**Become a Front End Developer**

**with Altimetrik | 2nd Edition**

**Documentation**

[Vanessa Martínez]

Index

**Agile Software Development Methodologies** .……………………….1

Types of Methodologies …………………………………………………………1

Scrum .…………………………………………………………………………………………1

Kanban .………………………………………………………………………………………..2

Extreme Programming ………………………………………………………………………...3

**Git** ………………………………………………………………………3

More common git commands ……………………………………………………4

Tags and commits ………………………………………………………………..6

Stash commands …………………………………………………………………6

Hooks …………………………………………………………………………….6

Git rebase and squash ……………………………………………………………7

Difference between git merge and git rebase ……………………………………8

**Javascript - Part 1** ……………………………………………………..8

Syntax ……………………………………………………………………………8

Console in Javascript …………………………………………………………….8

Data types in Javascript ………………………………………………………….8

Variables in Javascript …………………………………………………………...9

Comments in Javascript ………………………………………………………...10

Functions in Javascript …………………………………………………………10

**HTML** ………………………………………………………………...10

HTML Elements ………………………………………………………………..10

HTML Tags ……………………………………………………………………..11

Basic tags ……………………………………………………………………………………11

Unclosed tags ………………………………………………………………………………...11

Meta - tags ……………………………………………………………………………………11

Input types ……………………………………………………………………...12

Difference between HTML and XHTML ………………………………………13

HTML Attributes………………………………………………………………..13

Accessibility ……………………………………………………………………14

Semantic HTML ………………………………………………………………..14

**SEO** ……………………………………………………………………15

The three pillars of SEO ………………………………………………………..15

**DOM** …………………………………………………………………..15

Difference between SEO and DOM ……………………………………………16

**CSS** ……………………………………………………………………16

Syntax …………………………………………………………………………..16

Selectors ………………………………………………………………………...16

Specificity ………………………………………………………………………17

Box model ………………………………………………………………………18

**Javascript - Part 2** ……………………………………………………19

Hoisting in Javascript …………………………………………………………..19

How to access DOM in Javascript ……………………………………………...19

Scope in Javascript ……………………………………………………………..21

Strict Mode in Javascript ……………………………………………………….21

**CSS - Parte 2** ………………………………………………………….21

Media Queries …………………………………………………………………..21

Syntax ……………………………………………………………………………………….21

REM, EM, PX ………………………………………..

**Agile Software Development Methodologies**

*Agile software development, or “Agile”, refers to a group of development methodologies that anticipates the need for flexibility, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams to the delivery of the finished product. It focuses on the clean and fast delivery of individual pieces or parts of the software and not on the entire application.*

*Some benefits:*

*- The ability to help teams in a complex area while it’s still focused on the delivery of business value.*

*- Improves efficiency throughout the organization as teams work together and understand their specific roles in the process.*

*- Companies can trust on who are using agile methodologies because they can feel they are releasing a high-quality product since testing is performed throughout development.*

*- Provide the opportunity to make changes as needed and alert teams to any potential issues*

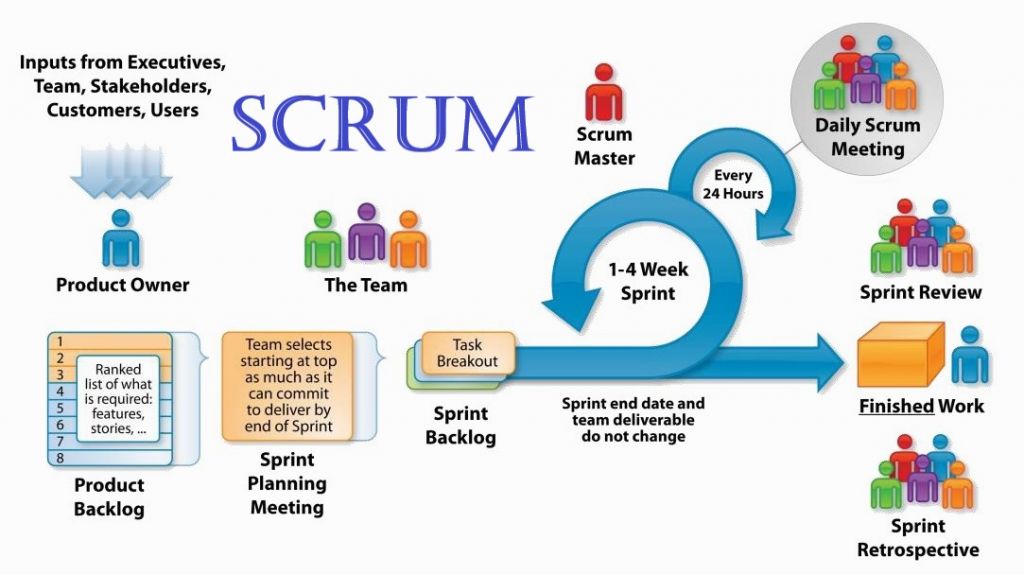
*´.*

***Types of Agile Methodologies***

***Scrum***

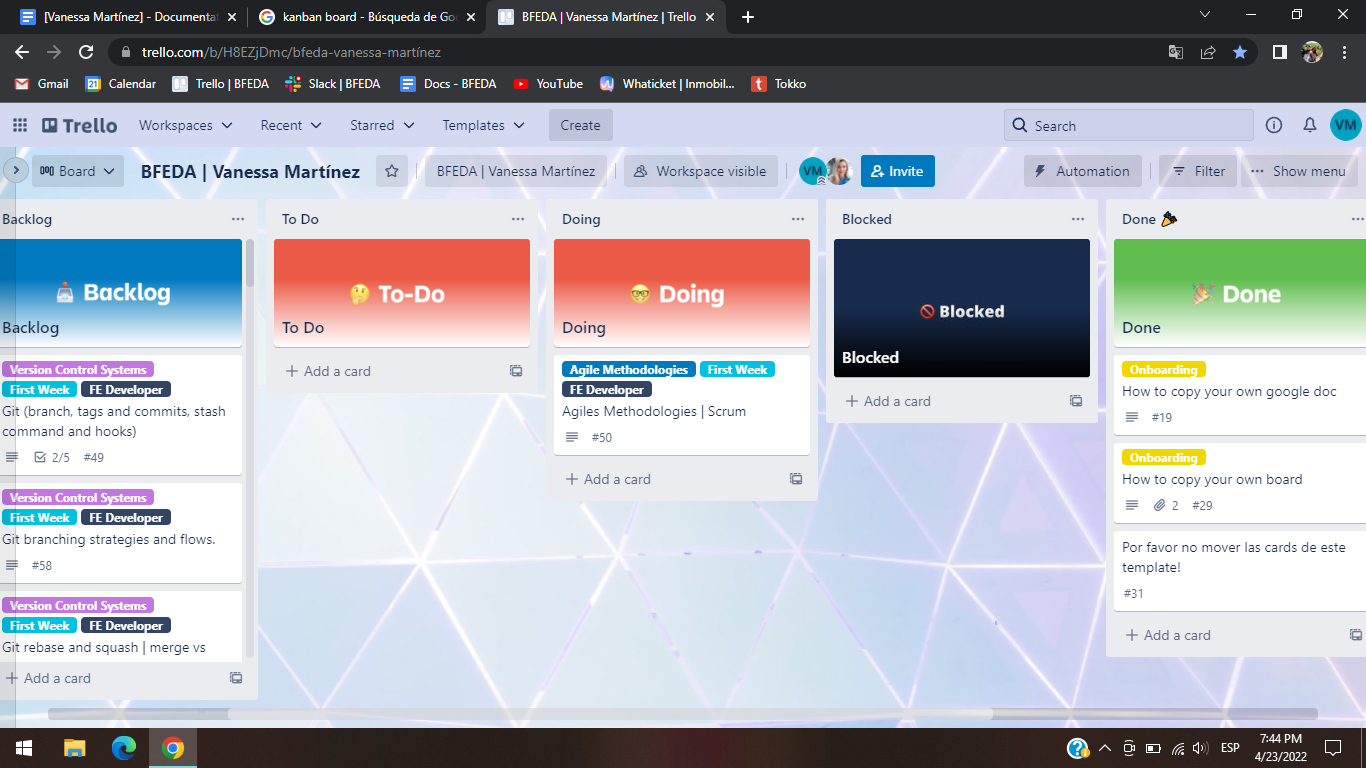
*Scrum is a methodology that helps people, teams and organizations to self-organize, and applies a set of good practices to work together and obtain the best results, generating value by solving complex problems. It replaces a programmed algorithmic approach with a heuristic one (a set of techniques to solve a problem).*

***Roles and process:***

******

***Kanban***

*Kanban is a form to organize tasks with cards on a board that has no limit. You can have a few columns to pass, beginning on the ‘backlog’ column, constantly updated, once the task has begun, you can move the cards to another column called ‘in progress’, or similar, and then through the correspondent columns until it is ‘done’.*

