# Unit 1

## INTRODUCCIÓN A JAVASCRIPT

### DESARROLLO WEB EN ENTORNO CLIENTE 23/24

ALEJANDRO VIANA RÍOS

# Unit 1

Introducción a JavaScript

# Index

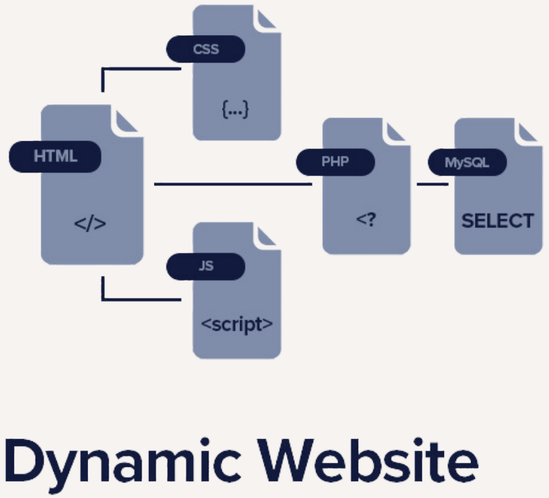
# Basic concepts

1. Static vs dynamic websites
2. Frontend vs backend
3. Frameworks


## Static websites

- Web page is already present in the server with the information fullfiled
- Client gets web page from server
- Renders it and shows to the user
- Information never change
- User interacts only by reading it

---




## Dynamic websites

- Web page is not present in the server  when the client ask for it
- When client requires a web page, the server creates it by connecting to a database
- Every time a web page is required, it could be different depending on user, time, settings...

# Front-end

## What is?

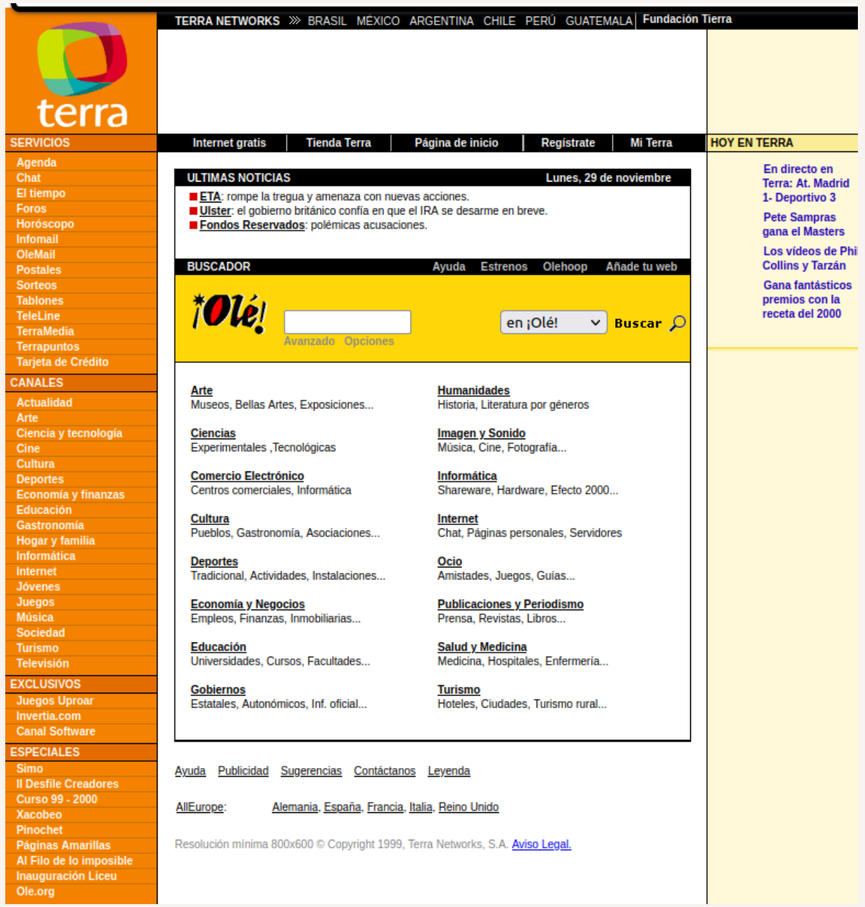
- Client-side part of a web application
- Everything a user sees and interacts with
- How fast the website loads
- How easy and confortable it is to navigate through it
- How accessible it is to people with disabilities.

## Why is so important?

Clients may run away, never to be seen, if a bad user experience is provided

## Where could I get more information?

- Diseño web en entorno cliente (DWEC, 2º DAW)
- Diseño de interfaces web (DIW, 2º DAW)




Back on the 1990's, all sites that wanted to be on the top offered a lot of information. It was not easy to find the information you were looking for nor navigate through them

# Front-end

## Developer tasks

- Create a style guide (typography, images, colors, etc.)
- Create a user interface that matches brand's identity
- Create a website usable and accesible on a wide variety of devices, browsers, connections and screen widths.
    - Usable: optimized load times, easy interaction...
    - Accesible: people with disabilities

## Developer skills

- Knowledge of your target audience and demographic.
- Visual creativity
- Prefer artistic side of the job
- Careness about how users feel

# Front-end technologies

## Technologies

For building static websites

- HTML 5. Markup language used for structuring and presenting content on the World Wide Web
- CSS 3. Language used for describing the presentation of an HTML document.
- JavaScript ES6 or avobe (up to ES13). Language used to provide interactivity to a web site

