**La línea de comandos**

# La ubicación actual

Escribir: $ pwd y oprimir Enter.

# Listando carpetas y archivos

Escribir: $ ls:

# Cambiando de ubicación

Escribir: $ cd / $ cd ~ $ cd ..

# Creando una carpeta

Escribir: $ mkdir Projects

# Abriendo archivos y carpetas en Atom

$ atom archivo1.txt 🡪 Abrir archivo

$ atom 🡪 Abrir carpeta

# Eliminando archivos y carpetas

$ rm /Users/germanescobar/Desktop/archivo1.txt 🡪 Eliminar archivo

$ rm -r /Users/germanescobar/Desktop/mi-carpeta 🡪 Eliminar carpeta

# Moviendo archivos y carpetas

$ mv archivo1.txt mi-carpeta 🡪 mover archivo

$ mv mi-carpeta otra-carpeta 🡪 mover carpeta

# Copiando archivos y carpetas

$ cp archivo1.txt mi-carpeta 🡪 copiar archivo en carpeta

$ cp -r mi-carpeta otra-carpeta 🡪 copiar carperta en otra carpeta

# Introducción a HTML

<p>Esto es un párrafo</p>

# Documentos de HTML

Un documento HTML es un archivo con extensión .html

<!DOCTYPE html>

<html>

<head>

<title>El título de la página</title>

</head>

<body>

<p>Este es un párrafo</p>

</body>

</html>

# Elementos

## Párrafos

<p>Esto es un párrafo</p>

<p>Esto es otro párrafo</p>

## Títulos (encabezados)

<h1>Este es un título 1</h1>

<h2>Este es un título 2</h2>

<h3>Este es un título 3</h3>

…

## Vínculos (links)

<a href="http://makeitreal.camp/">Make it Real</a>

El destino del vínculo se define en el **atributo** href.

## Imágenes

<img src="imagen.jpg" alt="Una imagen">

## Comentarios

<!-- Escribe acá tu comentario -->

# Introducción a CSS

Existen 3 formas para definir los estilos CSS:

* Inline (en línea): Utilizando el atributo style de los elementos HTML.
* Interno: Utilizando la etiqueta <style> dentro de <head>.
* Externo: Utilizando un archivo externo con extensión .css.

La forma más recomendada es utilizar un archivo externo.

# Inline CSS

Se utiliza para aplicarle estilos a un único elemento HTML:

<h1 style="color: blue">Este es un título de color azul</h1>

# CSS Interno

El CSS se define en el <head> dentro de la etiqueta <style>:



# CSS Externo

Es un archivo con extensión .css

Para referenciar el archivo CSS desde HTML se utiliza la etiqueta <link> en el <head>:

# 

# Reglas CSS

Cuando utilizas CSS interno o externo los estilos se definen utilizando **reglas CSS**.

# 

# Selectores

## Clases

# 

## Ids

# 

# Propiedades CSS

Las propiedades: un atributo, seguido de dos puntos (:), un espacio y el valor:

color: blue;

# Más elementos de HTML

# Importancia y énfasis al texto

<strong> muestra **negrilla** y <em> itálica.

<p>Esto es <strong>importante</strong> y quiero <em>hacer énfasis</em> en esto.</p>

# Saltos de línea y línea horizontal

<p>Hola<br>Amigos</p>

<br> no necesita una **etiqueta de cierre**. 🡪 Salto de Línea (Enter)

<hr> muestra una línea horizonal en el documento.

# Listas

Puedes crear listas ordenadas y no ordenadas.

Una **lista ordenada** está enumerada. Una **lista no ordenada** utiliza viñetas.

1. Item 1
2. Item 2
3. Item 3

* Item 1
* Item 2
* Item 3

Una **lista ordenada** se crea con la etiqueta <ol> y los items con la etiqueta <li>:

<ol>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ol>

Una **lista no ordenada** se crea con la etiqueta <ul> y los items con la etiqueta <li>:

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

**Nota:** Por ningún motivo utilices la etiqueta <li> por fuera de un <ul> o <ol>.

# Etiquetas invisibles

Existen dos etiquetas que no son visibles en el documento pero se utilizan para agrupar otros elementos y aplicarles estilos: <div> y <span>.

Veamos un ejemplo de <div> en donde estamos agrupando un título <h1>, una imagen <img> y un párrafo <p>:

<div class="main-section">

<h1>Título</h1>

<img src="imagen.jpg" alt="Imagen">

<p></p>

</div>

<span> se utiliza para aplicarle estilos a un texto (generalmente dentro de un parrafo):

<p>Hola, tu saldo es <span class="saldo">$12000</span></p>

# Elementos de bloque y en línea

Existen elementos que, independiente de su contenido, ocupan todo el ancho de la página como p, ol, ul, li y div. A estos elementos se les conoce como **elementos de bloque**.