**

***“Scrum is a technique and Kanban is a specific form to apply it”***

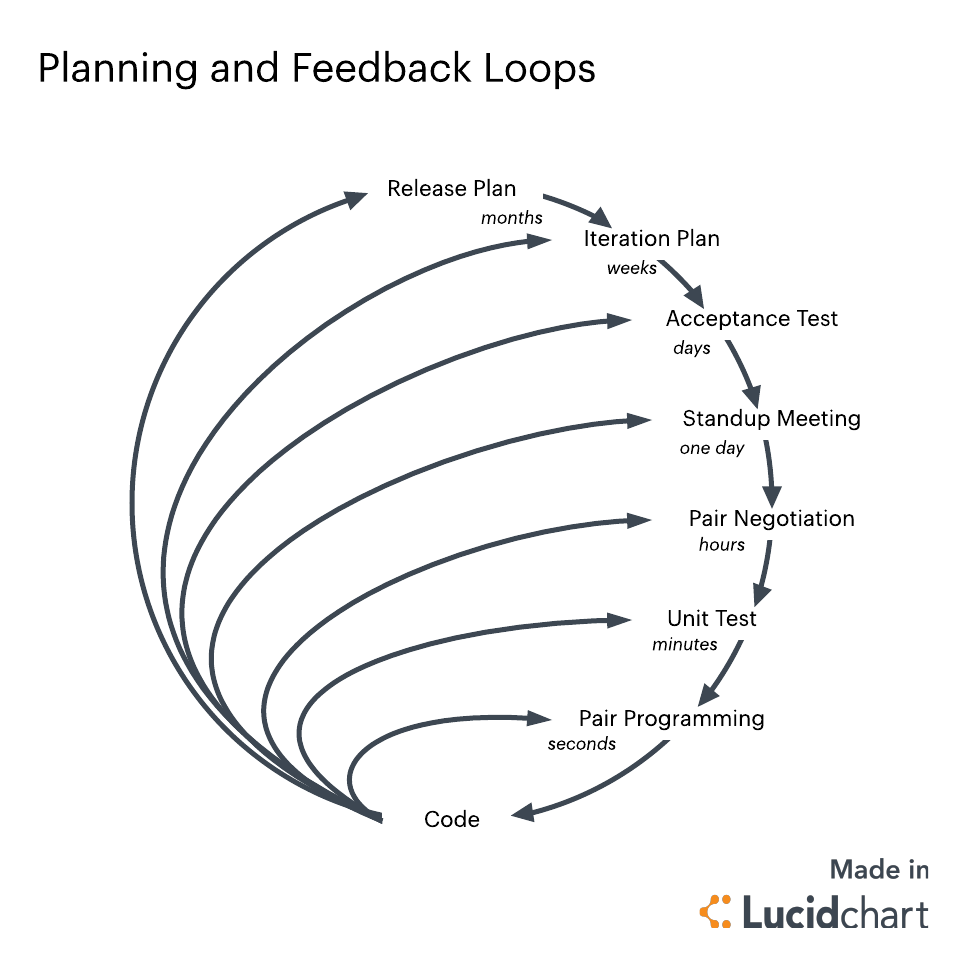
***Extreme Programming (XP)***

*Extreme programming is a framework that aims to produce higher quality software. XP is the most specific of the agile frameworks regarding appropriate engineering practices.*

*XP has 5 values:*

* *Simplicity*
* *Communication*
* *Feedback*
* *Courage*
* *Respect*

***Rules of XP:***

**

**Git**

*Git is a function that controls code’s versions in distributive form that is an important part of daily programming. It allows you to have a complete version history in a fast and free way, this system works with branches and these branches can grow up in different directions from the main one to try out new functionalities.*

***More common git commands***

* ***Git clone:*** *It’s a command that makes an identical copy of the latest version of a project in your local workspace.*

*git clone <https://name-of-the-repository-link>*

* ***Git branch:*** *It’s used to create, list and delete branches.*

*To create one locally, you use: git branch <branch-name>*

*To push it into the remote repository, you use: git push -u <remote> <branch-name>*

*To list or view it, you use: git branch or git branch --list*

*To delete it, you use: git branch -d <branch-name>*

* ***Git Checkout:*** *It’s mostly used for switching from one branch to another and for checking out files and commits.*

*git checkout <name-of-your-branch>*

*You can also use another command to create and switch a branch at the same time:*

*git checkout -b <name-of-your-branch>*

* ***Git Status:*** *It gives you all the information about branches and files, gathering information like if the preset branch is up to date, if there is anything to commit, pull or push, if there are files created, modified or deleted.*

*git status*

* ***Git Add:*** *If we create, make a change or erase files, those changes will be on our workspace and won’t be in the next commit. To save it we have to use this command to include those changes into the next commit.*

*To add a file: git add <file>*

*To add everything at once: git add -A*

* ***Git Commit:*** *It’s a command used to save changes we have done after a specific task or issue on our local workspace. We need to write a short message to explain what we did in the source code.*

*git commit -m "commit message"*

* ***Git Push:*** *It sends your changes to the remote server which already are committed.*

*git push <remote> <branch-name>*

*If your branch is new and you need to upload it too, you can use: git push --set-upstream <remote> <name-of-your-branch> or git push -u origin <branch\_name>*

* ***Git Pull:*** *It’s used to get updates from the remote repository and applies the latest changes in your local.*

*git pull <remote>*

* ***Git Revert:*** *It’s used to undo changes in a safe way.*

*First we have to see our commit history, using: git log -- oneline*

*Then specify the hash code of the commit that we want to undo: git revert 3321844*

*Finish pressing shift + q to exit.*

*Really, this command doesn't delete the commit from the history, just create a new “revert” one, you can still see every single move that you did.*

* ***Git Merge:*** *when everything is working ok, it’s used to integrate your feature branch with all of its commits back to the dev branch.*

*Before merging, you have to switch to the dev branch: git checkout dev*

*Then update it using: git fetch*

*At the end, merge your feature branch into dev: git merge <branch-name>*

***Tags and Commits***

*Tags is a feature used to specify a point on a repository. It’s mostly used to keep the release version of a repo, using this tag you are able to move it to an earlier version.*

*git status*

*git add <file>*

*git commit -m “short message”*

*git tag v-1.0 <the correspondent version>*

*git tag -n*

*A* ***tag*** *can be created to a specific* ***commit*** *too, to create it, follow the commands bellow:*

*git status*

*git add <file>*

*git commit -m “short message”*

*git log -- oneline*

***Stash command***

*We can use* ***git stash*** *when we want to temporarily record our current state of the working directory, saving our local modifications away, to manage this copy on another side, here we can make the changes needed without damaging the main and feeling overwhelmed, when everything is fine, you can put it back on the original one.*

***Hooks***

*They are scripts that run every time a certain important action occurs in a repo. Exist two groups of these hooks:*

* ***Client-side hooks:*** *are activated by operations such as committing and merging.*
* ***Server-side hooks:*** *run on network operations such as receiving pushed commits.*

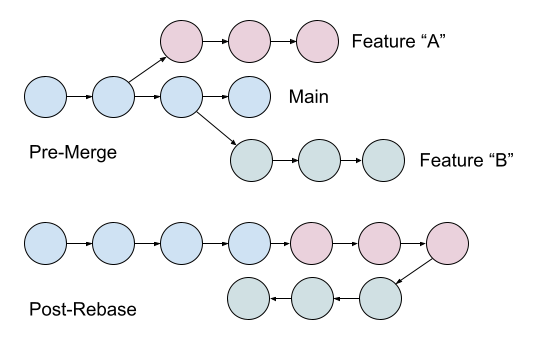
*The most useful local hooks:*

* *pre-commit*
* *prepare-commit-msg*
* *commit-msg*
* *post-commit*
* *post-checkout*
* *pre-rebase*

***Git rebase and squash***

***Rebase:*** *When we have 2 different features branches to include in the main one, mostly you will get conflict errors. If a feature branch doesn’t consider the other one, bugs cloud appear in a feature branch.*

*Using git rebase, it will change the base of a branch, not create another original point, just putting that on a different point in time.*

**

*To get it, use something like this:  
git checkout featureA*

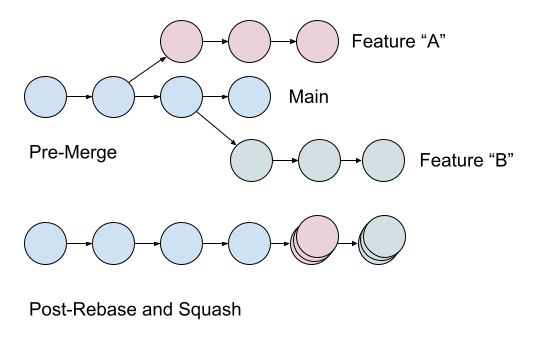
*git fetch origin*

*git rebase origin/main Rebase the latest "main" to "Feature A"*

*git checkout main*

*git merge featureA*

***Squash:*** *It’s used to merge all your commits of one feature branch into a single commit, and then you can put it at the end of the main branch.*

**

*To get it, first you have to rebase and then add:  
git merge featureA --squash Squash "Feature A" to end of main*

***Difference between git merge - git rebase***

* *Rebase reapplies commits of the top of another base branch (rewrite it).*
* *Merge joins two or more development histories together (preserved history as it happened).*

**Javascript - Part 1**

***Javascript*** *is a dynamic programming language used for web development.*

*With Javascript you will have more versatility to create style declarations, supporting math calculations, extracts contents from another website, and more.*

