} from './employee';
2
3  let e = new Employee('Fabio');
4
5  e.work(); //Fabio is working
```

OTHER ES6 FEATURES

- Default Parameters
- map, filter, reduce
- Iterators
- Map / Set
- Generators
- Destructuring
- Spread
- Short End assignment

SUMMARY

- use transpilers (Babel) to write ES6 today for any browser.
- learn ES6 step by step, you don’t have to know everything at once.
- many features are syntactic sugar, use with moderation.

WHAT'S NEXT?

- **MOZILLA DOCS** https://developer.mozilla.org/en-US/docs/Web/JavaScript
- **ES SPEC** https://github.com/tc39/ecma262
- **PLURALSIGHT** https://www.pluralsight.com/courses/javascript-fundamentals-es6
- **YOUTUBE** https://www.youtube.com/channel/UCO1cgjhGzsSYb1rsB4bFe4Q

SINGLE PAGE APPLICATION (SPA)

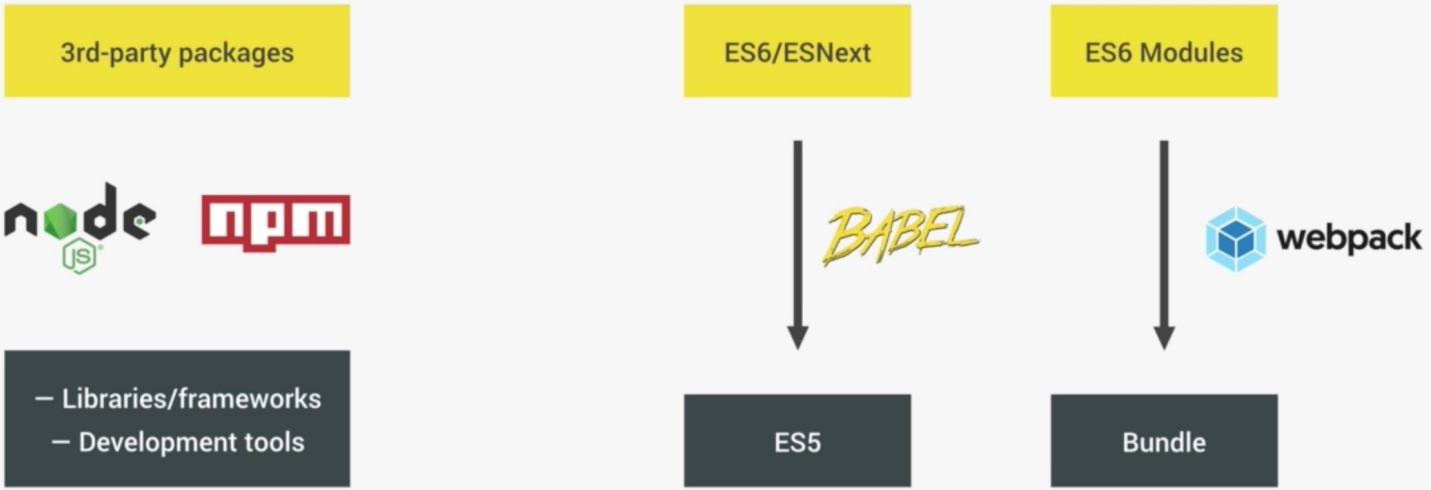
- A single-application (SPA) is a web app or a website where all the content is framed in a single web page. The purpose is to give a more fluid and solid experience to the user, similar to desktop applications and traditional OS.

- In an SPA, either all necessary code – HTML, CSS, JavaScript – is retrieved with a single page load,[1] or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions.

- The page does not reload at any point in the process, nor does control transfer to another page, although the location hash can be used to provide the perception and navigability of separate logical pages in the application.

- Interaction with the single page application often involves dynamic communication with the web server behind the scenes.

MODERN JAVASCRIPT : A BRIEF OVERVIEW



digital GARAGE

NODEJS & NPM

NodeJS is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side.

npm is the package manager for JavaScript and the world's largest software registry.

To create a new npm project, once you have npm installed, just type :

```
npm init
```

Alberto Giovanelli

digital GARAGE

INSTALLING NODE.JS

To install nodejs use a package manager like homebrew for macOS.