# Front-end technologies

## Technologies

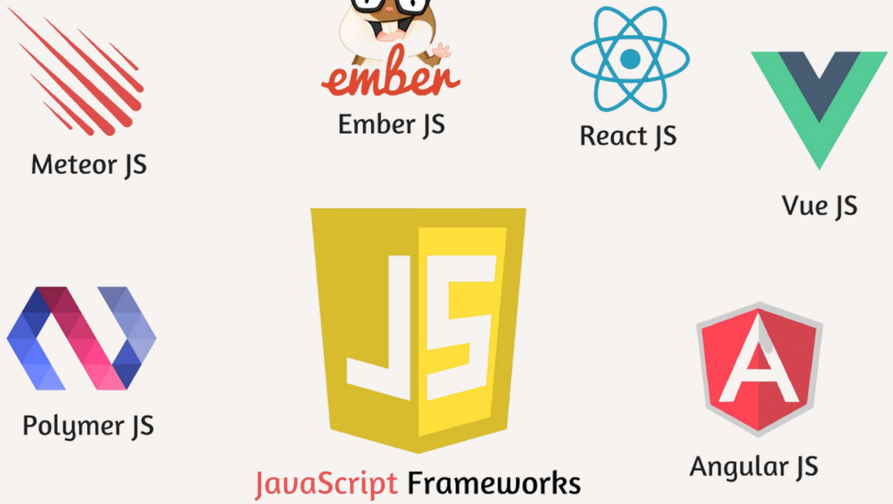
For building dynamic websites

- JavaScript ES6 or avobe (up to ES13)
- Frameworks (Vue, Redux, Angular, etc) (DWEC). Set of tools, focused on a programming language, to support and aid on building software

programa

- Libraries (Node.JS, JQuery, etc.). Pieces of functional code, already proven, free and ready to be used on a project.
- Prototyping (Figma, Sketch, etc.) (DIW)

lo usaremos en la otra asignatura, prototipado: la primera fase de construcción de algo, no tiene toda la funcionalidad, puede cambiar más adelante, pero me sirve para ver cómo va y enseñarselo al cliente. Es útil.





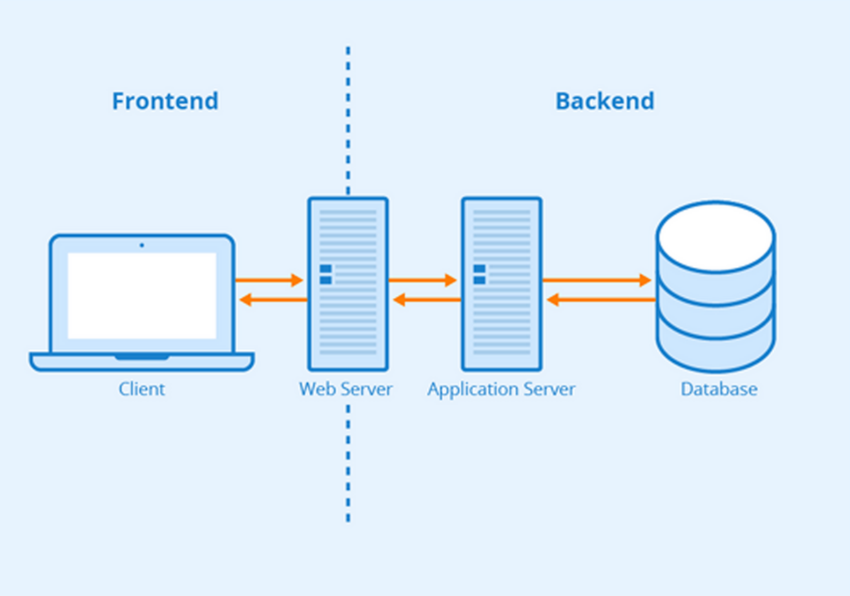
# Back-end

## What is?

- Server-side part of a web application
- Related with how data is stored and manipulated
- Algorithm that determines what application does

## Why is so important?

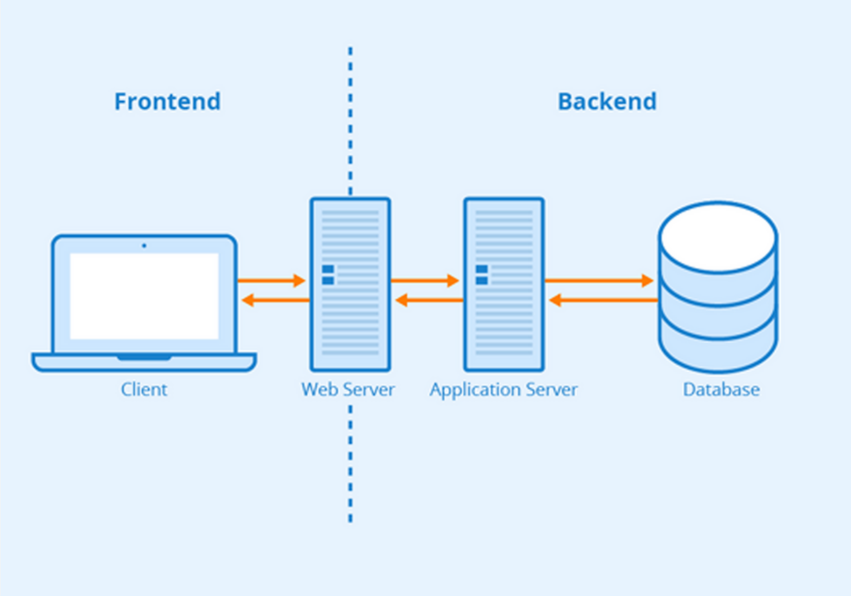
- Responsible of presenting data to client
- Responsible of how the website behaves and what it allows to users
- Bad design could leave website unusable or slow




