

*Bienvenidos*

---



# Sobre Mi

---



David Luna

Venezolano, vivo en Costa Rica desde hace 3 años

Consultor, emprendedor, instructor con mas de 18 años de experiencia en el área de tecnología, especializado en la tecnología Java.

He tenido la oportunidad de trabajar en varias empresas de tecnología como Encava, EIS Consulting, Smartmatic, Cibersys e Inntec ocupando diferentes posiciones tanto como desarrollador y líder técnico, usando diferentes tecnologías como Java. PHP, Laravel, NodeJs, Angular, React, WebRTC, websockets, api rest, microservices, etc.

En los últimos 5 años he venido trabajando con Spring Boot y el desarrollo de Microservicios y API Rest, usando dockers, kubernetes y jenkins para el proceso de integración continua.

Tengo más de 15 años de experiencia docente, he sido profesor universitario en UNITEC y IUTV, así como instructor de diversos cursos de tecnología en Softrain.

[davidrlunag@gmail.com](mailto:davidrlunag@gmail.com)

+506 62265935

@davidrlunag

@david\_luna



- **Introducción al Desarrollo Web**
- **Introducción a Java Enterprise 7**
- **Conociendo Glassfish**
- **Java Server Faces**
- **Java Persistence API**
- **Enterprise Java Beans**
- **HTTP Endpoints**



- **Introducción a HTML5**
- **Introducción a CSS3**
- **Introducción a Javascript**
- **Introducción a GIT**



# Introducción a HTML5

HTML, siglas en inglés de **HyperText Markup Language** ('lenguaje de marcado de hipertexto'), hace referencia al **lenguaje de marcado** para la elaboración de **páginas web**. Es un estándar que sirve de referencia del software ya que, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, listas, entre otros

Es un estándar a cargo del **World Wide Web Consortium** (W3C) o Consorcio WWW



<https://caniuse.com/>

<https://html.spec.whatwg.org/>

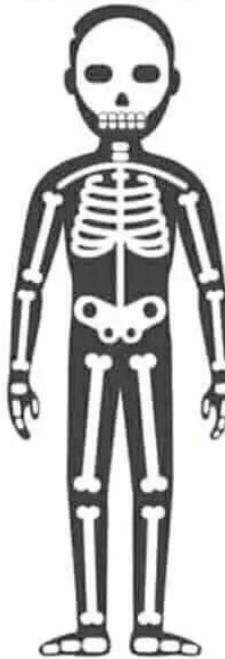
**WEB = HTML**



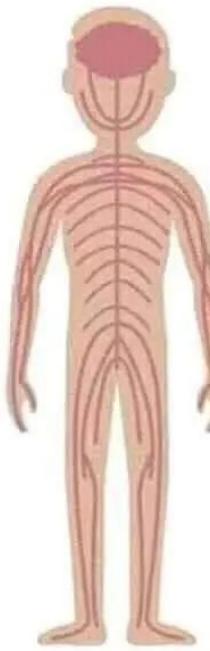
# Introducción a HTML5

---

HTML



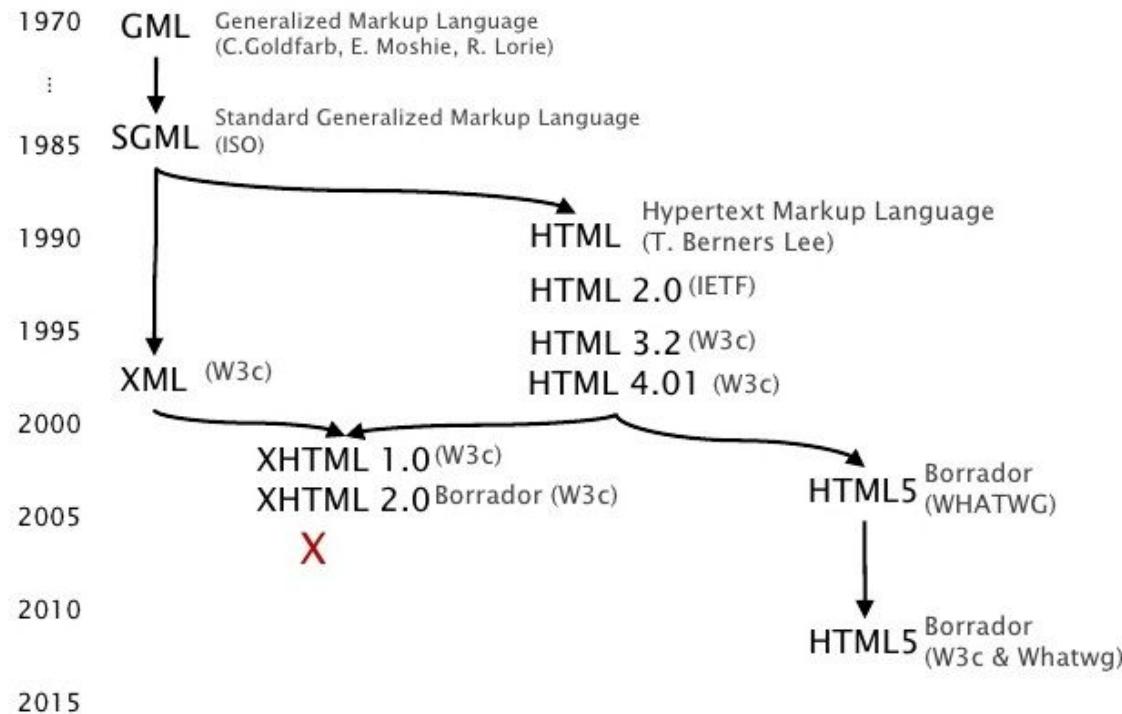
JS



CSS



## Evolución



# **Introducción a HTML5**

---

Estructura Básica de una pagina HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML CSS JS</title>
  </head>
  <body>
    <h1 id="welcome">HTML CSS JS</h1>
    <p>Welcome to HTML-CSS-JS.com</p>
    <p>Online HTML, CSS and JavaScript editor
       with instant preview.</p>
  </body>
</html>
```

# Introducción a HTML5

---



# Introducción a HTML5

## Etiquetado del documento

### DOCTYPE

XHTML 1.0

HTML5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD  
XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/x  
html1-strict.dtd">
```



```
<!DOCTYPE html>
```

## Etiquetado del documento

### SCRIPT

HTML 4.01

HTML5

```
<script type="text/javascript"  
src="file.js"> </script>
```

```
<script type="text/javascript"  
.....  
</script>
```



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



```
<script>  
.....  
</script>
```

## Etiquetado del documento

### META

HTML 4.01

HTML5

```
<meta http-equiv="Content-Type"  
content="text/html; charset=UTF-8">
```



```
<meta charset="UTF-8">
```

## Etiquetado del documento

### HOJAS DE ESTILO

HTML 4.01

HTML5

```
<link rel="stylesheet" type="text/css"  
href="estilos.css">
```



```
<link rel="stylesheet"  
href="estilos.css">
```