*It has more features we can explain, but by understanding the basic constructs you can begin to use it, and learn more and more while you are working with it.*

***Syntax***

*Refers to the set of rules that determines how JavaScript programs are constructed: variables, operator, expression, keyword, comments, data types, functions, etc.*

***Console in Javascript***

*It shows the informacion of the web you are playing at the moment. It’s an object used to access the web console of the browser.*

*The console.log ( ) function is used to print any kind of variables defined before in it or to just print any message that needs to be displayed to the user.*

*The console.table ( ) function is used to display data in tabular form, taking data (can be an array or an object) and an additional parameter.*

***Data types in Javascript***

1. ***Number:*** *Numerical value.*
2. ***String:*** *Sequences of characters that represents a value.*
3. ***Boolean:*** *Logical values, true or false.*
4. ***Null:*** *Denote a null value. (Remembering JS is case-sensitive, null isn’t the same than NULL).*
5. ***Undefined:*** *Represents non-existence of data.*
6. ***Symbol:*** *Unique values created from string keys.*
7. ***Object:*** *it is a related data collection, consists in variables and functions, when these are inside objects, they are known as properties and methods.*
8. ***Array:*** *It is a set of related data by positions associated by one variable.*

***Variables in Javascript***

*Variables are containers for storing data values. They are identificators, inasmuch as JS differences uppercase and lowercase, you have a lot of characters to use, from ‘A’ to ‘Z’ and from ‘a’ to ‘z’, underscore and symbols to begin a variable, it can be followed by numbers.*

***3 forms to declare a variable in Javascript***

| *var* | *let* | *const* |
| --- | --- | --- |
| * *It’s used in all JavaScript code from 1995 to 2015.* * *It creates a function scoped variable.* * *Can be reassigned.* * *Can be redeclared.* * *It’s hoisted to the top of its executions (global or function) and initialized as undefined.* | * *Added to JavaScript in 2015, in ES6.* * *It creates a block - scope variable.* * *Can be reassigned.* * *Cannot be redeclared.* * *It’s hoisted to the top of its executions (global or block) and left uninitialized.* | * *Added to JavaScript in 2015, in ES6.* * *It creates a block - scope variable.* * *Cannot be reassigned.* * *Cannot be redeclared.* * *It’s hoisted to the top of its executions (global or block) and left uninitialized.* |

***Scope of a variable***

* ***Global Variable:*** *it’s when you declare a variable outside a function and it will be available to others codes.*
* ***Local Variable:*** *it’s when you declare a variable inside a function and it will be just available inside this function.*

***Comments in Javascript***

*Comments are notes in the source code of a program that can be used to write important things about the code, explaining it and making the code more readable, and also be used to prevent execution.*

***Functions in Javascript***

*Functions are the block of code designed, used to do particular tasks, they are used when something calls it, it can call functions several times, so we can say functions are reusables. To write it you can implement the same rules of variables.*

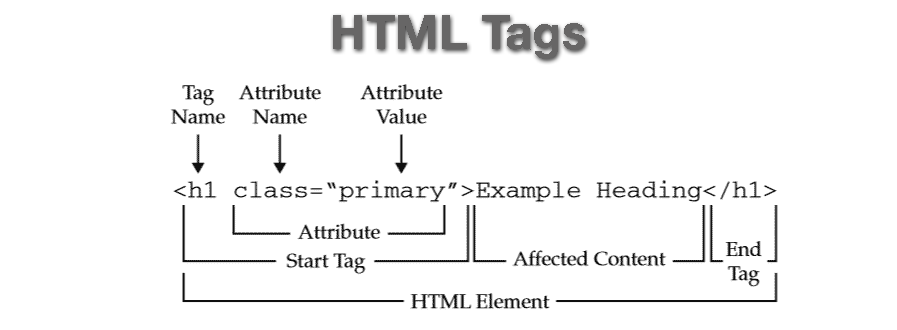
**HTML**

*Hyper Text Markup Language, as its name says, is a markup language used for creating web pages and applications, describing the structure of these last one, with the help of styling.*

***HTML Elements***

*It’s an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements. It’s defined by a start tag, some content, and an end tag*

*<tagname> words </tagname>*

**

*Example:*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title> write the title </title>*

*</head>*

*<body>*

*<h2> something like a subtitle</h2>*

*<p>This is a paragraph tag</p>*

*<p style="color: purple">The style is attribute of paragraph tag</p>*

*</body>*

*</html>*

***HTML tags***

*HTML Tags are always written in lowercase letters, HTML5 includes 142 tags. Exists different categories, here are some of them:*

***Basic tags***

*<!DOCTYPE> : defines the document type*

*<html> : defines an HTML document*

*<head> : contains metadata/information for the document*

*<title> : defines a title for the document*

*<body> : defines the document's body*

*<h1> to <h6> : defines HTML headings*

*<p> : defines a paragraph*

*<br> : inserts a single line break*

*<hr> : defines a thematic change in the content*

*<!--...--> : defines a comment*

***Unclosed tags***

* *<br> : br stands for break line, it breaks the line of the code.*
* *<hr> : hr stands for Horizontal Rule to put a line across the page.*

***Meta - tags***

* *<!DOCTYPE> : defines the document type.*
* *<title> : defines a title for the document.*
* *<link> : defines the relationship between a document and an external resource (most used to link to style sheets)*
* *<style> : defines style information for a document.*
* *<meta> : defines metadata about an HTML document.*

***Text tags***

*<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <strong>, <em>, <abbr>, <acronym>, <address>, <bdo>, <blockquote>, <cite>, <q>, <code>, <ins>, <del>, <dfn>, <kbd>, <pre>, <samp>, <var>, <br>*

***Link tags***

*<a>, <base>*

***Image and object Tags***

*<img>, <area>, <map>, <param>, <object>*

***List tags***

*<ul>, <ol>, <li>, <dl>, <dt>, <dd>*

***Table tags***

*table, tr, td, th, tbody, thead, tfoot, col, colgroup, caption*

***Form tags***

*form, input, textarea, select, option, optgroup, button, label, fieldset, legend*

***Scripting tags***

*script, noscript*

***Input types***

*Those are an important part of an element that defines the information field.*

* *text : defines a one-line text input field*
* *password : defines a one-line password input field*
* *submit : defines a submit button to submit the form to server*
* *reset : defines a reset button to reset all values in the form*
* *radio : defines a radio button which allows selecting one option*
* *checkbox : defines checkboxes which allow selecting multiple options form*
* *button : defines a simple push button, which can be programmed to perform a task on an event*
* *file : defines to select the file from device storage*
* *image : defines a graphical submit button*

*And in HTML5 we also have:*

* *color : defines an input field with a specific color.*
* *date : defines an input field for selection of date.*
* *datetime-local : defines an input field for entering a date without a timezone.*
* *email : defines an input field for entering an email address.*
* *month : defines a control with month and year, without a time zone.*
* *number : defines an input field to enter a number.*
* *url : defines a field for entering URL*
* *week : defines a field to enter the date with week-year, without a time zone.*
* *search : defines a single line text field for entering a search string.*
* *tel : defines an input field for entering the telephone number.*

***Difference between HTML and XHTML***

* *XHTML is a stricter version of HTML*
* *XHTML is support by all major browsers*
* *<!DOCTYPE>, <html>, <head>, <title>, and <body> are mandatory*
* *Elements must always be properly nested, closed and in lowercase, tags should appear in pairs.*
* *Attribute names must always be in lowercase*
* *Attribute values must always be quoted*

***HTML Attributes***

*In HTML attributes are special words that give additional information about the element or modify it, it can define the behavior of the element, always apply it with the start tag and its name and value pair. The attributes name and value are case - sensitive and you can write 2 or more attributes in the same element giving space between each one.*

*<element attribute\_name="value">content</element>*

***Global attributes***

*Exists different types and categories of attributes, here are some the global one:*

* *Accesskey : specifies a shortcut key to activate/focus an element*
* *class : specifies one or more classnames for an element (refers to a class in a style sheet)*
* *contenteditable : specifies whether the content of an element is editable or not*
* *dir : specifies the text direction for the content in an element*
* *draggable : specifies whether an element is draggable or not*
* *hidden : specifies that an element is not yet, or is no longer, relevant*
* *id : specifies a unique id for an element*
* *lang : specifies the language of the element's content*
* *spellcheck : Specifies whether the element is to have its spelling and grammar checked or not*
* *style : specifies an inline CSS style for an element*
* *tabindex : specifies the tabbing order of an element*
* *title : specifies extra information about an element*
* *translate : specifies whether the content of an element should be translated or not*
* *data-\* : can be used to define our own custom data attributes. It is used to store custom data in private to the page or application. There are mainly 2 parts of the Data Attributes:*
* ***Attribute Name:*** *Must be at least one character long, contain no capital letters and be prefixed with ‘data-‘.*
* ***Attribute Value****: Can be any string.*

***Accessibility (A11Y)***

*Designing and creating web pages, it’s always important to have accessibility in mind, giving a nice experience to all audiences to navigate and interact with your site, and to make your code semantic. The goal is to make your content accessible to visually-impaired or blind users.*

***Semantic HTML***

*Basically it refers to the correct use of HTML elements for their real purpose, semantic HTML gives context to screen readers making the element focusable.*