Frontend and backend. It is common that web server, application server and database server are hosted on the same server

cliente pide algo al servidor, este busca en base de datos y cuando la encuentra se la devuelve al cliente. El servidor web está siempre pendiente, escuchando.
El cliente es el navegador

# Back-end




Frontend and backend. It is common that web server, application server and database server are hosted on the same server

## How does it work?

- A server (web server) is listening for incoming requests from the client (web browser)
- Server-side scripts processes the request
- They collects data from database, process it and send it to the client

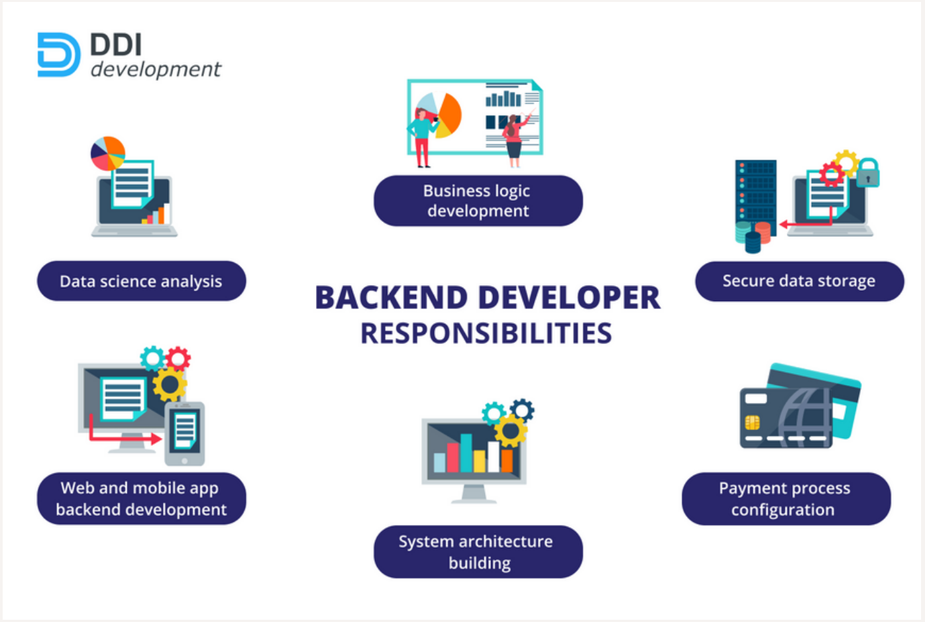
## Where could I get more information?

Diseño web en entorno servidor (2º DAW)

# Back-end

## Developer tasks

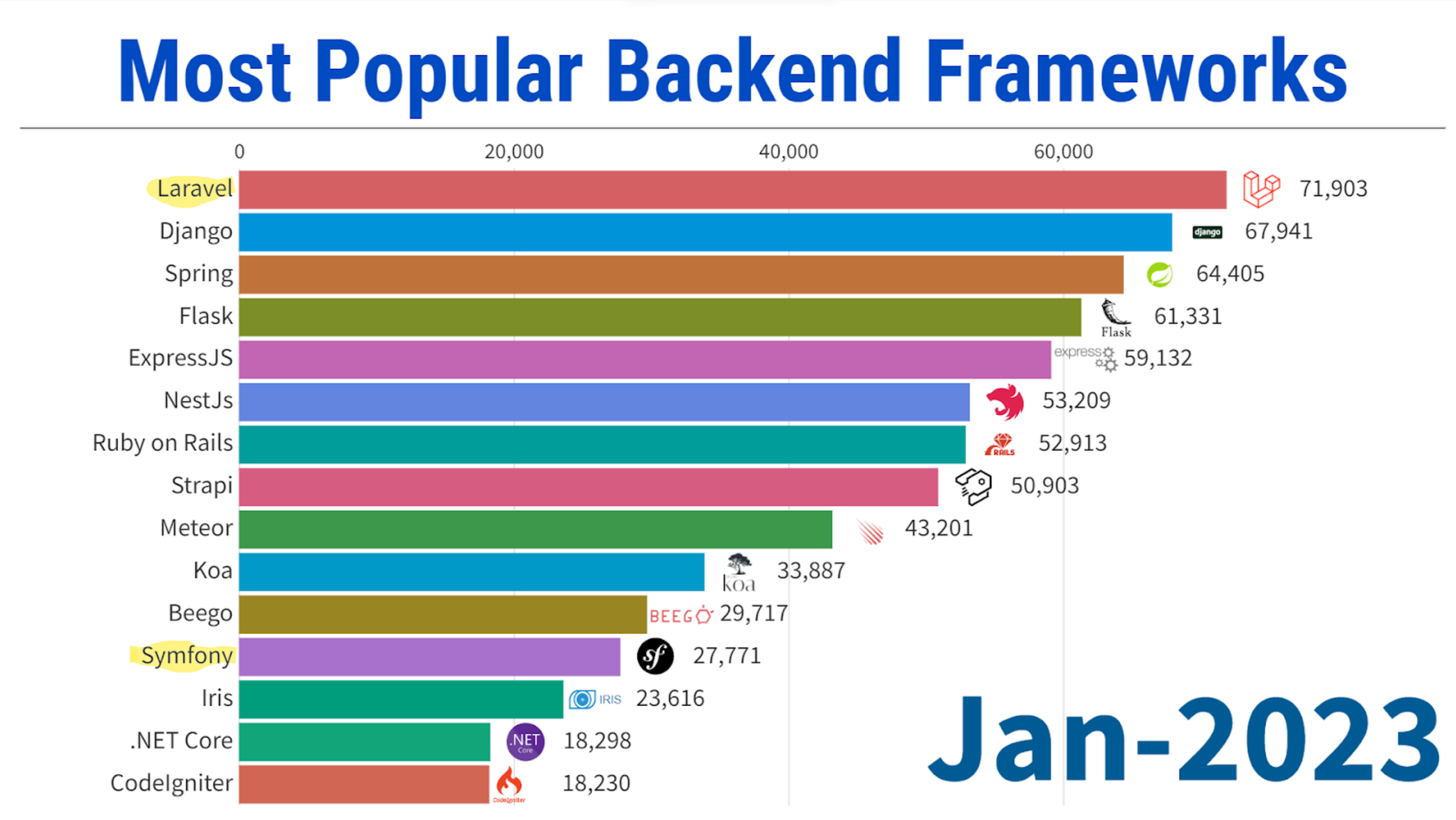
- Creating, managing, and maintaining the database
- Installing and mantaining servers
- Creating scripts to develop website functionallity
- Working with APIs that, usually, support CRUD opperations
- Interacting with third-parties services