## Nuevas etiquetas de presentación

<div id="header">

<div id="menu">

<div>

<div>

<div>

<div id="footer">

<header>

<nav>

<article>

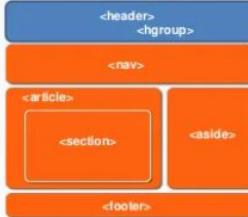
<section>

<aside>

<footer>

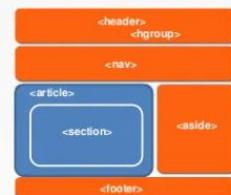
## Web Semántica

# Introducción a HTML5

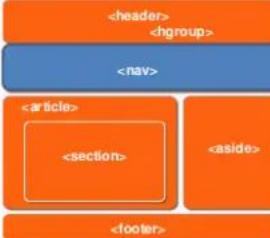


**<header>**  
representa la cabecera de un documento o sección  
**<hgroup>**  
representa el título de una sección. Se usa para agrupar conjuntos de elementos h1-h6 (títulos y subtítulos)

```
<header>
  <hgroup>
    <h1>Mi Blog</h1>
    <h2>Esforzándome para trabajar menos</h2>
  </hgroup>
</header>
```

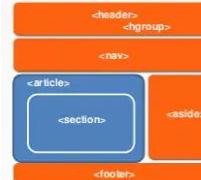


**<article>**  
representa una pieza de contenido **independiente** dentro de un documento  
**<section>**  
representa una sección del documento (un capítulo, un apartado, etc) **agrupa una serie de contenidos** con una temática común



**<nav>**  
representa una sección del documento que contiene navegación

```
<nav>
  <ul>
    <li><a href="#">home</a></li>
    <li><a href="#">blog</a></li>
    <li><a href="#">galeria</a></li>
    <li><a href="#">contacto</a></li>
  </ul>
</nav>
```

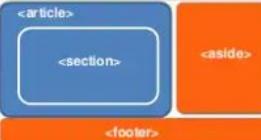


```
<article>
  <hgroup>
    <h1>Título del artículo</h1>
    <h2>Subtítulo del artículo</h2>
  </hgroup>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
  <section>
    <h3>Capítulo 1</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer bibendum scelerisque neque, ac facilisis neque</p>
  </section>
  <section>
    <h3>Capítulo 2</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer bibendum scelerisque neque, ac facilisis neque</p>
  </section>
</article>
```

# Introducción a HTML5

<header>  
<hgroup>

<nav>



<figure>

representa un diagrama, una ilustración, una fotografía, etc

<figcaption>

representa la "nota al pie" del elemento incluido en <figure>

<figure>

```

<figcaption> Javier González impartiendo seminarios sobre tecnologías web</figcaption>
```

<header>  
<hgroup>

<nav>



<footer>

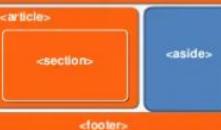
representa el pie de una sección o página. Suele contener información sobre el autor, copyright, etc

<footer>

```
<p>© 2012 Bla bla bla bla</p>
</footer>
```

<header>  
<hgroup>

<nav>



<aside>

representa contenidos que no están directamente relacionados con el resto de contenido de la página o que aporta información adicional

<article>

```
<header><h1>Tecnologías web</h1></header>
<p>bla bla bla</p>
<aside>
  <ul>
    <li><a href="#">Links sobre HTML5</a></li>
    <li><a href="#">Links sobre CSS3</a></li>
  </ul>
</aside>
</article>
```

CANVAS

```
<canvas id="miLienzo" width="360"
        height="240">
  <p>Tu navegador no soporta canvas</p>
</canvas>
<script>
var lienzo =
  document.getElementById('miLienzo');
var contexto = lienzo.getContext('2d');
</script>
```

Lienzo utilizado para representar imágenes, gráficos, dibujos y/o elementos visuales "al vuelo" con Javascript

No requiere plugins, ni codecs

Mapa de bits (no hay reescalado)

El contenido no se añade al DOM

Puede ser exportado



<http://www.whatwg.org/specs/web-apps/current-work/#2dcontext>

# Introducción a HTML5

## VIDEO

The diagram illustrates various attributes for the `<video>` element. It includes icons for each attribute: a clapperboard for `width & height`, a movie poster for `poster`, a clapperboard with a play button for `controls`, a clapperboard with a circular arrow for `loop`, a clapperboard with a `SRC` icon for `source`, and a clapperboard with a floppy disk for `preload`. A film strip icon is also present.



## FORMULARIOS

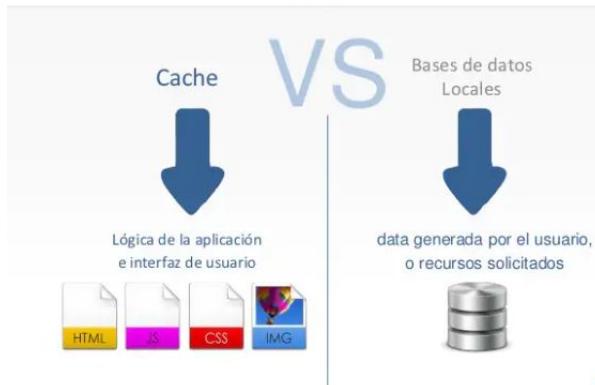
This diagram shows the `<input>` element with various type attributes and their associated features. It includes a list of types: search, Email, phone, url, tel, range (\*), number (\*), date, datetime, datetime-local, month, colour. It also shows examples of dropdown menus, a date picker, and a color picker. Below the diagram is a link: <http://www.findmebyip.com/litmus/#html5-forms-inputs>.

## FORMULARIOS

This diagram shows more advanced attributes for the `<input>` element. It includes: `placeholder` (with a placeholder example), `autofocus` (with a focused input example), `required` (with a red border example), `autocomplete` (with a dropdown menu example), `pattern` (with a regular expression example), and `on` and `off` (with a switch example). Below the diagram is a link: <http://www.findmebyip.com/litmus/#html5-forms-inputs>.

# Introducción a HTML5

## ELEMENTOS QUE DESAPARECEN



## DRAG & DROP

**draggable (true|false):** el elemento puede ser arrastrado hacia otro elemento

### EVENTOS

- dragstart
- drag
- dragenter
- dragover
- drop
- dragend
- dragleave



## ALMACENAMIENTO LOCAL

## Técnicas de Almacenamiento con HTML5

### Diferentes APIs:

- Web Storage (Local Storage or DOM Storage)
- Web SQL Database
- IndexedDB
- File Storage

### PRINCIPIOS:

- Normas **estándar** para "todos" los navegadores.
- Información **solo accesible desde el propio navegador**.
- Interacción de la API y la Base de Datos es **asíncrona**

# Introducción a HTML5

## WebSockets

Permite la comunicación bidireccional con cualquier servidor mediante un determinado protocolo de red.

La conexión con el servidor se establece de forma **asíncrona**, en segundo plano, y la gestión del todo su ciclo de vida se realiza a través de **callbacks** que reciben **eventos**.

```
<script>
  var ws = new WebSocket("ws://echo.websocket.org");
</script>
```

Debemos utilizar "ws://" para establecer conexiones con el protocolo websocket.

El constructor admite además un parámetro adicional para que indicar un conjunto de subprotocolos, pero aún está sin definir.

## WEB WORKERS

Pueden procesar eventos, callbacks, e incluso es posible crear otros workers.

Limitación: no tienen un contexto de navegación asociado.

No pueden acceder al DOM, window, document o parent, Pero sí a navigator, location, XMLHttpRequest, timers, applicationCache o Web SQL database.