Otros elementos como span, strong y em ocupan solo el espacio que ocupa su contenido interno. A estos elementos se les conoce como **elementos en línea**.

# Tablas

Una tabla se define con la etiqueta <table>.

Las filas se definen con la etiqueta <tr>.

Las celdas se definen con la etiqueta <td>.

<table>

<tr>

<td>Pedro</td>

<td>Perez</td>

</tr>

<tr>

<td>Juan</td>

<td>Gomez</td>

</tr>

</table>

# Agregando bordes

table, th, td {

border: 1px solid black;

}

# Uniendo los bordes

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

# Encabezados

<table>

<tr>

<th>Nombre</th>

<th>Apellido</th>

</tr>

<tr>

<td>Juan</td>

<td>Gomez</td>

</tr>

</table>

# Agregar espacio interno en las celdas

th, td {

padding: 15px;

}

**Celdas que ocupan más de una columna o fila**

# 

# Formularios

Los formularios nos permiten pedirle información a los usuarios.

Un formulario se crea con la etiqueta <form>:

# Campo de texto

<form>

<input type="text">

</form>

# Campo de contraseña

<form>

<input type="password">

</form>

# Casilla de verificación (checkbox)

<form>

<input type="checkbox"> Acepto recibir información

</form>

**Botón de envío**

<form>

<input type="submit" value="Enviar">

</form>

**Labels**

<form>

<label for="nombre">Nombre:</label> <input type="text" id="nombre" name="nombre">

</form>

**Botones de radio**

<form>

<input type="radio">

</form>

# Áreas de texto

<form>

<textarea rows="5"></textarea>

</form>

# Lista de selección

<form>

<select name="genero">

<option value="masculino">Masculino</option>

<option value="femenino">Femenino</option>

</select>

</form>

# El modelo de caja en CSS

**Márgenes**

p {

margin-top: 5px;

margin-right: 5px;

margin-bottom: 5px;

margin-left: 5px;

}

Atajo con margin

p {

margin: 5px 5px 5px 5px;

}

**Bordes**

p {

border-width: 1px;

border-style: solid;

border-color: red;

}

Atajo con border

p {

border: 1px solid blue;

}

## Bordes redondeados

p {

border: 2px solid red;

border-radius: 5px;

}

# Márgenes externas (padding)

p {

padding-top: 5px;

padding-right: 5px;

padding-bottom: 5px;

padding-left: 5px;

}

Atajo con padding

p {

padding: 5px 5px 5px 5px;

}

**Ancho y alto**

p {

width: 400px;

height: 300px;

}

# Fondos (backgrounds)

Para definir el color, la imagen, la posición, etc. de fondo en CSS:

* background-color: define el color de fondo.
* background-image: define la imagen de fondo.
* background-position: define la ubicación de la imagen de fondo.
* background-repeat: define si se repite la imagen de fondo horizontal y verticalmente.
* background-attachment: define si la imagen se mantiene fija o se desplaza con la página.

## background-color

p {

background-color: blue;

}

## background-image

p {

background-image: url('mi-imagen.jpg')

}

## background-repeat

* no-repeat: no repetir horizontal ni verticalmente.
* repeat-x: repetir solo horizontalmente.
* repeat-y: repetir solo verticalmente.

## background-position

p {

background-image: url('imagen.jpg');

background-repeat: no-repeat;

background-position: center center;

}

## background-size

p {

background-image: url('imagen.jpg');

background-repeat: no-repeat;

background-size: 60px 40px;

}

## background-attachment

background-attachment: fixed;

El atajo background

background: #00ff00 url("smiley.gif") no-repeat fixed center;

# Posicionamiento

## position: relative

p {

position: relative:

top: 20px;

left: 20px;

}

## position: absolute

p {

position: absolute:

left: 100px;

top: 100px;

}

## position: fixed

div.fixed {

position: fixed;

bottom: 0;

right: 20px;

}

# Selectores CSS

Lista de selectores (, )

h1, h2, h3 {

margin-bottom: 21px;

}

Más Específicos

p.importante {

font-size: 21;

}

Descendientes

div a.link span {

color: red;

}

Descendientes directos

div > .link {

color: red;

}

Cualquier selector

\* {

box-sizing: border-box;