## Developer skills

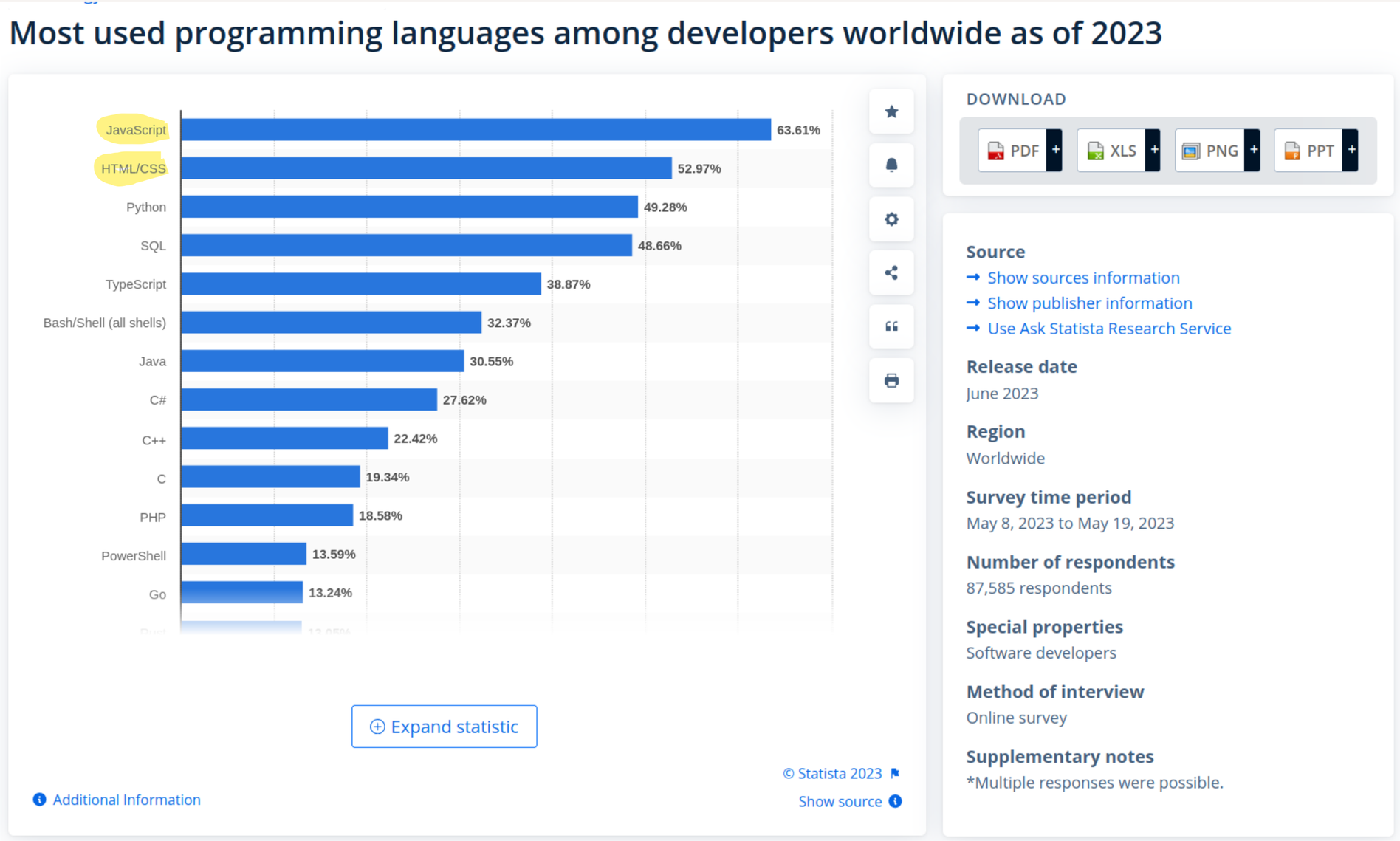
- Problem-oriented profile perfil orientado al problema
- Structured, analytical and methodical person
- Proeficiency analyzing and organizing large amounts of data