Permiten la posibilidad de ejecutar el código de otros scripts

```
<script>
  importScripts("script1.js"); //De uno en uno...
  importScripts("script2.js");
  importScripts("script3.js", "script4.js"); //.. o varios a la vez
</script>
```

## WEB WORKERS

Tareas JavaScript que pueden lanzarse en segundo plano, a modo de threads.

Su objetivo es permitir que las aplicaciones web puedan lanzar hilos de ejecución concurrentes con una gran carga de trabajo y duración indeterminada.

Las tareas funcionan al margen del proceso normal de gestión de eventos de los controles de la interface de usuario, evitando bloquear la página durante su ejecución.

```
<script>
  var worker = new Worker("worker.js");
</script>
```

```
<button type="button" onclick="worker.terminate();"> Kill</button>
```

<https://testdrive-archive.azurewebsites.net/GraphIcs/WorkerFountains/Default.html>

---

## El validador

¿Cómo sé que lo estoy haciendo bien?

<http://validator.w3.org>

Nos asegura que:

- Los tags están bien tipeados
- Están correctamente anidados
- No falta ningún elemento requerido
- No hay errores de sintaxis

# **Introducción a HTML5**

---

<https://www.w3schools.com/html/default.asp>

<http://desarrolloweb.dlsi.ua.es/libros/html-css/ejercicios>

<https://cloud.netlifyusercontent.com/assets/344dbf88-fdf9-42bb-adb4-46f01eedd629/06ffd509-ac92-4b81-bfa0-9c3a34b3ab35/html5-cheat-sheet.pdf>

[https://www.w3schools.com/tryit/tryit.asp?filename=tryhtml\\_hello](https://www.w3schools.com/tryit/tryit.asp?filename=tryhtml_hello)

<https://vscode.dev/>

<https://platform.html5.org/>

<https://www.w3.org/TR/2012/CR-html5-20121217/>

<https://developers.google.com/web>

<https://developer.mozilla.org/es/docs/Web/HTML>

**CSS 3**



# Introducción a CSS3

## HISTORIA DE CSS 3

- 1996      **CSS 1:** permite dar estilos independientemente del navegador y del HTML
- 1998      **CSS2:** nuevas funcionalidades, pero implementación lenta  
**Semilla del CSS3.**
- Se plantea una [lista de mejoras de CSS2](#)
- 2000      Borrador de CSS3
- 2002      **CSS2.1:** Crea lo que ahora consideramos el estándar
- 2005      **Empieza el desarrollo** de CSS3
- 2009      **Implementación** en algunos navegadores de algunas partes de CSS3

## CSS 3 VS CSS 2

-  Mejora en los selectores
-  Nuevos estilos  
Sombra  
Opacidad  
esquinas redondeadas  
...
-  Mejora en tipografías
-  Transformaciones

---

-  Reduce la cantidad de HTML (divitis)
-  Reduce las peticiones de imágenes
-  ¡Nuestra pagina va a ser más rápida!

# Introducción a CSS3

## Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

## Internal CSS

```
<head>
  <style type = text/css>
    body {background-color: blue;}
    p { color: yellow;}
  </style>
</head>
```

## External CSS

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



## Incluir el CSS

### Elemento link

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

### Print CSS

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

### Media queries - “responsive design”

```
<link rel="stylesheet" href="css/720.css"  
media="screen and (min-width: 720px)" />
```

### Class

Class me permite agrupar por tipo.

```
.button {  
    color: #FFFFFF;  
    background: #637580;  
}  
    
```

```
<a class="button">Ver posts anteriores</a>
```



```
h1 {  
    color: #249999;  
}  
    
```

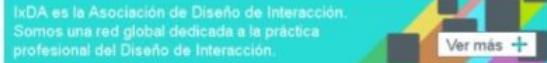
```
a {  
    color: #0071BC;  
}  
    
```

A screenshot of a Mozilla Firefox browser window. The address bar shows the URL 'http://dev-cvam.cc'. The main content area displays the 'IxDA Buenos Aires' website. The page includes a header with the IXDA logo, a main content section with text and images, and a footer with copyright information and publication dates. The styling of the page, including colors and fonts, is consistent with the examples shown in the previous slides, demonstrating the application of CSS rules like h1 and a.

### ID

Un elemento que es único en la página.

```
#intro {  
    color: #FFFFFF;  
    background: #28DBD5;  
}  
    
```



```
<p id="intro">IxDA es la Asociación de Diseño de Interacción.  
Somos una red global dedicada a la práctica profesional...  
    <a href="info.html">Ver más</a>  

```

# Introducción a CSS3

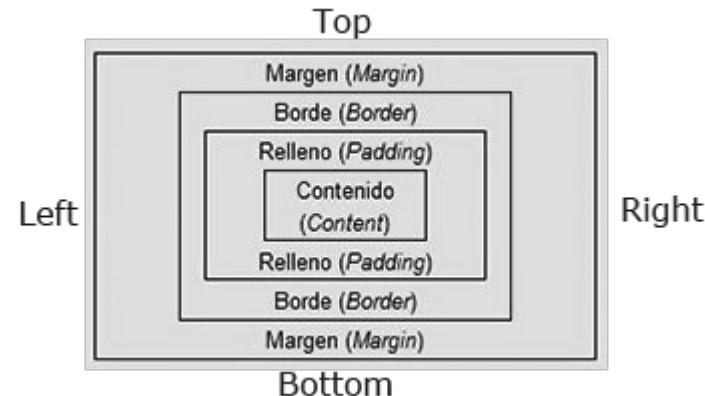
## Múltiples selectores

Una regla puede tener más de un selector.

```
.button {  
background: #586875;  
}  
  
#footer {  
background: #586875;  
}
```

```
.button,  
#footer {  
background: #586875;  
}
```

separados por comas



# Introducción a CSS3

## BORDES

### border-radius

border-radius: 15px;



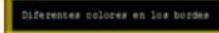
### border-image

border-image: url(border.png)  
27 27 27 27 round round;



### border-color

border: 5px solid #000;  
border-colors: #e00 #c30 #c50 #c60 #c70



### box-shadow / text-shadow

box-shadow: 10px 10px 5px #888;



## MULTIPLE BACKGROUNDS

background:

```
url(..../topImage.jpg) top left no-repeat,  
url(..../centerImage.jpg) top right repeat-y,  
url(..../bottomImage.jpg) bottom center no-repeat;
```



<http://www.css3.info/wp-content/uploads/2007/09/multiple-backgrounds-example.html>

## TRANSFORM

### rotate :

transform: rotate(30deg);



<http://www.ejhansel.com/transform/>

### skew :

transform: skew(-30deg);



### translate

transform: translate(30px,10px);



### scale :

transform: scale(0.5,2.0);



<http://www.the-art-of-web.com/css/css3-animation/>

## COLOR

### Opacity

Opacity: 1.0

Opacity: 0.5

### HSL:

(Hue, Saturation, Lightness)

hsl(21,97%,52%)



### RGBA:

(Red, Green, Blue, Alpha)

rgb(255,192,0,1);

rgb(255,192,0,0.5);

### HSLA:

(Hue, Saturation, Lightness, Alpha)

hsla(21,97%,52%,1);

hsla(21,97%,52%,0.5);