*Examples of* ***non-semantic*** *elements: <div> and <span> - Tells nothing about its content.*

*Examples of* ***semantic*** *elements: <form>, <table>, and <article> - Clearly defines its content.*

**SEO**

***Search Engine Optimization*** *is the process or form to position from free, organic, editorial, or natural search results in search engines. With SEO you elevate your website so it can appear in a good position in search results pages, with a high position you will get more views.*

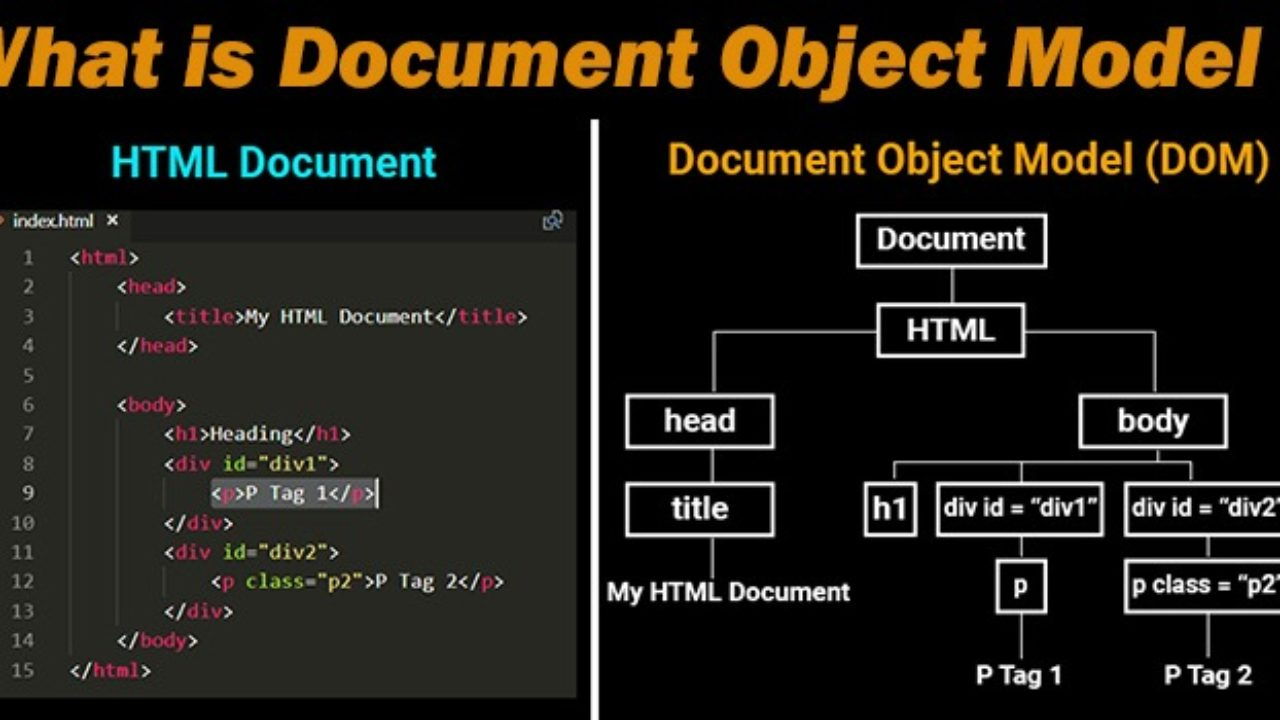
***The three pillars of SEO***

1. ***Technical Optimization:*** *It happens behind the scenes, is the process of completing activities on your site to improve SEO.*
2. ***On-Page Optimization:*** *is the process of ensuring the content on your site is relevant, targeting the right keywords by a content management system like WordPress, Expression Engine, Wix and Shopify.*
3. ***Off-Page Optimization:*** *is the process of improving your site’s rankings, it’s used to help the reputation of your page with activities outside of the site.*

**DOM**

***Document Object Model*** *is a programming API that defines the logical structure of documents and the way a document is accessed and manipulated. Programmers can build documents, navigate their structure, and add, modify, or delete elements and content, with a few exceptions in XML.*

***Example of a DOM representation with HTML:***

**

***Difference between SEO and DOM***

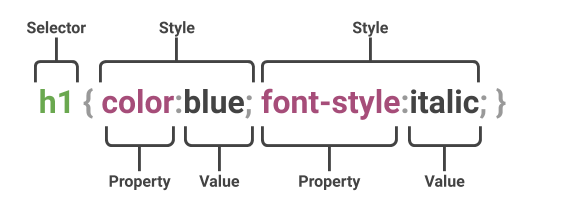
*Search Engine Optimization is a process to position your pages and Document Object Model defines the logical structure of that page.*

**CSS**

***Cascading Style Sheets*** *is the language we use to style an HTML document describing how HTML elements should be displayed. HTML and CSS are strongly tied together, since HTML is a markup language (the very foundation of a site) and CSS emphasizes style (all of the aesthetics of a website), they go hand in hand. HTML without CSS would look completely simple and bare.*

***Syntax***

*The selector points to the HTML element you want to style. The style block contains one or more declarations separated by semicolons. Each style includes a CSS property name and a value separated.*

**

***Selectors***

*Those are used to "find" (or select) the HTML elements you want to style. We can divide CSS selectors into five categories:*

1. ***Simple selectors*** *(select elements based on name, id, class)*
2. ***Combinator selectors*** *(select elements based on a specific relationship between them)*
3. ***Pseudo-class selectors*** *(select elements based on a certain state)*
4. ***Pseudo-elements selectors*** *(select and style a part of an element)*
5. ***Attribute selectors*** *(select elements based on an attribute or attribute value)*

***Basic Selectors***

* ***Element selector****: based on the element name.*

*p {text-align: center; color: purple;}*

* ***ID selector****: uses the id attribute of an HTML element to select a specific element. Considering the ID is unique, to select it you write a hash (#) character, followed by the id of the element (it cannot start with just a number.*

*#para 1 {text-align: center; color: purple;}*

* ***Class selector:*** *selects HTML elements with a specific class attribute, writes a period (.) character, followed by the class name. You can also specify that only specific HTML elements should be affected by a class*
* *.center {text-align: center; color: purple;}*

*- - - - - - - - - - - - - - - - - - - - - - -*

* *p.center {text-align: center; color: purple;}*
* ***Universal selector:*** *selects all HTML elements on the page.*

*\* {text-align: center; color: purple;}*

* ***Grouping Selector:*** *selects all the HTML elements with the same style definitions.*

| *h1 {text-align: center; color: purple;}*  *h2 {text-align: center; color: purple;}*  *p {text-align: center; color: purple;}* | *h1, h2, p {text-align: center; color: purple;}* |
| --- | --- |

***Specificity***

*When you have 2 or more CSS rules pointing to the same element, you have to consider the selector will take the highest one and its style declaration will be applied to that HTML element. Specificity determines which one is ultimately applied to the element.*

*<html>*

*<head>*

*<style>*

*.test {color: purple;}*

*p {color: red;}*

*</style>*

*</head>*

*<body>*

*<p class="test">Specificity example</p>*

*</body>*

*</html>*

***Box Model***

*The box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.*

* ***Content:*** *where text and images appear.*
* ***Padding:*** *The padding is a transparent area that creates space around an element's content, inside of any defined borders. Specify sides of the padding:*

*Padding-top, Padding-right, Padding-bottom, Padding-left*

*Having the following values:*

* *length (px, pt, cm, etc).*
* *% of the width of the containing element.*
* *Inherit, from the parent element.*
* ***Border:*** *goes around the padding and content, allowing you to specify the style, width, and color of an element's border.*

*- Border Style : specifies what kind of border to display and can have from one to four values top, right, bottom, left.The following values are allowed:*

*dotted : defines a dotted border*

*dashed : defines a dashed border*

*solid : defines a solid border*

*double : defines a double border*

*groove : defines a 3D grooved border. The effect depends on the border-color value*

*ridge : defines a 3D ridged border. The effect depends on the border-color value*

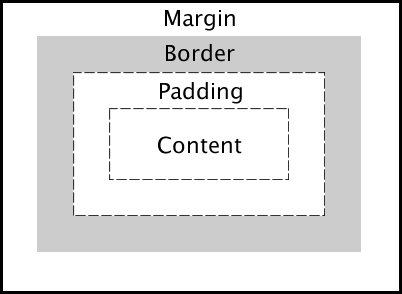
*inset : defines a 3D inset border. The effect depends on the border-color value*

*outset : defines a 3D outset border. The effect depends on the border-color value*

*none : defines no border*

*hidden : defines a hidden border*

* *Margin: area outside the border. The margin is transparent.*

**

**Javascript - Part 2**

*.*

***Hoisting in Javascript***

*It’s the default behavior of moving all the declarations at the top, no matter where functions and variables are declared, their scope regardless of whether their scope is global or local. JavaScript only hoists declarations, not the initializations.*

***How to access DOM in Javascript***

*With the HTML DOM, JavaScript can access and change all the elements of an HTML document.*