# Back-end technologies



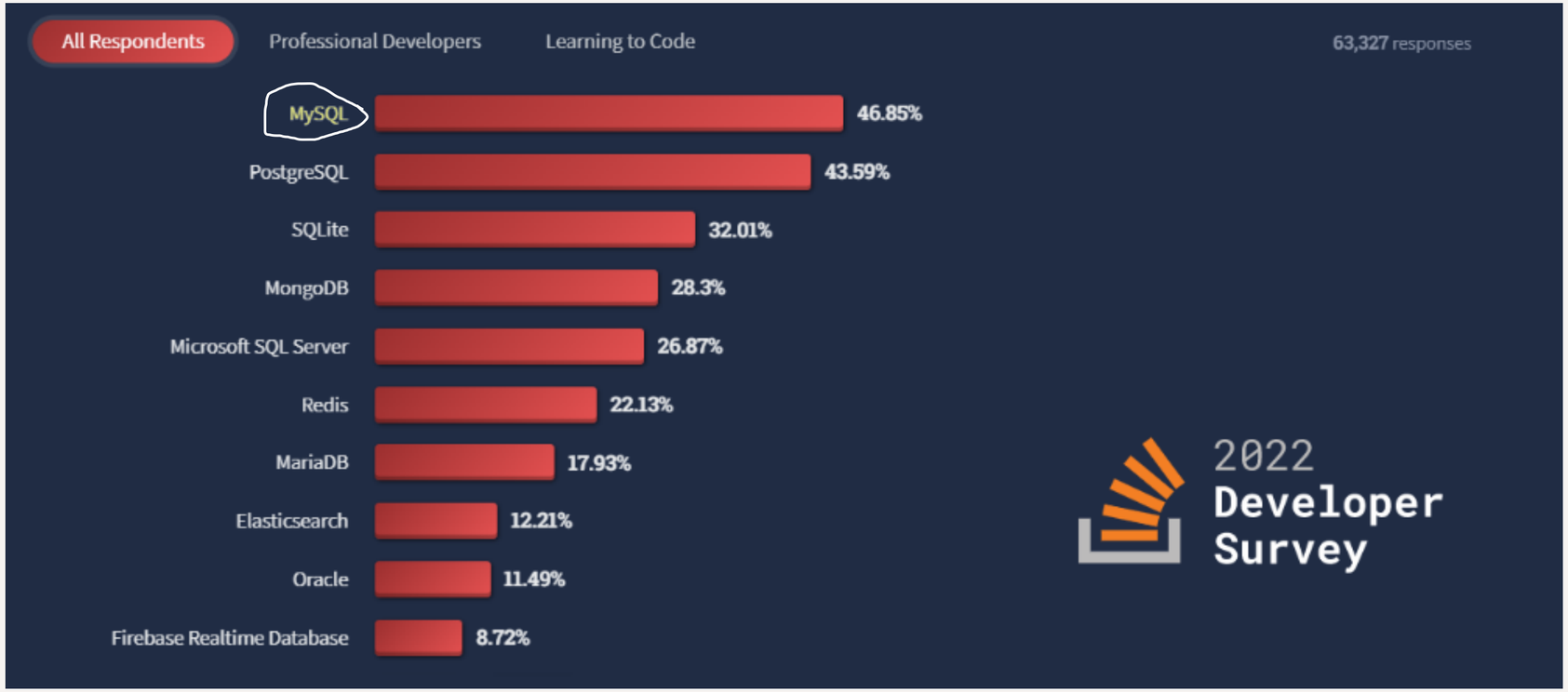

Backend frameworks. A framework is a Set of tools, focused on a programming language, to support and aid on building software

# Back-end technologies



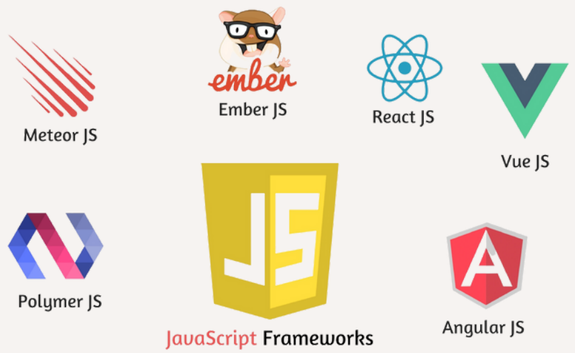

Backend most used programming languages. Yes, Javascript. Thanks to Node.js

# Back-end technologies




Backend most used database servers

# Frameworks




Javascript Frameworks


PHP Frameworks

## What is a Framework?

- Software that helps to build software
- Provides a scaffolder to work with
- Provides aids, tasks and programmes that speeds up the software building
- Each one is focused on one programming language

Desventaja: Cuando uso ayudas para programar el código que se genera no es el óptimo nunca. Genera una serie de estructuras que él considera que es lo mejor, de forma genérica, pero no la que se adapta como un guante = hecho a medida

Pero no es razonable hacer las cosas en binario (por ejemplo), así que tiene que haber balance entre optimización de máquina y tiempo de desarrollo. Merece la pena usar framework

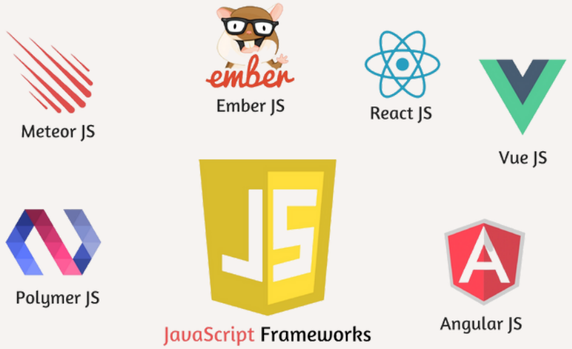
# Frameworks

## Advantages of using a Framework

- you can focus on the core software development rather than the details
- It saves time
- Reduces the number of errors
- It’s easier to collaborate with colleagues
- Avoids redundant software
- Helps to write clean and secure code