# Introducción a CSS3

## MULTI-COLUMN LAYOUT

- column-count
- column-width
- column-gap
- column-rule



## WEBFONTS

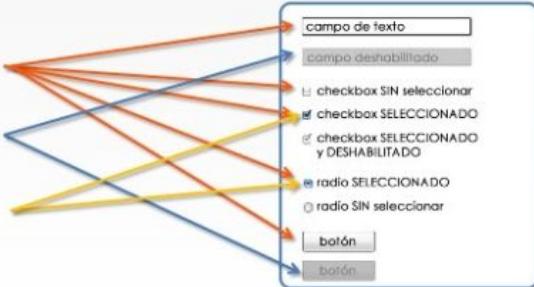
HTML 5 FONTS  
**HTML 5 FONTS**  
**HTML 5**  
HTML 5 FONTS  
HTML 5 FONTS  
**HTML 5 FONTS**  
**HTML 5 FONTS**  
Html 5 Fonts  
**HTML 5 FONTS**  
**HTML 5 FONTS**

```
@font-face {  
    font-family: 'FontName';  
    src: url('Gondola_SD-webfont.eot');  
    src: local(' '), url('FontName.woff')  
         format('woff'), url('FontName.ttf')  
         format('truetype'),  
         url('FontName.svg#webfontsgM4b18D')  
format('svg');  
    font-weight: normal;  
    font-style: normal;  
}  
  
div {  
    font-family: FontName;  
}
```

 <http://www.fontsquirrel.com/fontface/generator>

## NUEVAS PSEUDO-CLASES

- :enabled
- :disabled
- :checked



## TEXTOS

### Text-shadow

Text-shadow: Xpos Ypos Blur Color;

#### ejemplo de sombra

 <http://lab.simurai.com/flashlight>

### Word-wrap

word-wrap: break-word;

This paragraph has long words  
thisisaveryverylongwordthatisntreal  
yoneword and again a  
longwordwithnospacesinit

### Text-overflow

Text-overflow: ellipsis-word;

Lore ipsum dolor sit...



GEEK MONKEY TECH



# Introducción a CSS3

## SELECTORES DE ATRIBUTOS

[att\*=val]

contiene val

[att^=val]

empieza por val

[att\$=val]

termina por val

a[href^="http://web"]



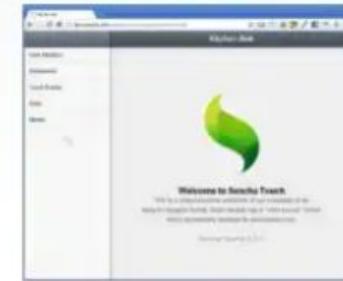
a[href\*=".es"]

a[href\$=".pdf"]

## MEDIA QUERIES

**min-width & max-width**

diferentes estilos según el tamaño de la pantalla



<http://mediaqueri.es/>

# *Introducción a CSS3*

---

## **Recursos**

<https://www.css3.info/preview/>

<https://cloud.netlifyusercontent.com/assets/344dbf88-fdf9-42bb-adb4-46f01eedd629/d7fb67af-5180-463d-b58a-bfd4a220d5d0/css3-cheat-sheet.pdf>

<https://www.w3schools.com/css/default.asp>



# JavaScript

# Introducción a Javascript

**JAVASCRIPT**



{ Es un lenguaje de programación que añade dinamismo e interactividad a una página Web. }



JavaScript es el único lenguaje de programación que funciona en los navegadores de forma nativa (lenguaje interpretado sin necesidad de compilación). Por tanto se utiliza como complemento de HTML y CSS para crear páginas webs.

## HISTORIA DE JAVASCRIPT

Javascript es un lenguaje que fue creado por Brendan Eich con el nombre de mocha, mientras se encontraba trabajando en Netscape, compañía propietaria de un navegador con el mismo nombre hoy desaparecido, no pasaría mucho tiempo para que luego el lenguaje fuera renombrado a LiveScript. Su lanzamiento se produjo en 1995. En 1997 el ECMA, organismo internacional creó la primera edición de la norma ECMAScript, que inicio a normalizar el lenguaje. Desde ese momento JavaScript se convirtió en el lenguaje más usado. Su autor lo creó en el tiempo de una semana.



# Introducción a Javascript

## CARACTERÍSTICAS DE JAVASCRIPT



**Javascript es un lenguaje de tipado dinámico.**  
Es un lenguaje que nos da la facilidad de no definir los tipos de datos al declarar una variable, ya que lo asigna durante el tiempo de ejecución basado en su valor en ese momento..

### Javascript es un lenguaje basado en prototipos.

Los prototipos son un mecanismo por el cual un objeto hereda propiedades y métodos de un padre, en Javascript la herencia funciona por prototipos. Un objeto en JavaScript tiene otro objeto, llamado prototype, cuando pedimos a un objeto una propiedad que no tiene, la busca en su prototipo. Entonces, un prototipo es otro objeto que se utiliza como una fuente de propiedades alternativa.



**Javascript es un lenguaje multiparadigma.**  
JavaScript es un lenguaje multiparadigma y, por tanto, podemos usarlo de distintas formas para desarrollar las aplicaciones:



**Programación imperativa:** se escriben las instrucciones paso a paso para generar la solución al problema.



**Programación funcional:** usando funciones que se pueden ir anidando hasta conseguir el resultado que se espera.



**Programación Orientada a Objetos:** se usan objetos que encapsulan variables y métodos, para interactuar con otros objetos, y de esta forma se intenta obtener el resultado esperado.

# Introducción a Javascript

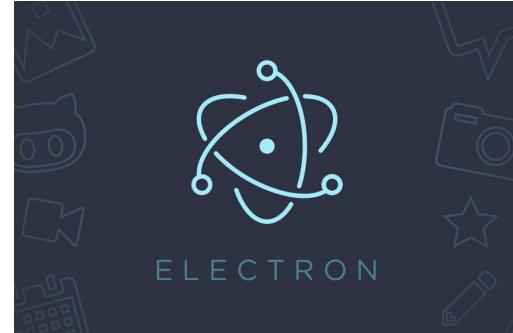
Por qué aprender Javascript?