***The DOM Programming Interface***

* *The HTML DOM can be accessed with JavaScript (and with other programming languages).*
* *In the DOM, all HTML elements are defined as objects.*
* *The programming interface is the properties and methods of each object.*
* *A property is a value that you can get or set (like changing the content of an HTML element).*
* *A method is an action you can do (like adding or deleting an HTML element).*
* ***The getElementById Method:*** *The most common way to access an HTML element is to use the id of the element.*
* ***The innerHTML Property:*** *The easiest way to get the content of an element is by using the innerHTML property. It is useful for getting or replacing the content of HTML elements.*

*The innerHTML property can be used to get or change any HTML element, including <html> and <body>.*

*To Find HTML Elements* ***- Method:***

* *document.getElementById(id) : Find an element by element id*
* *document.getElementsByTagName(name): Find elements by tag name*
* *document.getElementsByClassName(name: Find elements by class name*

*To Change HTML Elements -* ***Property:***

* *element.innerHTML = new html content - Change the inner HTML of an element*
* *element.attribute = new value - Change the attribute value of an HTML element*
* *element.style.property = new style - Change the style of an HTML element*

*To Change HTML Elements -* ***Method:***

* *element.setAttribute (attribute, value) - Change the attribute value of an HTML element*

*Adding and Deleting Elements -* ***Method:***

* *document.createElement (element) - Create an HTML element*
* *document.removeChild (element) - Remove an HTML element*
* *document.appendChild (element) - Add an HTML element*
* *document.replaceChild (new, old) - Replace an HTML element*
* *document.write (text) - Write into the HTML output stream*

***Scope in Javascript***

*It’s an important concept that manage the visibility of variables, exists 3 types:*

* ***Block scope:*** *defines a scope for variables declared using let and const.*
* ***Function scope:*** *defines a scope for variables declared using var, let and const.*
* ***Global scope:*** *The global scope is the outermost scope, it is accessible from any inner scope. Variables declared Globally have Global Scope.*

***Strict Mode in Javascript***

*Strict mode is a feature added in ES5, it was created to reduce the random behavior and increment the detectability of poorly written code. These made the code much more secure and maintained a high standard of coding in general. Using the strict-mode it was seen that developers could now write highly optimized programs.*

*use strict*

*CSS - Part 2*

***Media queries***

*It’s a feature of CSS 3 allowing content rendering to adapt to different conditions, a CSS technique and a breakpoint.*

*Media queries are useful when you want to modify your site or app depending on a device's general type (such as print vs. screen) or specific characteristics and parameters (such as screen resolution or browser viewport width).*

* *To conditionally apply styles with the CSS @media and @import at-rules.*
* *To target specific media for the <style> , <link> , <source> , and other HTML elements with the media= attribute.*
* *To test and monitor media states using the Window.*

***Syntax***

*A media query is composed of an optional media type and any number of media feature expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.*

* ***Media types*** *define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.*
* ***Media features*** *describe a specific characteristic of the user agent, output device, or environment: any-hover, any-pointer, aspect-ratio, color, color-gamut, color-index, device-aspect-ratio Deprecated, device-height Deprecated, device-width Deprecated, display-mode, dynamic-range, forced-colors, grid, height, hover, inverted-colors, monochrome, orientation, overflow-block, overflow-inline, pointer, prefers-color-scheme, prefers-contrast, prefers-reduced-motion, resolution, scripting, update, video-dynamic-range, width. Each media feature expression must be surrounded by parentheses.*
* ***Logical operators*** *can be used to compose a complex media query: not, and, and only. You can also combine multiple media queries into a single rule by separating them with commas.*

***REM, EM, PX***

***rem:*** *to define the size of a letter of the different sections and components we use rem. It’s depende of the base font-size. If we haven’t defined any font-size in HTML, it will take the navigator one. Namely, 1rem = 16px.*

*Example : if we want a title to have a size of 24px, we apply font-size: 1.5rem.*

***em:*** *Em is a relatively flexible and scalable unit of typography. It is equal to the font-size specified to the parent element. Your browser translates em to pixels. It can turn out to be a tricky part. Em is a relatively flexible and scalable unit of typography. It is equal to the font-size specified to the parent element. Your browser translates em to pixels.By using em, we can create components on the page that respond automatically, should the font size change.*

***Pixel:*** *it is an absolute and fixed-size unit in CSS. Although the size of a pixel isn’t always the same, the font-size, margin, and padding in pixel(px) we use the pixels as an absolute unit, and it remains the same for all screens.*

***Layouts***

*CSS page layout techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors: their default position in normal layout flow, the other elements around them, their parent container, and the main viewport/window. The page layout techniques we'll be covering in more detail in this module are:*

* *Normal flow*
* *Positioning*
* *Flexbox*
* *Grid*
* *Floats*
* *The display property*
* *Table layout*
* *Multiple-column layout*

***Positioning***

*Positioning allows you to take elements out of normal document flow and make them behave differently, for example, by sitting on top of one another or by always remaining in the same place inside the browser viewport.*

*The position property specifies the type of positioning method used for an element.*

*There are five different position values:*

* *Static positioning is the default that every element gets. It just means "put the element into its normal position in the document layout flow — nothing special to see here".*
* *Relative positioning allows you to modify an element's position on the page, moving it relative to its position in normal flow, as well as making it overlap other elements on the page.*
* *Absolute positioning moves an element completely out of the page's normal layout flow, like it's sitting on its own separate layer. From there, you can fix it to a position relative to the edges of its closest positioned ancestor (which becomes <html> if no other ancestors are positioned). This is useful for creating complex layout effects, such as tabbed boxes where different content panels sit on top of one another and are shown and hidden as desired, or information panels that sit off screen by default, but can be made to slide on screen using a control button.*
* *Fixed positioning is very similar to absolute positioning except that it fixes an element relative to the browser viewport, not another element. This is useful for creating effects such as a persistent navigation menu that always stays in the same place on the screen as the rest of the content scrolls.*
* *Sticky positioning is a newer positioning method that makes an element act like position: static until it hits a defined offset from the viewport, at which point it acts like position: fixed.*

*Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.*

***Flexbox layout***

*Flexbox is the short name for the Flexible Box Layout CSS module, designed to make it easy for us to lay things out in one dimension, either as a row or as a column. To use flexbox, you apply display: flex to the parent element of the elements you want to lay out; all its direct children then become flex items.*

***Setting display: flex***

*The HTML markup below gives us a containing element with a class of wrapper, inside of which are three <div> elements. By default these would display as block elements, that is, below one another in our English language document.*

*However, if we add display: flex to the parent, the three items now arrange themselves into columns. This is due to them becoming flex items and being affected by some initial values that flexbox sets on the flex container. They are displayed in a row because the property flex-direction of the parent element has an initial value of row. They all appear to stretch in height because the property align-items of their parent element has an initial value of stretch. This means that the items stretch to the height of the flex container, which in this case is defined by the tallest item. The items all line up at the start of the container, leaving any extra space at the end of the row.*

***Grid Layout***

*Grid Layout is designed to make it easy for us to lay things out in two dimensions, in rows and columns.*

***Setting display: grid***

*Similar to flexbox, we enable Grid Layout with its specific display value — display: grid. The below example uses similar markup to the flex example, with a container and some child elements. In addition to using display: grid, we also define some row and column tracks for the parent using the grid-template-rows and grid-template-columns properties respectively. We've defined three columns, each of 1fr, as well as two rows of 100px. We don't need to put any rules on the child elements; they're automatically placed into the cells our grid's created.*

**CSS Frameworks**

*……*

***Bootstrap***

*Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites. Bootstrap 5 is the newest version, it supports all major browsers except Internet Explorer 11 and down.*

***Bootstrap Quickstar:***

*<div class="bg-primary text-white p-5 text-center">*

*<h1>My First Bootstrap Page</h1>*

*<p>example</p>*

*</div>*

*<div class="container-fluid">*

*<div class="row">*

*<div class="col-sm-4">*

*<h2>a title</h2>*

*<p>text</p>*

*<p>text</p>*

*</div>*

*<div class="col-sm-4">*

*<h2>title</h2>*

*<p>text.</p>*

*<p>text</p>*

*</div>*

*<div class="col-sm-4">*

*<h2>title</h2>*

*<p>text</p>*

*<p>text</p>*

*</div>*

*</div>*

*</div>*

***Bootstrap Containers:*** *The container class is one of the most important Bootstrap classes. It provides margins, padding, alignments, and more, to HTML elements.*

*<div class="container">*

*<h1>This is a paragraph</h1>*

*<p>This is a paragraph</p>*

*<p>This is a paragraph</p>*

*<p>This is a paragraph</p>*

*<p>This is a paragraph</p>*

*</div>*

***Bootstrap Colors***

*<div class="container bg-primary text-white p-4">*

*<p>text</p>*

*</div>*

*<div class="container bg-success text-white p-4">*

*<p>more text</p>*

*</div>*

***Bootstrap Text Colors***

*<div class="container">*

*<p class="text-muted">This text is muted.</p>*

*<p class="text-primary">This text is important.</p>*

*<p class="text-success">This text indicates success.</p>*

*<p class="text-info">This text represents some information.</p>*

*<p class="text-warning">This text represents a warning.</p>*

*<p class="text-danger">This text represents danger.</p>*

*</div>*

***Bootstrap Columns:*** *Three equal-width columns, on all devices and screen widths.*

*<div class="row">*

*<div class="col">.col</div>*

*<div class="col">.col</div>*

*<div class="col">.col</div>*

*</div>*

* ***Responsive Columns:*** *Three equal-width columns scaling to stack on top of each other on small screens:*