## Disadvantages of using a Framework

- the code is not optimal
- you lose control over your code




Javascript Frameworks


PHP Frameworks

# 2.- Introduction to JavaScript

1. What is JavaScript?
2. What in-browser JavaScript can and can’t do?
3. JavaScript-based languages
4. How to add Javascript to your website
5. DOM and BOM
6. JavaScript tips

# What is Javascript

## What is?

- Programming language used to provide interaction to websites
- Alongside HTML and CSS, it's the third technology absolutely essential for a website to be created.
- Traditionally clien-side, but thanks to Node, also server-side
- Provides interpreted code

p=parrafo

```
const para = document.querySelector("p");
    evento es cuando ocurre algo (movimiento de raton, etc)
para.addEventListener("click", updateName);

function updateName() {
  const name = prompt("Enter a new name");
  para.textContent = `Player 1: ${name}`;
}
```

a Javascript code for....

```
1   const colors = ["green", "red", "rgba(133,122,200)", "#f15025"];
2   const btn = document.getElementById("btn");
3   const color = document.querySelector(".color");
4
5   btn.addEventListener("click", function () {
6     const randomNumber = getRandomNumber();
7     // console.log(randomNumber);
8
9     document.body.style.backgroundColor = colors[randomNumber];
10    color.textContent = colors[randomNumber];
11  });
12
13  function getRandomNumber() {
14    return Math.floor(Math.random() * colors.length);
15  }
```

a Javascript code for....

# What in-browser Javascript can and can’t do?

## What can be done?

- Modifying the HTML document. Anything
- React to user actions
- Interact with remote servers
- Running code in response to events
- Work with cookies
- Work with browsers’ APIs: DOM, Geolocation, Audio, Video, Canvas and WebGL.
- Work with third parties’ APIs: Twitter, Google Maps, etc.

## What can’t be done?

- Interact with other tabs or windows
- Communicate with other servers differents where the page came from without permission
- Read or write files on hard disk without permission
- Execute programs
- Acess to camera or microphone without permission

# JavaScript-based languages

## Other languages

- JavaScript syntax does not please to everybody
- Some languages, similar to JavaScript, have appeared : CoffeScript, TypeScript, etc.
- They have to be translated (transpiled) into JavaScript for the browser to execute it.

| Typescript | Javascript |
| --- | --- |
| class Greeter {<br>  greeting: string;<br>  constructor (message: string) {<br>  this.greeting = message;<br>  }<br>  greet() {<br>  return "Hello, " + this.greeting;<br>  }<br>} | var Greeter = (function () {<br>  function Greeter(message) {<br>  this.greeting = message;<br>  }<br>  Greeter.prototype.greet = function () {<br>  return "Hello, " + this.greeting;<br>  };<br>  return Greeter;<br>})(); |

# How to add Javascript to your website

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
    <script>
        let d = new Date();
        alert("Today's date is " + d);
    </script>
</head>

<body>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
</head>

<body>
  <script>
      let d = new Date();
      document.body.innerHTML = "<h1>Today's date is " + d + "</h1>"
  </script>
</body>
</html>
```

Code is directly written on HTML code, under script tag. It is really a bad idea.

# How to add Javascript to your website

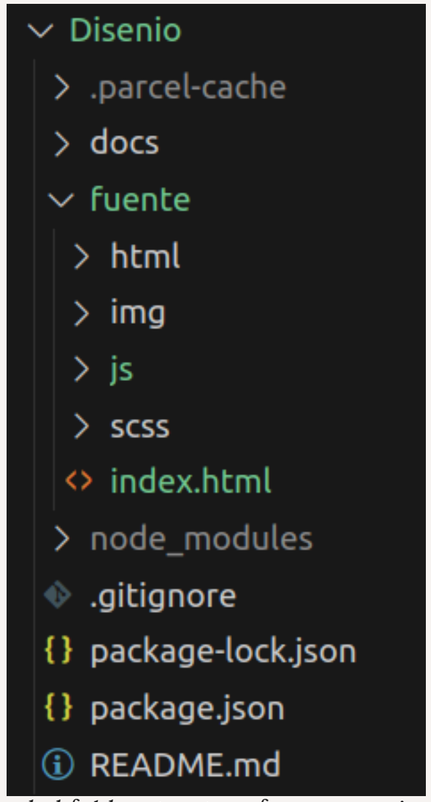
```html
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Today's Date</title>
    <link rel="stylesheet" href="css/style.css">
</head>

<body>
    <script src="js/script.js"></script>
</body>

</html>
```

As an external file summoned from the HTML file, by using script tag. It is the recommended way




Recommended folder structure for your projects. Javascript must be under js folder

# How to add Javascript to your website

## How HTML code is processed?

- HTML code is rendered in the same order it is written on the file.
- When browser finds a script tag, it stops HTML rendering and starts executing the script.

---

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>A heading</h1>
  <p>...content before script...</p>
  <a href="html/about.html">About</a>

  <script src="https://mydomain.com/long.js"></script>

  <!-- Everything below isn't visible until the script loads -->
  <p>...content after script...</p>
  <a href="html/terms.html">Terms</a>
</body>
</html>
```

What would happen if long.js modified all “a” tags?