Frontend / PWA



Backend / IOT

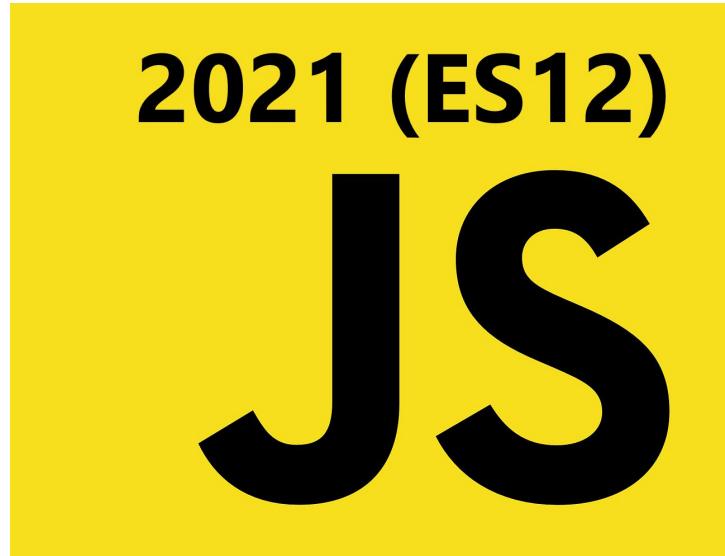
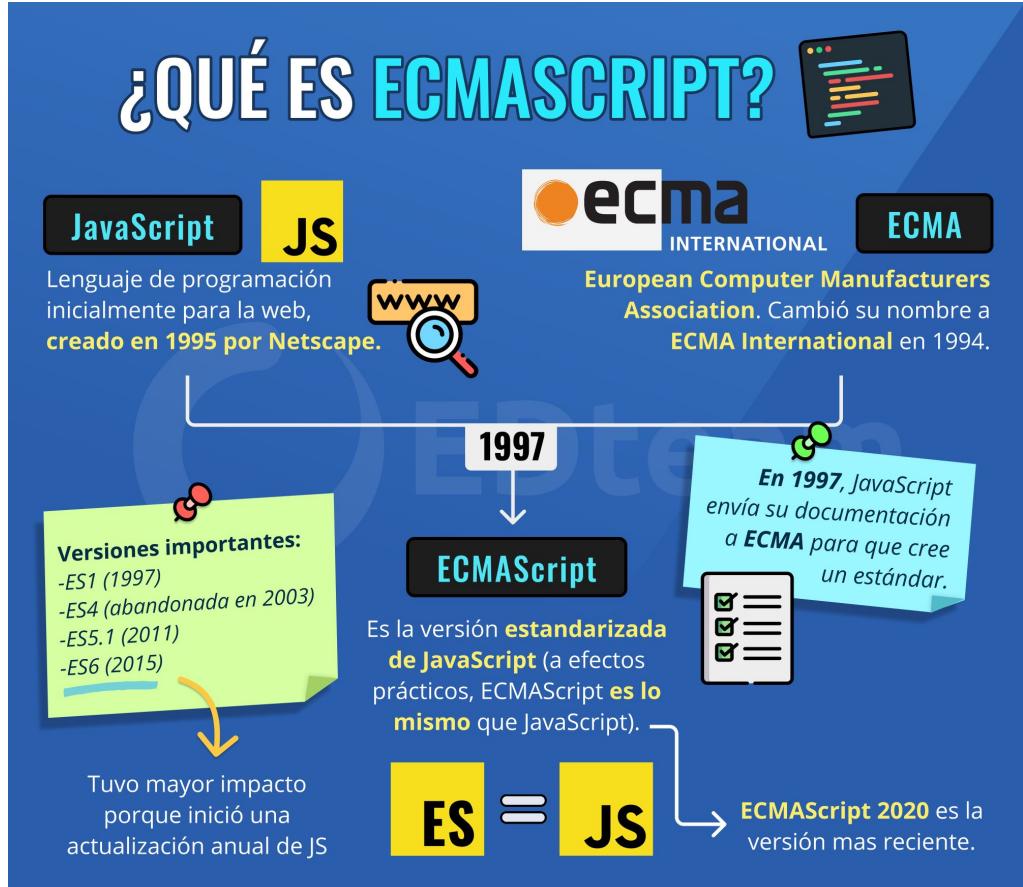


Desktop

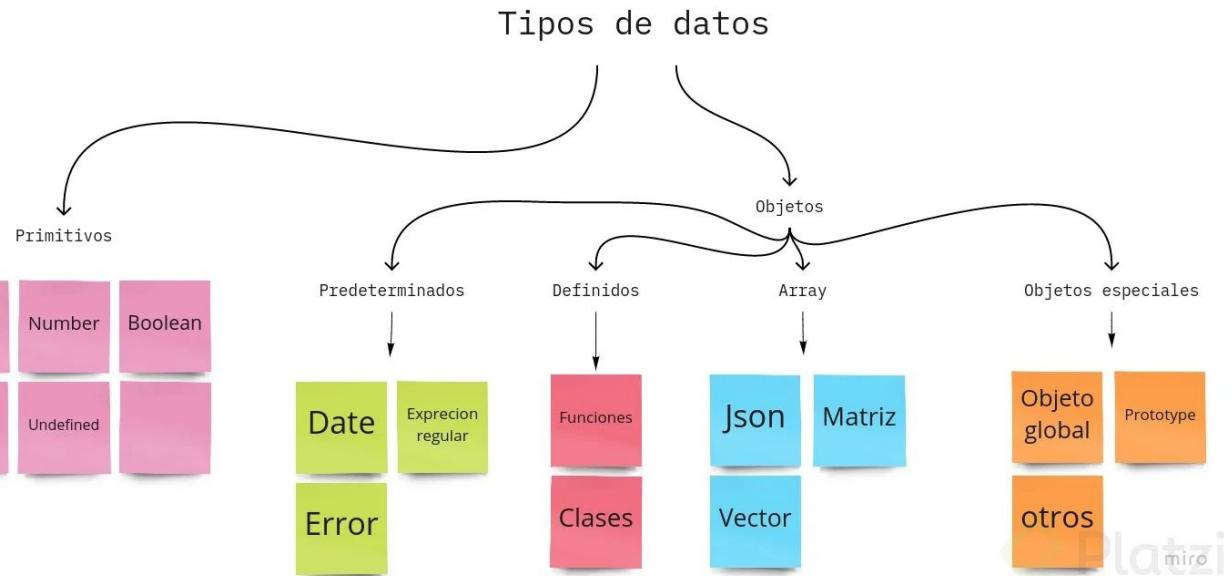


Mobile

# Introducción a Javascript



# Introducción a Javascript



```
> 40
< 40
> typeof 40
< "number"
> typeof ""
< "string"
> typeof "Jheyson"
< "string"
> typeof Jheyson
< "undefined"
> typeof true
< "boolean"
> typeof false
< "boolean"
> typeof null
< "object"
> typeof undefined
< "undefined"
> typeof []
< "object"
> typeof {}
< "object"
```

# Introducción a Javascript



# Introducción a Javascript

## var, let, const

DIFERENCIA EN DECLARAR VARIABLES

### Var

Es la manera más antigua de declarar variables. Dichas variables tienen scope (alcance) local, y pueden modificar su valor. También nos permite utilizar un comportamiento llamado hoisting, sin generar error.

Hoisting: independientemente de donde se declare la variable, ésta es movida al inicio del ámbito al que pertenece.

### let

El alcance se va a reducir al bloque (block scope) donde la variable es declarada y utilizada. Esta variable podrá cambiar su valor pero no podrá ser declarada en ninguna otra parte de la función.

### const

El alcance se reduce al bloque donde es utilizada. Las variables declaradas con const no podrán modificar su valor ni ser declaradas nuevamente.

	BLOCK SCOPED	TDZ	CREATES GLOBAL PROPERTY	REASSIGNABLE	REDECLARABLE
var	✗	✗	✓	✓	✓
let	✓	✓	✗	✓	✗
const	✓	✓	✗	✗	✗

# Introducción a Javascript

## ¿QUÉ ES UNA FUNCIÓN?

Una función es un trozo de código reutilizable en el que hay un conjunto de instrucciones, este código solo se ejecuta cuando se llama a dicha función.