*<div class="row">*

*<div class="col-sm-4">.col-sm-4</div>*

*<div class="col-sm-4">.col-sm-4</div>*

*<div class="col-sm-4">.col-sm-4</div>*

*</div>*

***Bootstrap Tables***

*<table class="table table-striped table-bordered">*

*<thead>*

*<tr>*

*<th>Firstname</th>*

*<th>Lastname</th>*

*<th>Email</th>*

*</tr>*

*</thead>*

*<tbody>*

*<tr>*

*<td>Vanessa</td>*

*<td>Martínez</td>*

*<td>vansssaamartinezg2611@gmail.com</td>*

*</tr>*

*</tbody>*

*</table>*

***Bootstrap Alerts:*** *Bootstrap provides an easy way to create predefined alert messages.*

*<div class="alert alert-success">*

*<strong>Success!</strong> Indicates a successful or positive action.*

*</div>*

***Bootstrap Buttons:*** *Bootstrap provides different styles of buttons*

*<button type="button" class="btn">Basic</button>*

*<button type="button" class="btn btn-primary">Primary</button>*

*<button type="button" class="btn btn-secondary">Secondary</button>*

*<button type="button" class="btn btn-success">Success</button>*

*<button type="button" class="btn btn-info">Info</button>*

*<button type="button" class="btn btn-warning">Warning</button>*

*<button type="button" class="btn btn-danger">Danger</button>*

*<button type="button" class="btn btn-dark">Dark</button>*

***Bootstrap Cards***

*<div class="card" style="width:400px">*

*<img src="img\_avatar1.png" alt="Card image">*

*<div class="card-body">*

*<h4 class="card-title">Vanessa Martínez</h4>*

*<p class="card-text">example text.</p>*

*<a href="#" class="btn btn-primary">See Profile</a>*

*</div>*

*</div>*

***Materialize CSS***

*Materialize, a modern responsive front-end framework based on Material Design, is a UI component library created with CSS, JavaScript, and HTML. Materialize reusable UI components help in constructing attractive, consistent, and functional web pages and web apps while adhering to modern web design principles such as browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites. It’s similar to bootstrap but inspired by Google Material Design.*

*It works similar to bootstrap, with this library you can apply:*

* *Colors*
* *Colors texts*
* *Grids*
* *Shadows*
* *Tables*
* *Typography*
* *Badges*
* *Buttons*
* *Cards*
* *Forms*
* *Icons*
* *Dialogs*
* *Collections*

*And other useful predetermined things.*

**CSS preprocessors**

*A CSS preprocessor is a program that lets you generate CSS from the preprocessor's own unique syntax.*

***SASS***

*Syntactically Awesome Style Sheets, is a CSS preprocessor, reduces repetition of CSS and therefore saves time, being completely compatible with all versions of CSS. It lets you use features that do not exist in CSS, like variables, nested rules, mixins, imports, inheritance, built-in functions, and other stuff. Sass files have the ".scss" file extension and it supports standard CSS comments /\* comment \*/, and in addition it supports inline comments // comment.*

***Sass Variables***

*Variables are a way to store information that you can re-use later. Sass uses the $ symbol, followed by a name, to declare variables. With Sass, you can store information in variables, like:*

* *strings*
* *numbers*
* *colors*
* *booleans*
* *lists*
* *null*

*$variablename: value;*

***Sass Nested Rules***

*Sass lets you nest CSS selectors in the same way as HTML.*

| ***Nested*** | ***Not nested*** |
| --- | --- |
| *nav {*  *ul {*  *margin: 0;*  *padding: 0;*  *list-style: none;*  *}*  *li {*  *display: inline-block;*  *}*  *a {*  *display: block;*  *padding: 6px 12px;*  *text-decoration: none;*  *}*  *}* | *nav ul {*  *margin: 0;*  *padding: 0;*  *list-style: none;*  *}*  *nav li {*  *display: inline-block;*  *}*  *nav a {*  *display: block;*  *padding: 6px 12px;*  *text-decoration: none;*  *}* |

***Sass Importing Files***

*Just like CSS, Sass also supports the @import directive. The @import directive allows you to include the content of one file in another. The CSS @import directive has a major drawback due to performance issues; it creates an extra HTTP request each time you call it. However, the Sass @import directive includes the file in the CSS; so no extra HTTP call is required at runtime. You can import as many files as you need in the main file.*

*@import filename;*

***Sass Mixins***

*The @mixin directive lets you create CSS code that is to be reused throughout the website.*

*@mixin name {*

*property: value;*

*property: value;*

*...*

*}*

*The @include directive is created to let you use (include) the mixin. And A mixin can also include other mixins.*

*selector {*

*@include mixin-name;*

*@include mixin1-name;*

*@include mixin2-name;*

*@include mixin3-name;*

*}*

***Sass @extend Directive***

*The @extend directive lets you share a set of CSS properties from one selector to another. The @extend directive is useful if you have almost identically styled elements that only differ in some small details.*

*By using the @extend directive, you do not need to specify several classes for an element in your HTML code, like this: <button class="button-basic button-report">Report this</button>. You just need to specify .button-report to get both sets of styles. The @extend directive helps keep your Sass code very DRY.*

*.button-basic {*

*border: none;*

*padding: 15px 30px;*

*text-align: center;*

*font-size: 16px;*

*cursor: pointer;*

*}*

*.button-report {*

*@extend .button-basic;*

*background-color: red;*

*}*

*.button-submit {*

*@extend .button-basic;*

*background-color: green;*

*color: white;*

*}*

**CSS Methodologies**

*CSS methodologies are formal, documented systems for authoring CSS in a way that allows us to develop, maintain and scale the front-end as a set of small, isolated modules. Rather than as one massive lump of indivisible code.*

***OOCSS***

*Object-Oriented CSS advocates the separation of structure from skin. The methodology makes a clear distinction between content and its containers. In OOCSS, style rules are written exclusively using CSS class selectors.*

*OOCSS can help us maintain DRY stylesheets. The methodology attempts to achieve this goal by using lots of small, modular, specialist CSS classes. Very few style properties are applied via type selectors (e.g. h1, div and body).*

*The style of your button elements might be set via two classes that you have given the class of:*

*.button — provides the button’s basic structure*

*.grey-btn — applies colors and other visual properties*

***BEM***

*Block, Element, Modifier is a CSS class-naming system devised. BEM complements OOCSS because OOCSS doesn’t impose any particular class-naming convention. In BEM terminology:*

* *A block is an independent, modular UI component. A block may be composed of multiple HTML elements, or even multiple blocks.*
* *An element is a component of a block. An element serves a singular purpose.*
* *A modifier is a CSS class that changes the default presentation of a block or element.*

*The BEM class-naming syntax:*

* *.block*
* *.block--modifier*
* *.block\_\_element*
* *.block\_\_element--modifier*

***SMACSS***

*Scalable and Modular Architecture for CSS , this CSS methodology is how we categorize our CSS style rules.It offers a simpler naming convention than BEM. There are no names for base styles because only type selectors (h1, p, a, etc.) are used for those. Modules are given their own unique class names. Sub-components and variations are prefixed with the name of their parent module.*

*Five categories:*

1. *Base*
2. *Layout*
3. *Module*
4. *State*
5. *Themes*

**Javascript - Part 3**

*.*

***Fetch API (Application Programming Interface)***

*The Fetch API is a modern interface that allows you to make HTTP requests to servers from web browsers. One of the main differences is that Fetch API uses Promise to deliver more flexible features to make requests to servers from the web browsers and provides a way to avoid callbacks and boilerplate heavy code that XMLHttpRequest(XHR) provides.*

*The fetch() method is available in the global scope that instructs the web browsers to send a request to a URL.*

***Promises***

*Represents the eventual completion (or failure) of an asynchronous operation and its resulting value.*

*Provides us a simpler alternative to executing, composing and managing asynchronous operation compared to the traditional callback-bases approach.*

*We have to be conscient about its 3 states:*

1. *When a Promise is* ***Pending****, it can transition to either Fulfilled or Rejected. Once a Promise transitions to either Fulfilled or Rejected, it cannot transition to any other state and its value will not change as well.*
2. *When a Promise is* ***Fulfilled****, this means the asynchronous operation has completed and the Promise has a value.*
3. *When a Promise is* ***Rejected****, this means the asynchronous operation has failed, and the Promise will never be fulfilled.*

***Ajax (XHR)***

*Ajax stands for Asynchronous Javascript and XML. Ajax is a programming technique that allows us to create dynamic, complex, and asynchronous web applications. Ajax allows us to send and receive data from the web server asynchronously without interfering with the current state or behavior of the web page or application.*

* *Read data from a web server - after the page has loaded*
* *Update a web page without reloading the page*
* *Send data to a web server - in the background*

*XHR is the XMLHttpRequest Object which interacts with the server. Ajax technique in the nutshell leverages the XHR request to send and receive data from the web server. This object is provided by the browser’s javascript environment. It transfers the data between the web browser and server.*

***How to send an XHR request?***

*First you have to know what you are going to need and where you are going to put it:*