```
brew install node
```

Otherwise go to : https://nodejs.org

- After installing, run **node -v**. The version should be v8.9.1 or higher.
- When you install node.js, npm is automatically installed. However, npm gets updated more frequently than Node.js, so be sure that you have the latest version.
- To test, run **npm -v**.

BABEL & WEBPACK

The compiler for writing next generations JavaScript.

Babel Website

Webpack is a module bundler, its main purpose is to bundle all Javascript files for usage in a browser, even if you can do much more with it.

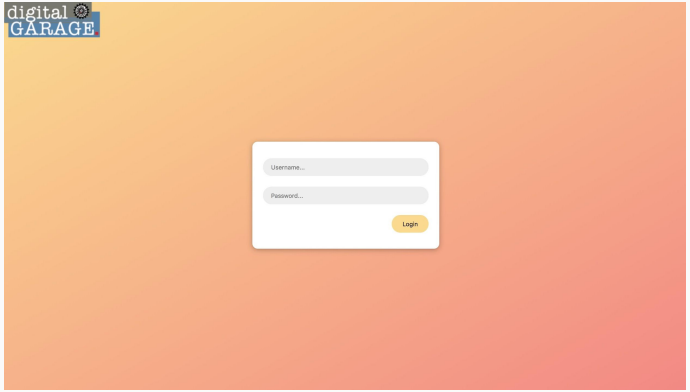
Webpack Website

Alberto Giovanelli

digital GARAGE

PROJECT

- Create a fake login
    - First, make sure to retrieve data from the view.
    - Second, compare credentials inserted with some mocks.

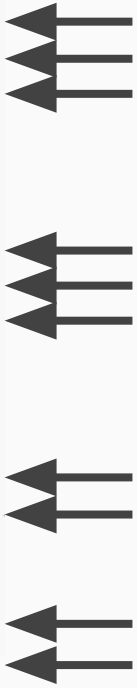


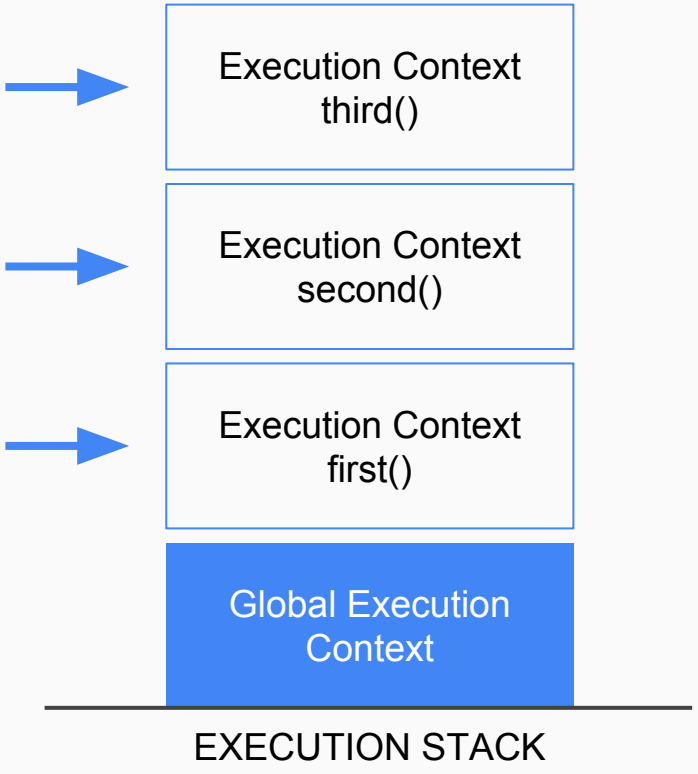
- git clone **https://github.com/albertogiovanelli/digitalgarage-introduction-to-ES6.git**

digital GARAGE

EXECUTION CONTEXT

```
var name = 'John';

function first() {
    var a = 'Hello!';
    second();
    var x = a + name;
}

function second() {
    var b = 'Hi!';
    third();
    var z = b + name;
}

function third() {
    var c = 'Hey!';
    var z = c + name;
}

first();
```





Alberto Giovanelli

digital GARAGE