### DECLARACIÓN

#### 1 DECLARAR

- \* Palabra reservada `function`
- \* **Nombre** de la función
- \* Escribir los **Parámetros**

#### 2 BLOQUE DE CÓDIGO

- \* Se escribe el conjunto de instrucciones que va a realizar la función

#### 3 INVOCAR FUNCIÓN

- \* Llamar a la función
- \* Escribir los argumentos (valores)

```
// Function declaration //
function someName(param1, param2) { ①
    // bunch of code as needed...
    var a = param1 + "love" + param2;
    return a;
} ②

// Invoke (run / call) a function
someName("Me", "You") ③
```

mGuedez  
tzi

```
// Declarativas

function miFuncion() {
    return 3;
}

//Expresión

var miFuncion = function(){
    return a + b;
}
```

# Introducción a Javascript

## SCOPE

### 1. ¿QUÉ ES EL SCOPE?

- \* Es el alcance que tienen las variables
- \* Según el lugar donde declaras e inicializas una variable dependerá si puedes o no acceder a ella.

### 2. TIPOS DE SCOPE

- \* Global: el todo

→ aquí van a existir todas las variables, validaciones, funciones que se tengan.

- \* Local: pequeño universo dentro del scope global.

→ al inicializar una función, en ese momento, dentro de la función se va a crear el scope local.

→ Solo lo que está dentro de la función va a tener acceso a eso mismo.

→ Sin embargo, desde dentro de una función podemos acceder a una variable declarada en el scope global, pero no en viceversa.

```
var nombre = "Diego" → scope global.  
function fun() {  
    var apellido = "De Grandia"  
    return nombre + " " + apellido  
}  
scope ←  
local
```

```
console.log(apellido) // apellido is not defined
```



@ANDRECONTI

## Hoisting en variables

El interprete de JavaScript reserva un lugar en memoria para las variables declaradas con var antes de que el código se ejecute, y esto causa que si se utiliza una variable antes de su declaración ésta tenga el valor undefined. Éste fenómeno se identifica como hoisting.

## Coerción

La coerción es la conversión automática o implícita de valores de un tipo de dato a otro (Ejemplo: de cadena de texto a número).

Existen dos tipos de coerción:

### Coerción implícita

Es cuando el lenguaje nos ayuda a cambiar el tipo de valor.

Ejemplos:

```
4 + "7";      /* 47 */  
4 * "7";      /* 28 */
```

### Coerción explícita:

Es cuando obligamos a que cambie el tipo de valor.

Ejemplos:

```
String(10);  
Number("10")  
ParseInt("10");
```

# Introducción a Javascript

OPERADORES LÓGICOS Y RELACIONALES	DESCRIPCIÓN	EJEMPLO
<code>==</code>	Es igual	<code>a == b</code>
<code>===</code>	Es estrictamente igual	<code>a === b</code>
<code>!=</code>	Es distinto	<code>a != b</code>
<code>!==</code>	Es estrictamente distinto	<code>a !== b</code>
<code>&lt;, &lt;=, &gt;, &gt;=</code>	Menor, menor o igual, mayor, mayor o igual	<code>a &lt;=b</code>
<code>&amp;&amp;</code>	Operador and (y)	<code>a &amp;&amp; b</code>
<code>  </code>	Operador or (o)	<code>a    b</code>
<code>!</code>	Operador not (no)	<code>!a</code>

Operadores lógicos y relacionales básicos en JavaScript

Operador	Operación	Valor x	Valor y	Resultado	Equivale a
<code>=</code>	<code>x = y</code>	10	5	<code>x = 5</code>	
<code>+=</code>	<code>x += y</code>	10	5	<code>x = 15</code>	<code>x = x + y</code>
<code>-=</code>	<code>x -= y</code>	10	5	<code>x = 5</code>	<code>x = x - y</code>
<code>*=</code>	<code>x *= y</code>	10	5	<code>x = 50</code>	<code>x = x * y</code>
<code>/=</code>	<code>x /= y</code>	10	5	<code>x = 2</code>	<code>x = x / y</code>
<code>%=</code>	<code>x %=</code>	10	5	<code>x = 0</code>	<code>x = x % y</code>

# Introducción a Javascript

```
> function getRandomNumber(number) {
    return Math.floor(Math.random()*number);
}

var opciones = ["piedra", "papel", "tijera"]

function juego(opcion) {
    var number=getRandomNumber(3);
    console.log(number);
    var opcionPC = opciones[number];
    console.log(opcionPC);
    if(opcion==opcionPC){
        return "Empate";
    }else if(opcion==="piedra"){
        if(opcionPC==="tijera"){
            return opcionPC + "=> Gana el jugador";
        } else{
            return opcionPC + "=> Gana PC";
        }
    }else if(opcion==="papel"){
        if(opcionPC==="piedra"){
            return opcionPC + "=> Gana el jugador";
        } else{
            return opcionPC + "=> Gana PC";
        }
    }else if(opcion === "tijera"){
        if(opcionPC==="papel"){
            return opcionPC + "=> Gana el jugador";
        } else{
            return opcionPC + "=> Gana PC";
        }
    }else{
        return "Opción no válida";
    }
}

var opcion = "piedra";
< undefined
> juego(opcion)
< "tijera=> Gana el jugador"
```

```
var piedra = "piedra";
var papel = "papel";
var tijeras = "tijeras";

var mensaje = prompt("Juego de Piedra, Papel o Tijeras\nCon cual deseas jugar\n\nPiedra\n\nPapel\n\nTijeras");
var miRespuesta = mensaje.toLowerCase();

analiza(tijeras);

function analiza(respuesta) {
    if (respuesta == piedra && miRespuesta == piedra) {
        console.log("Respuesta del computador: " + respuesta + " Intenta de nuevo");
    } else if (respuesta == papel && miRespuesta == papel) {
        console.log("Respuesta del computador: " + respuesta + " Intenta de nuevo");
    } else if (respuesta == tijeras && miRespuesta == tijeras) {
        console.log("Respuesta del computador: " + respuesta + " Intenta de nuevo");
    } else if (miRespuesta == piedra && respuesta == tijeras) {
        console.log("Gane yo");
    } else if (miRespuesta == papel && respuesta == piedra) {
        console.log("Gane yo");
    } else if (miRespuesta == tijeras && respuesta == papel) {
        console.log("Gane yo");
    } else if (miRespuesta == piedra && respuesta == papel) {
        console.log("Gano el computador");
    } else if (miRespuesta == papel && respuesta == tijeras) {
        console.log("Gano el computador");
    } else if (miRespuesta == tijeras && respuesta == piedra) {
        console.log("Gano el computador");
    } else {
        console.log("No se pudo procesar\nIntenta de Nuevo");
    }
}
```



# Introducción a Javascript

```
1 // Piedra, papel o tijeras
2
3 /*las variables no son necesarias?
4 var piedra = 1;
5 var papel = 2;
6 var tijeras = 3;*/
7
8 // random choice
9 var r = function(){
10     var random = Math.floor((Math.random() * 3) + 1);
11     return random;
12 }
13
14 var play = function(yo,pc){
15     switch(yo + "," + pc){
16
17         case '1,2':
18             console.log("PC: papel cubre a la piedra")
19             break;
20         case '1,3':
21             console.log("Gane: piedra rompe tijeras")
22             break;
23
24         case '2,1':
25             console.log("Gane: papel cubre a la piedra")
26             break;
27         case '2,3':
28             console.log("PC: tijeras corta papel")
29             break;
30
31         case '3,1':
32             console.log("PC: piedra rompe tijeras")
33             break;
34         case '3,2':
35             console.log("Gane: tijeras corta papel")
36             break;
37
38         default:
39             console.log("Empate")
40     }
41 }
42
43 play(1,r()) // cual sera el resultado?
```

```
var numero = 1;

switch (numero) {
    case 1:
        console.log("Soy uno!");
        break;
    case 10:
        console.log("Soy un 10!");
        break;
    case 100:
        console.log("Soy un 100!");
        break;
    default:
        console.log("No soy nada"); |
```