*open(method, url[, async[, user[, password]]]) It initializes the request.*

*method request type such as GET,POST etc*

*url Request URL*

*async true or false*

*user Username for basic authentication*

*password Password for basic authentication*

*setRequestHeader(header,value) It sets the header for the HTTP request.*

*header name of header parameter*

*Value value of the parameter*

*onreadystatechange = callback() It is a EventHandler called when the readyState attribute changes.*

*readyState attribute It is an attribute that returns the current state of XMLHttpRequest object*

*status attribute It is an attribute that returns the status code to the HTTP XHR request.*

responseText attribute  *It is an attribute that returns the DOMstring response as the text.*

1. *Create a XMLHttpRequest object : var xhrobj = new XMLHttpRequest();*
2. *Send the request : xhrobj.open('GET','example.com/get');*

*xhrobj.send();*

1. *Post request (in case):*

*xhrobj.open("POST", 'example.com/post', true);*

*xhrobj.setRequestHeader("Content-type", "application/x-www-form-urlencoded");*

*xhrobj.send("username=john");*

1. *Receiving the response :*

*xhrobj.onreadystatechange = function () {*

*if (this.readyState == 4 && this.status == 200) {*

*document.getElementById("response").innerHTML = xhrobj.responseText;*

*}*

*}*

**Cookies**

*A cookie is information that a website puts on a user's computer. Cookies store limited information from a web browser session on a given website that can then be retrieved in the future.*

***Types of cookies***

* *HTTP cookies. This is the overall category of computer cookies used with modern web browsers to enable specific capabilities. All the cookies in this list -- except for flash cookies -- are forms of HTTP cookies.*
* *Session cookies. A session cookie is only persistent while the user is navigating or visiting a given website.*
* *Persistent cookies. Also sometimes referred to as permanent cookies, these persist for a configurable length of time or until a certain date that is set by the web server.*
* *First-party cookies. Also known as SameSite cookies, the cookie and information it contains is restricted to the same site on which it was set.*
* *Third-party cookies. These cookies are not restricted to the initial site where the cookie was created. Third-party cookies enable entities other than the original site to access them for user tracking and personalization purposes.*
* *Zombie cookies. This refers to a type of cookie that persists, even after the user attempts to delete it.*
* *Flash cookies. These are not browser or HTTP cookies but, rather, a specific type of cookie that works with Adobe Flash. With the decline in the use of Flash, these cookies are no longer widely used.*
* *Secure cookies. These are first- and third-party cookies that can only be sent over encrypted HTTPS connections.*

***Local storage - Session storage***

|  | *Localstorage* | *Sessionstorage* |
| --- | --- | --- |
| *Expires* | *No expire* | *Expire when the tab is closed* |
| *Storage* | *Browser only* | *Browser only* |
| *Clear/Delete* | *Javascript or manually* | *Javascript or automatic* |
| *Secured* | *No* | *No* |
| *Data transfer on HTTP request* | *No* | *No* |
| *Browser support* | *Lastest* | *Lastest* |
| *Data accessibility on Client side* | *Current tab* | *Window* |
| *Capacity* | *5-10 MB* | *5 MB* |
| *Accessible from* | *Any windows* | *Private to tab* |

**HTTP**

*HTTP stands for HyperText Transfer Protocol, is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests.*

***Methods***

*These methods correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other methods, too, but they are utilized less frequently.*

* *get: The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.*
* *head: The HEAD method asks for a response identical to a GET request, but without the response body.*
* *post: The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.*
* *put: The PUT method replaces all current representations of the target resource with the request payload.*
* *delete: The DELETE method deletes the specified resource.*
* *connect: The CONNECT method establishes a tunnel to the server identified by the target resource.*
* *options: The OPTIONS method describes the communication options for the target resource.*
* *trace: The TRACE method performs a message loop-back test along the path to the target resource.*
* *patch: The PATCH method applies partial modifications to a resource.*

***Response Codes***

1. *Informational responses (100–199),*
2. *Satisfactory responses (200–299),*
3. *Redirects (300–399),*
4. *Client errors (400–499),*
5. *Server errors (500–599).*

*Example :* [*https://http.cat/*](https://http.cat/)

***Session Management***

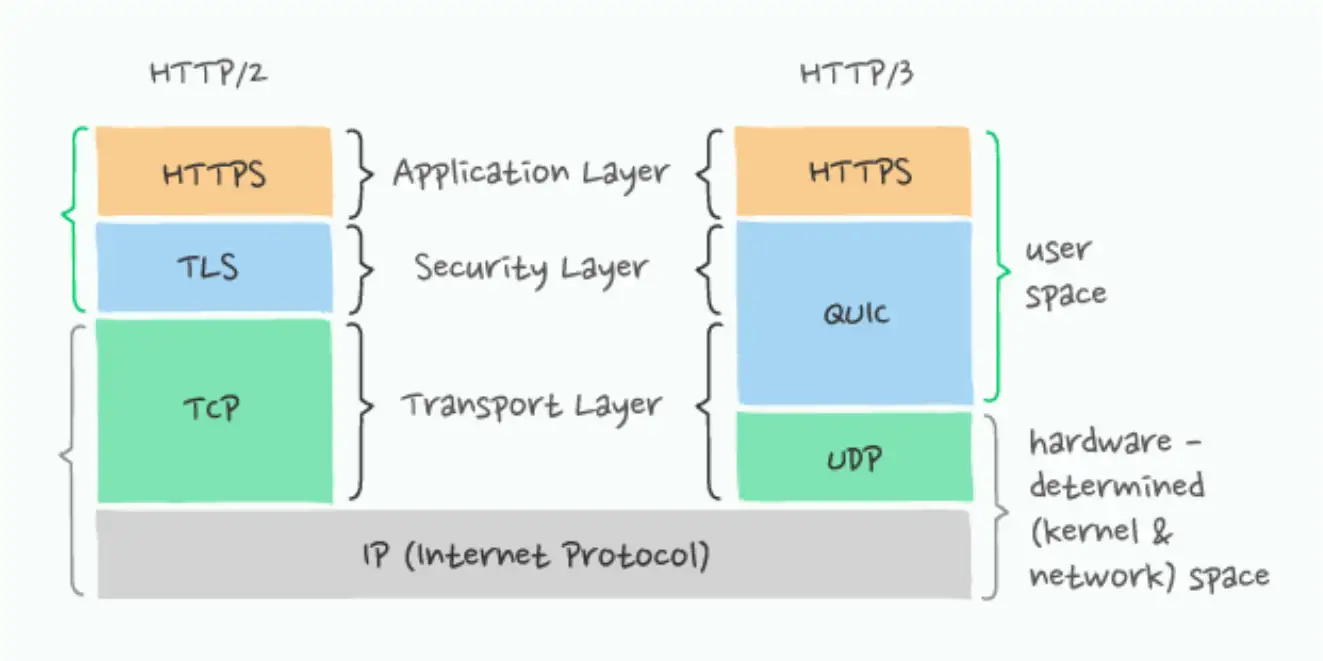
*Session management is used to facilitate secure interactions between a user and some service or application and applies to a sequence of requests and responses associated with that particular user. When a user has an ongoing session with a web application, they are submitting requests within their session and oftentimes are providing potentially sensitive information. The application may retain this information and/or track the status of the user during the session across multiple requests. More importantly, it is critical that the application has a means of protecting private data belonging to each unique user, especially within authenticated sessions.*

***HTTP/2 and HTTP/3***

***HTTP/2*** *modifies the data formatting, binary frame layer, and how the data is transported in the client-server architecture, while still abstracting the modification complexity for the existing applications.*

***HTTP/3*** *is to provide fast, reliable, and secure web connections across all forms of devices by resolving transport-related issues of HTTP/2. HTTP/3 uses a different transport layer network protocol called QUIC (Quick UDP Internet Connections), originally developed by Google, while HTTP/1.1 and HTTP/2 are mainly ‘HTTP-over-TCP’.*

*The main issue with TCP is that before establishing a session between a client and the server, a TLS handshake is needed to verify for a secure session. Another fundamental difference between HTTP/2 and HTTP/3 is that HTTP/3 runs over QUIC, and QUIC runs over connectionless UDP instead of the connection-oriented TCP (used by all previous versions of HTTP).*

**

***Https***

*Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP, which is the primary protocol used to send data between a web browser and a website. HTTPS is encrypted in order to increase security of data transfer. This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.*

*HTTPS is not a separate protocol from HTTP. Rather, it is a variant that uses Transport Layer Security (TLS)/Secure Sockets Layer (SSL) encryption over HTTP to secure communications. When a web server and web browser talk to each other over HTTPS, they engage in what's known as a handshake -- an exchange of TLS/SSL certificates -- to verify the provider's identity and protect the user and their data.*

***CORS***

*Cross-origin resource sharing (CORS) is a mechanism that allows browsers to request data from origins, which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). The CORS workflow starts when a script loaded from one origin attempts to make a request to another origin, allowing the server to tell the browser that it has permission or not to use an additional origin.*

***JSONP***

*JSONP stands for JSON with Padding, is a method for sending JSON data without worrying about cross-domain issues, and uses the <script> tag instead XMLHttpRequest.It is an API for the exchange of data that can be hosted on our server or on remote servers. It has become very popular because it is lightweight and compatible with most systems.*