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>A heading</h1>
  <p>...content before script...</p>
  <a href="html/about.html">About</a>
  <p>...content also before script...</p>
  <a href="html/terms.html">Terms</a>
  <script src="https://mydomain.com/long.js"></script>
</body>
</html>
```

A first solution would be to place script at the end of the HTML code but, what would happen if the script took A LOT OF TIME to load?

# How to add Javascript to your website

## First real solution: defer

- Scripts are loaded in background while HTML is rendered. HTML rendered is not blocked
- Scripts are executed only when the HTML is fully rendered (DOM is fully build)
- Script order is kept. Important when a script depends on the previous one.

---

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>A heading</h1>
  <p>...content before script...</p>
  <a href="html/about.html">About</a>

  <script src="https://mydomain.com/js/long.js" defer></script>
  <script src="js/short.js" defer></script>

  <p>...content after script...</p>
  <a href="html/terms.html">Terms</a>
</body>
</html>
```

defer loads scripts while creating the DOM

# How to add Javascript to your website

## Second real solution: async

- Scripts are loaded in background while HTML is rendered. HTML rendered is not blocked (same as defer)
- Scrip execution DOES NOT WAIT for the HTML to be fully rendered (DOM)
- Script order is NOT kept. First loaded, first executed.

---

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>A heading</h1>
  <p>...content before script...</p>
  <a href="html/about.html">About</a>

  <script src="https://mydomain.com/js/long.js" async></script>
  <script src="js/short.js" async></script>

  <p>...content after script...</p>
  <a href="html/terms.html">Terms</a>
</body>
</html>
```

async loads scripts while creating the DOM

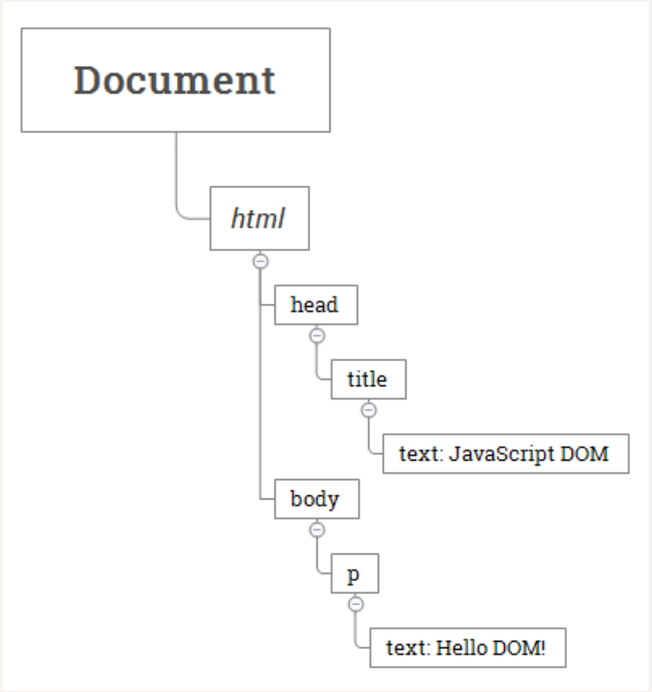
# DOM and BOM

## What is DOM?

- Javascript API to manipulate HTML documents
- Representation of an HTML code as a tree made by web browsers
- It allows Javascript to manipulate HTML document even after it has been rendered

---