# Introducción a Javascript

## Arrays

### Métodos para mutar el contenido del Array

**.push:** Añade elementos al final del Array

**.pop:** Elimina el elemento al final del Array

**.unshift:** Agrega un elemento al inicio del Array

**.shift:** Elimina el elemento al inicio del Array

**.indexOf:** Nos busca la posición en la que se encuentra el elemento dentro del Array

Son estructuras de datos. Es un valor que puede guardar múltiples valores adentro. Incluso pueden haber varios arrays dentro de arrays

```
> var frutas = ["Manzana", "Plátano", "Cereza", "Fresa"]  
console.log(frutas)  
► (4) ["Manzana", "Plátano", "Cereza", "Fresa"]  
< undefined  
> console.log(frutas.length)  
4  
< undefined  
> console.log(frutas[3]);  
Fresa  
< undefined  
> console.log(frutas[0])  
Manzana
```

```
> var frutas = ["Manzana", "Plátano", "Cereza", "Fresa"]  
console.log(frutas)
```

```
► (4) ["Manzana", "Plátano", "Cereza", "Fresa"]
```

```
< undefined
```

```
> console.log(frutas.length)
```

```
4
```

```
< undefined
```

```
> console.log(frutas[3]);
```

```
Fresa
```

```
< undefined
```

```
> console.log(frutas[0])
```

```
Manzana
```

¿Cómo acceder a la cantidad de elementos de un Array?

¿Cómo acceder a uno de los elementos del Array?



# Introducción a Javascript

## Topic: Loops

### Recall

#objetos iterables:

Array, Map, Set,  
Object, etc.

#keys:

Los Objects están  
constituidos por keys  
(propiedades) y  
values (valores):

```
var Object = {  
    key: 'value'  
};  
var User = {  
    name: "Luis"  
};
```

### Notes

#### for... in VS for... of

Ambas declaraciones iteran sobre objetos iterables; aunque los valores retornados son diferentes.

<· **for... in**

Devuelve una lista de las keys del objeto iterado.

```
// for... in statement example  
var list = [4, 5, 6];  
  
for (let i in list) {  
    console.log(i); // "0", "1", "2"  
}  
  
/*  
keys:      0 1 2  
var array = [4, 5, 6];  
*/
```

<· **for... of**

Devuelve una lista de los valores de las propiedades del objeto iterado.

```
// for... of statement example  
var list = [4, 5, 6];  
  
for (let i of list) {  
    console.log(i); // "4", "5", "6"  
}
```





<https://github.com/getify/You-Dont-Know-JS>

<https://jsconsole.com/>

<https://jsfiddle.net/>

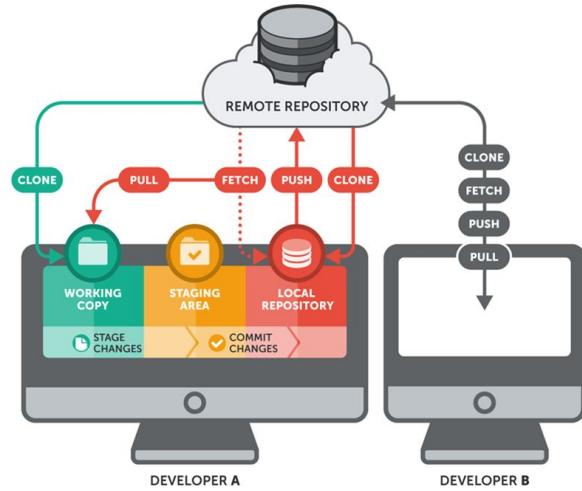
<https://html-css-js.com/>



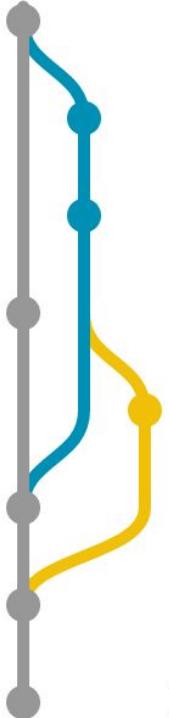
# Introducción a Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. En su lugar GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. GitHub sería la red social de código para los programadores, tu propio curriculum vitae.

Guarda solo los cambios que se hagan en un archivo y permite registrar quien hace el cambio.



# Introducción a Git



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] eaf618c New cool feature - John Feature

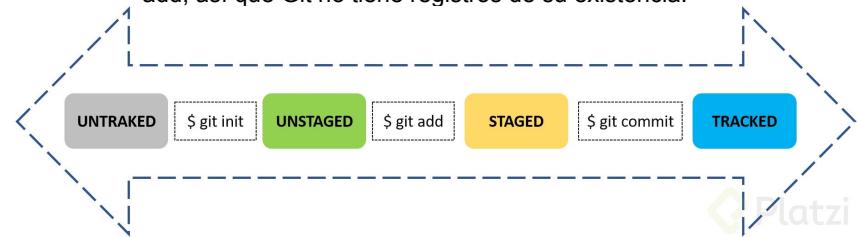
[master] 8f1e0e7 Merge branch 'develop' into 'master' - John Doe