*jSON and JSONP are basically the same thing, and the only thing that differs is their wrapper. In JSONP, instead of sending only the data, what is sent is a function, normally called a callback, which is like a JavaScript that includes the data that we have requested. Therefore, the name of JSONP is JSON with padding.*

***JSON Web Token***

*JSON Web Token is an open standard used to share information between two parties securely — a client and a server. In most cases, it’s an encoded JSON containing a set of claims and a signature. It’s usually used in the context of other authentication mechanisms like OAuth, OpenID to share user-related information. It’s also a popular way to authenticate/authorize users in a microservice architecture. JWT authentication is a token-based stateless authentication mechanism. JWTs can be encrypted, but they are typically encoded & signed.*

***SSO***

*Single sign-on (SSO) is an authentication method that enables users to securely authenticate by using just one set of credentials with multiple applications and websites regardless of the platform, technology, or domain the user is using.*

***OAuth 2.0***

*OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are being developed within the IETF OAuth Working Group.*

*OAuth 2 defines four types of authorization, each of which is useful in different cases:*

* *Authorization code: used with server-side applications*
* *Implicit: used with mobile applications or web applications (applications that run on the user's device)*
* *Resource Owner Password Credentials - Used with trusted applications, such as those belonging to the service*
* *Client Credentials - Used with Application API Access*

**Javascript - part 4**

*.*

***Events***

*The change in the state of an object is known as an Event. HTML events are "things" that happen to HTML elements, when JavaScript is used in HTML pages, JavaScript can "react" to these events, this process of reacting over the events is called Event Handling. An HTML event can be something the browser does, or something a user does, when events happen, you may want to do something. JavaScript lets you execute code when events are detected and handles the HTML events via Event Handlers. HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.*

*With single quotes:*

*<element event='some JavaScript'>*

*With double quotes:*

*<element event="some JavaScript">*

***Common HTML Events***

* *Onchange: An HTML element has been changed*
* *Onclick: The user clicks an HTML element*
* *Onmouseover: The user moves the mouse over an HTML element*
* *Onmouseout: The user moves the mouse away from an HTML element*
* *Onkeydown: The user pushes a keyboard key*
* *Onload: The browser has finished loading the page*

***Event bubbling***

*Event bubbling is a method of event propagation in the HTML DOM API when an event is in an element inside another element, and both elements have registered a handle to that event. It is a process that starts with the element that triggered the event and then bubbles up to the containing elements in the hierarchy. In event bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.*

*Example:*

*document.getElementById('mi\_boton\_1').addEventListener('click', () => console.log('boton 1 clicado!'))*

*document.getElementById('mi\_boton\_2').addEventListener('click', () => console.log('boton 2 clicado!'))*

*document.getElementById('mi\_div').addEventListener('click', () => console.log('div clicado!'))*

*If you press button 2, you will have an event from button 2, then one from button 1, and then another from the div, and this is the bubbling.*

**Bibliography**

* <https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development>
* <https://www.cprime.com/resources/what-is-agile-what-is-scrum/#:~:text=Agile%20software%20development%20refers%20to%20a%20group%20of%20software%20development,%2Dorganizing%20cross%2Dfunctional%20teams>.
* <https://www.scrum.org/resources/what-is-scrum>
* <https://lucidspark.com/blog/different-types-of-scrum-meetings>
* <https://www.agilealliance.org/glossary/xp/#q=~>(infinite~false~filters~(postType~(~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)
* <https://www.lucidchart.com/blog/what-is-extreme-programming>
* <https://www.freecodecamp.org/news/10-important-git-commands-that-every-developer-should-know/>
* <https://openwebinars.net/blog/que-es-git-y-para-que-sirve/>
* <https://linuxhint.com/add-tag-specific-commit-git/>
* <https://git-scm.com/docs/git-stash>
* <https://www.perforce.com/blog/vcs/git-beyond-basics-stashing-work-progress#:~:text=Using%20the%20git%20stash%20command,more%20easily%20switch%20between%20branches>.
* <https://www.atlassian.com/git/tutorials/git-hooks#:~:text=Git%20hooks%20are%20scripts%20that,in%20the%20development%20life%20cycle>.
* <https://medium.com/swlh/squash-and-rebase-git-basics-5cb1be1e0dac#:~:text=With%20%E2%80%9Csquash%E2%80%9D%2C%20you%20can,they're%20now%20just%201>.
* <https://betterprogramming.pub/differences-between-git-merge-and-rebase-and-why-you-should-care-ae41d96237b6#:~:text=Reading%20the%20official%20Git%20manual,it%20happened%2C%20rebase%20rewrites%20it%20>.
* <https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/>
* <https://www.geeksforgeeks.org/javascript-syntax/>
* <https://drive.google.com/file/d/11Qd_2a9YfHq7Yt4IGLXwWRs6OFpSu-6o/view>
* <https://www.w3schools.com/js/js_syntax.asp>
* <https://www.w3schools.com/js/js_comments.asp>
* <https://www.w3schools.com/js/js_variables.asp>
* <https://www.w3schools.com/js/js_functions.asp>
* <https://www.w3schools.com/js/js_arrays.asp>
* <https://www.w3schools.com/js/js_functions.asp>
* <https://www.javatpoint.com/what-is-html>
* <https://www.w3schools.com/html/html_elements.asp>
* <https://dmitripavlutin.com/javascript-scope/#2-block-scope>
* <https://www.w3schools.com/js/js_htmldom.asp#:~:text=The%20DOM%20defines%20a%20standard,and%20style%20of%20a%20document.%22>
* <https://www.w3schools.com/tags/ref_byfunc.asp>
* <https://www.javatpoint.com/html-tags>
* <https://www.w3schools.com/html/html_form_input_types.asp>
* <https://www.javatpoint.com/html-form-input-types>
* <https://www.codecademy.com/article/accessibility-and-html>
* <https://digitalmarketinginstitute.com/blog/what-is-seo>
* <https://www.w3.org/TR/REC-DOM-Level-1/introduction.html#:~:text=The%20Document%20Object%20Model%20>(DOM,document%20is%20accessed%20and%20manipulated.
* <https://www.w3schools.com/css/css_syntax.asp>
* <https://www.w3schools.com/css/css_selectors.asp>
* <https://www.w3schools.com/css/css_padding.asp>
* <https://www.w3schools.com/css/css_rwd_mediaqueries.asp>
* <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Positioning>
* <https://www.w3schools.com/css/css_positioning.asp>
* <https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction>
* <https://www.w3schools.com/whatis/whatis_bootstrap.asp>
* <https://materializecss.com/>
* <https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries>
* <https://www.w3schools.com/sass/sass_intro.php>
* <https://www.w3schools.com/sass/sass_variables.php>
* <https://www.webfx.com/blog/web-design/css-methodologies/#:~:text=CSS%20methodologies%20are%20formal%2C%20documented,massive%20lump%20of%20indivisible%20code>.
* <https://www.javascripttutorial.net/javascript-fetch-api/#:~:text=The%20Fetch%20API%20is%20a,is%20much%20simpler%20and%20cleaner>.
* <https://medium.com/@armando_amador/how-to-make-http-requests-using-fetch-api-and-promises-b0ca7370a444>
* <https://www.loginradius.com/blog/engineering/ajax-and-xhr-using-plain-javascript/#:~:text=XHR%20is%20the%20XMLHttpRequest%20Object,the%20web%20browser%20and%20server>.
* <https://www.techtarget.com/searchsoftwarequality/definition/cookie>
* <https://anexsoft.com/html-5-diferencias-y-ejemplos-entre-local-storage-y-session-storage>
* <https://developer.mozilla.org/en-US/docs/Web/HTTP>
* <https://www.w3schools.com/whatis/whatis_http.asp>
* <https://doc.oroinc.com/api/http-methods/#:~:text=The%20primary%20or%20most%20commonly,they%20are%20utilized%20less%20frequently>.
* <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>
* <https://http.cat/>
* <https://www.packetlabs.net/posts/session-management/>
* <https://www.section.io/engineering-education/http3-vs-http2/>
* <https://ably.com/topic/http-2-vs-http-3>
* <https://www.cloudflare.com/learning/ssl/what-is-https/>
* <https://www.techtarget.com/searchsoftwarequality/definition/HTTPS>
* <https://portswigger.net/web-security/cors>
* <https://www.stackhawk.com/blog/what-is-cors/>
* <https://youtu.be/4KHiSt0oLJ0>
* <https://web.dev/cross-origin-resource-sharing/>
* <https://www.w3schools.com/js/js_json_jsonp.asp>
* <https://www.arsys.es/blog/jsonp-api#:~:text=JSONP%20o%20JSON%20con%20padding,en%20la%20mayor%C3%ADa%20de%20sistemas>.
* <https://blog.miniorange.com/what-is-jwt-json-web-token-how-does-jwt-authentication-work/>
* <https://www.onelogin.com/learn/how-single-sign-on-works>
* <https://oauth.net/2/>
* <https://www.digitalocean.com/community/tutorials/una-introduccion-a-oauth-2-es>
* <https://www.w3schools.com/js/js_events.asp>
* <https://www.javatpoint.com/javascript-events>
* <https://www.geeksforgeeks.org/event-bubbling-in-javascript/>