```
<html>
    <head>
        <title>JavaScript DOM</title>
    </head>
    <body>
        <p>Hello DOM!</p>
    </body>
</html>
```

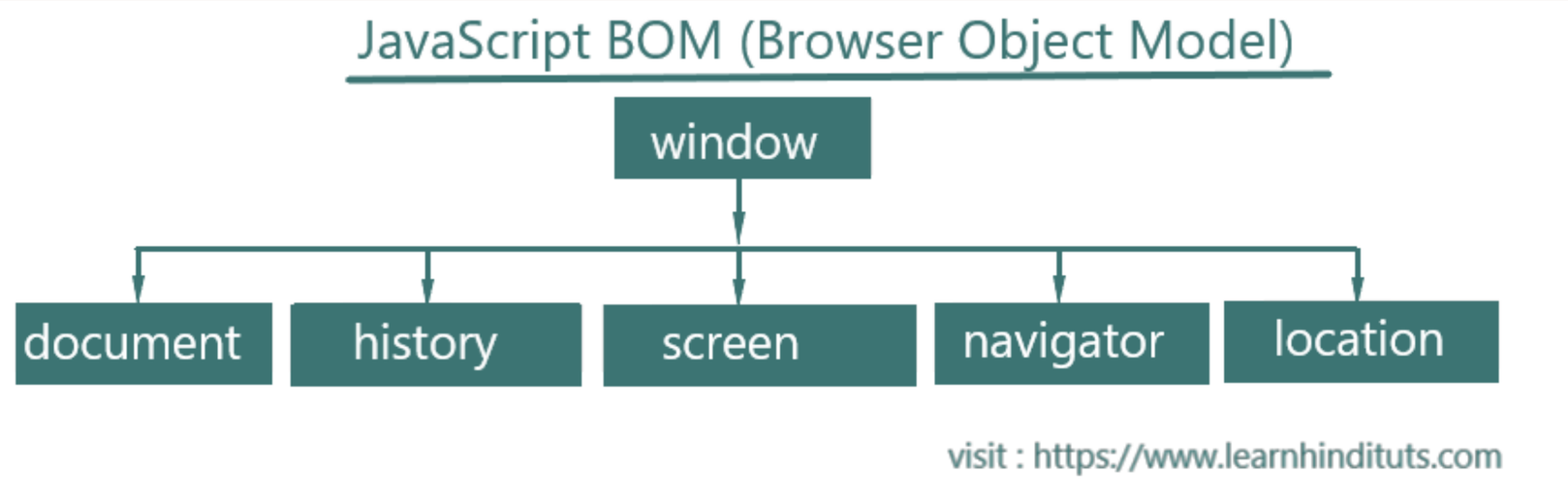


# DOM and BOM

## What is BOM?

- Javascript API to interact with browser and manipulate window and document objects
- It is possible thanks to several objects: window, location, history, navigator and screen
- It allows Javascript to manipulate HTML document even after it has been rendered

---



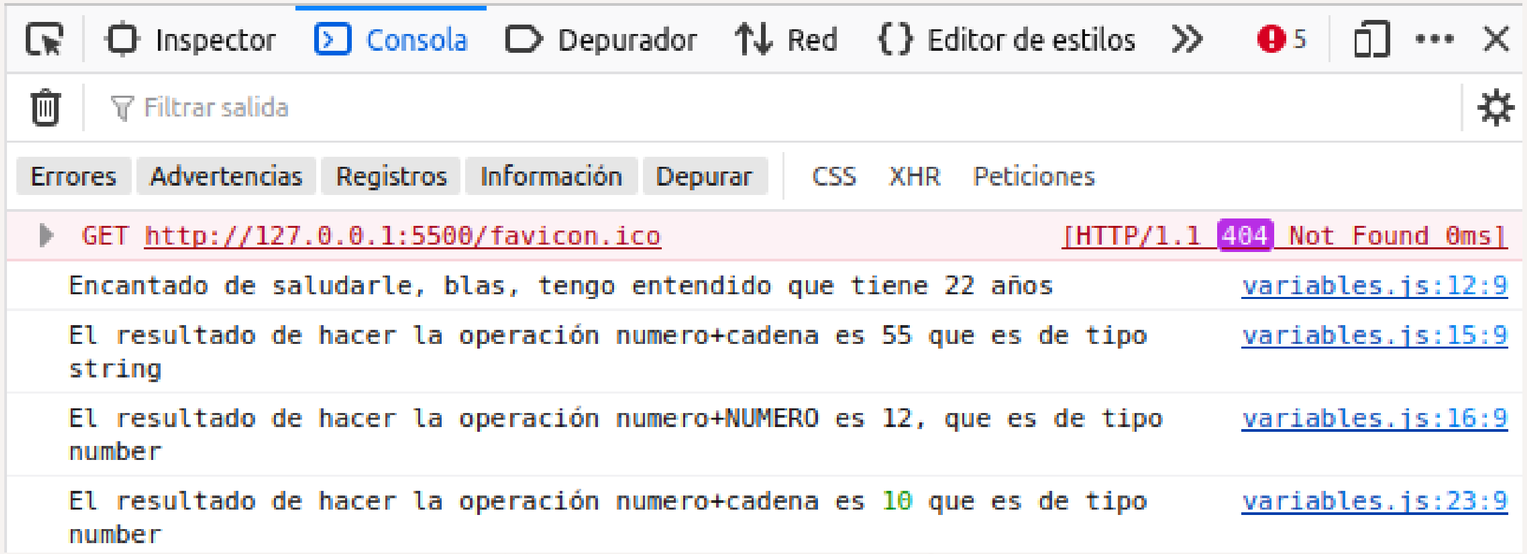
# JavaScript tips

- JavaScript is really flexible but in order to facitlitate debuggin it is better to...
  - ...have one sentence in one line. Do not split one sentence into several lines
  - ...have one sentence per line ending. Do not place several sencentes in the same line (although using semicolons)
  - ...use semicolons to finish each sentence although JavaScript automatically adds it
  - ...declare vars with let and initialize at creation, if possible.
  - ...do not change variable type during its life
  - ...use const when a variable is not changing its value ever
- Use plenty of comments (with // or /**/)
- Debug with console.log (better than alert)

# JavaScript tips

```
1  //puesto que vamos a usar codigo moderno, hay que especificar use strict
2  "use strict";
3
4  //ejemplo del uso de variables, constantes y tipos
5  //las variables se deben definir con let. var es anticuado
6  let numero=5;
7  let cadena;
8  cadena="5";
9  const NUMERO=7, CADENA="6";
10
11 const NOMBRE = prompt("Dígame su nombre");
12 console.log (`Encantado de saludarle, ${NOMBRE}, tengo entendido que tiene ${prompt ("dígame su edad")} años`);
13 alert (`Buenos días ${NOMBRE}`);
14
15 console.log ("El resultado de hacer la operación numero+cadena es", numero+cadena, `que es de tipo ${typeof(numero+cadena)}` );
16 console.log (`El resultado de hacer la operación numero+NUMERO es ${numero+NUMERO}, que es de tipo ${typeof(numero+NUMERO)}`);
17
18
19 /*Las variables en javascript son débilmente tipado, es decir, no declaro su tipo y éste puede cambiar a lo largo de la vida de la variable
20 Es recomendable que los nombres de las variables sean lo más descriptivos posible. El siguiente caso es una mala práctica*/
21
22 cadena=5 //¡la variable llamada "cadena" es de tipo entero! ¡¡MAL!!
23 console.log ("El resultado de hacer la operación numero+cadena es", numero+cadena, `que es de tipo ${typeof(numero+cadena)}` );
```

# JavaScript debuggin




web browser console showing results of JavaScript