[master] 6a3dacc Merge branch 'myfeature' into 'master' - John Doe

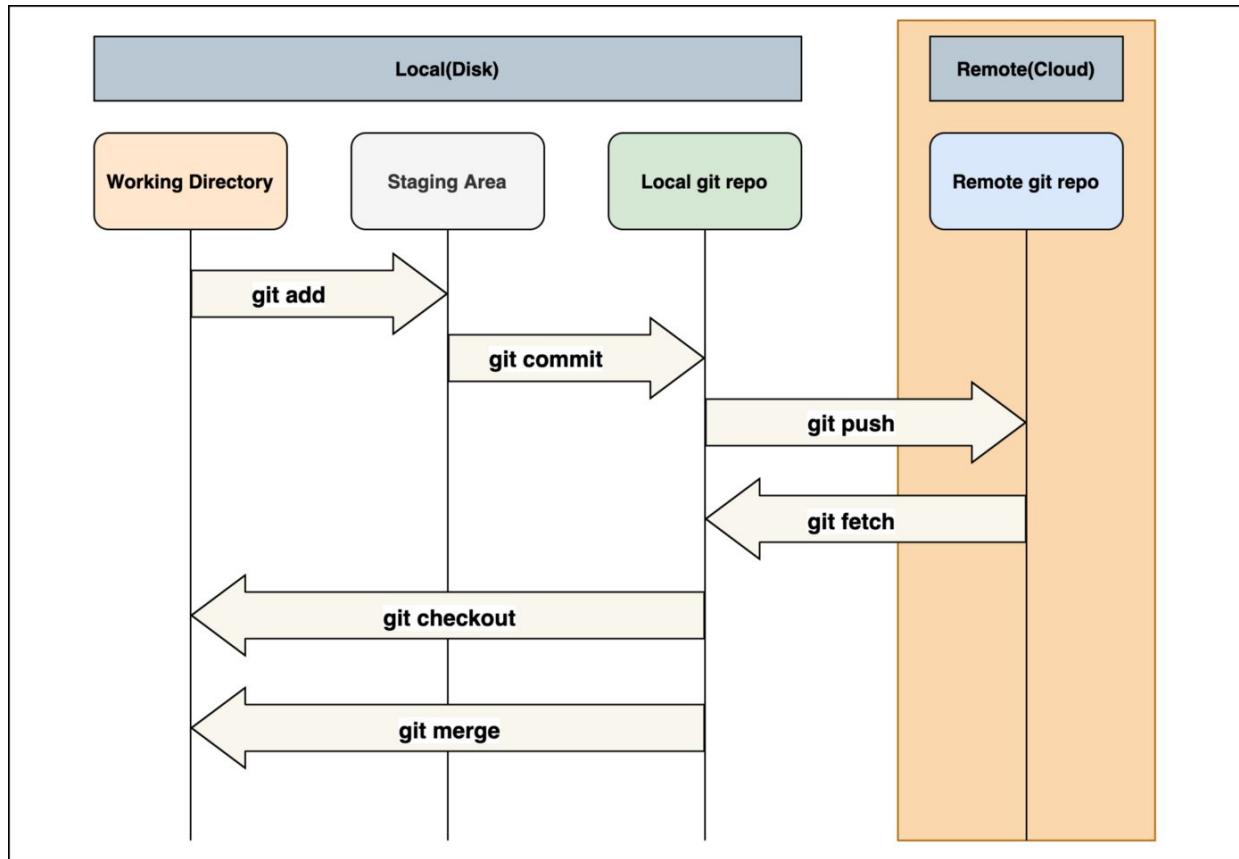
[master] abcdef0 Release of version 0.1 - John Releaser

Cuando trabajamos con Git, nuestros archivos pueden vivir y moverse entre 4 diferentes estados:

- **Archivos Tracked:** Son los archivos que viven dentro de Git, no tienen cambios pendientes y sus últimas actualizaciones han sido guardadas en el repositorio gracias a los comandos git add y git commit.
- **Archivos Staged:** Son archivos en Staging. Viven dentro de Git y hay registro de ellos porque han sido afectados por el comando git add, aunque no sus últimos cambios. Git ya sabe de la existencia de estos últimos cambios pero todavía no han sido guardados definitivamente en el repositorio porque falta ejecutar el comando git commit.
- **Archivos Unstaged:** Entiendelos como archivos “Tracked pero Unstaged”. Son archivos que viven dentro de Git pero no han sido afectados por el comando git add ni mucho menos por git commit. Git tiene un registro de estos archivos pero está desactualizado, sus últimas versiones solo están guardadas en el disco duro.
- **Archivos Untracked:** Son archivos que NO viven dentro de Git, solo en el disco duro. Nunca han sido afectados por git add, así que Git no tiene registros de su existencia.



# Introducción a GIT



# Introducción a GIT

## Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

## Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit] : [file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

## Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my\_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new\_branch

```
$ git branch new_branch
```

Delete the branch called my\_branch

```
$ git branch -d my_branch
```

Merge branch\_a into branch\_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

## Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

## Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

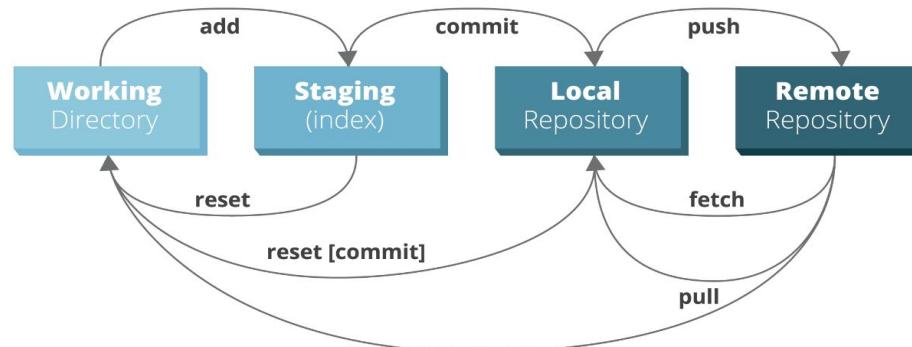
```
$ git push
```

## Finally!

When in doubt, use git help

```
$ git command --help
```

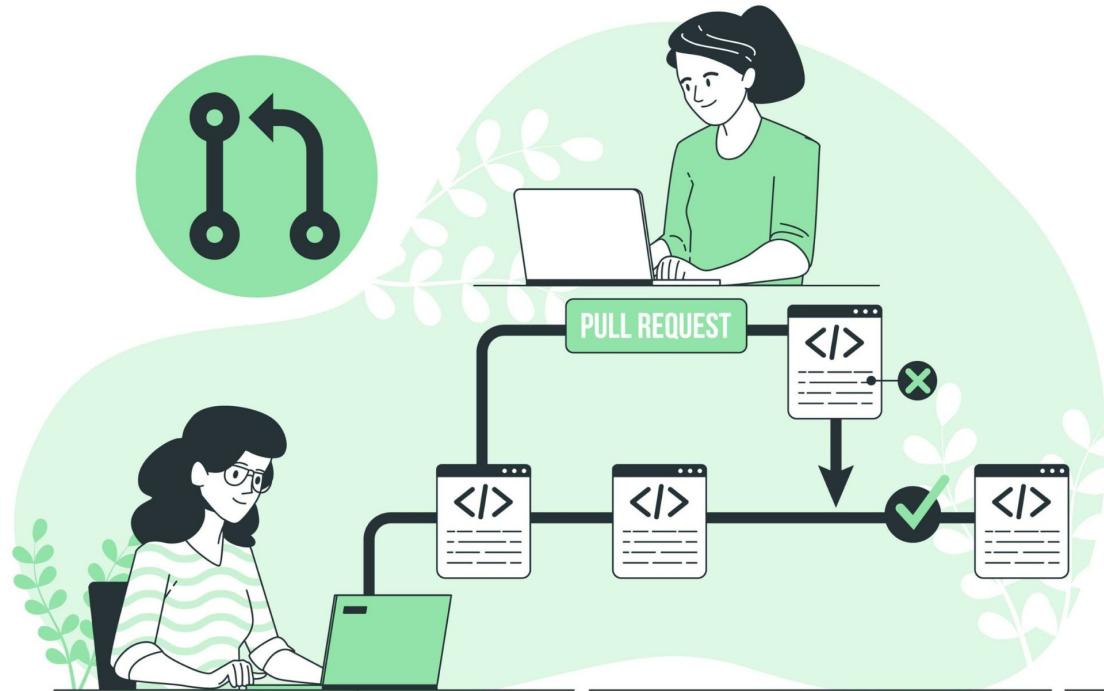
Or visit <https://training.github.com/> for official GitHub training.



<https://readme.md/>

**https://www.toptal.com/developers/gitignore**

# Introducción a GIT



# *Introducción a GIT*

---



**[https://learngitbranching.js.org/?locale=es\\_AR](https://learngitbranching.js.org/?locale=es_AR)**