}

# Pseudo clases



# Bordes, sombras y gradientes

# Bordes redondeados

div.redondeado {

border-radius: 5px;

}

div.redondeado {

border-radius: 5px 10px 10px 20px;

}

div.redondeado {

border-radius: 5px 10px 20px;

}

div.redondeado {

border-radius: 5px 10px;

}

**Gradientes**

Gradientes lineales

.gradiente {

background: linear-gradient(red, orange);

}

.gradiente {

background: linear-gradient(to right, blue, red, yellow, green);

}

.gradiente {

background: linear-gradient(to right, blue 20%, red 60%, yellow, green);

}

.gradiente {

background: linear-gradient(to right, rgba(255, 0, 0, 0), rgba(255, 0, 0, 1));

}

Gradientes radiales

.gradiente {

background: radial-gradient(red, yellow);

}

.gradiente {

background: radial-gradient(at 20px 20px, red, yellow);

}

.gradiente {

background: radial-gradient(circle at 20px 20px, red, yellow);

}

# Media Queries

body { // mobile first

font-size: 14px;

}

@media (min-width: 768px) { // tabletas

body {

font-size: 15px;

}

}

@media (min-width: 992px) { // escritorio

body {

font-size: 16px;

}

}

@media (min-width: 1200px) { // pantallas grandes

body {

font-size: 17px;

}

}

# JavaScript I

# Instalación y Verificación de Node.js

$ node -v

# Ejecutando JavaScript en el navegador

* + En Mac es Alt + Command + I.
  + A través de un documento HTML

Archivo llamado index.html y archivo llamado app.js

En app.js escribimos:

alert("Hola Amigo");

En index.html escribimos:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo JavaScript</title>

</head>

<body>

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

</body>

</html>

# Ejecutando código en Node.js

La consola de Node.js

## Desde la línea de comandos ejecutamos:

$ node

>

Para salir de la consola oprime Ctrl + D

## Desde un archivo y línea de comandos, ejecutamos:

Creamos un archivo app.js y escribimos:

console.log("Hola Mundo!");

Ejecutamos desde la línea de comandos

$ node app.js

# Tipos y operadores (Desde Archivo .js)

## Cadenas de texto

console.log('Y \'él dijo\': "Hola mundo"');

## Concatenando cadenas

console.log("Hola " + "Mundo " + "Cómo " + "Estás");

## Números

console.log(1 + 2)

console.log(3 \* 4 + 5)

console.log(8 / 2)

## Valores y expresiones booleanas (En línea de comandos)

$ node

> 5 > 3

true

> 5 >= 3

true

> 4 < 4

false

> 4 <= 4

true

> 2 === 2

true

> 2 !== 2

false

> "ruby" === "javascript"

false

> "ruby" !== "javascript"

true

# Variables

var name = "Germán"; // cámbialo por tu nombre

console.log("Hola " + name);

## La utilidad de las variables

var side = 5;

console.log("El perímetro de un cuadrado de lado " + side + " es " + (side \* 4));

console.log("El área de un cuadrado de lado " + side + " es " + (side \* side));

Se puede reasignar el valor a la variable de forma más fácil y cómoda.

Se pueden utilizar variables sin valor

# Condicionales

if (true) {

console.log("Hola Mundo");

}

if (1 == 1) {

console.log("Hola Mundo");

}

var num = 8;

if (num < 10) {

console.log("El número es menor a 10");

}

var num = 8;

if (num < 10) {

console.log("El número es menor a 10");

}

if (num >= 10) {

console.log("El número es igual o mayor a 10");

}

# De lo contrario (else)

var num = 8;

if (num < 10) {

console.log("El número es menor a 10");

} else {

console.log("El número es igual o mayor a 10");

}

# Condicionales Anidadas

var num = 8;

if (num < 10) {

console.log("El número es menor a 10");

} else {

if (num > 10) {

console.log("El número es mayor a 10");

} else {

console.log("El número es igual a 10");

}

}

# De lo contrario, si (else if)

var num = 8;

if (num < 10) {

console.log("El número es menor a 10");

} else if (num > 10) {

console.log("El número es mayor a 10");

} else {

console.log("El número es igual a 10");

}

# Condiciones compuestas

var num = 15;

if (num >= 10) {

if (num <= 20) {

console.log("El número está entre 10 y 20");

}

}

var num = 15;

if (num >= 10 && num <= 20) {

console.log("El número está entre 10 y 20");

}

var color = "negro";

if (color === "rojo" || color === "negro") {

console.log("Excelente elección");

}

# Operador condicional (ternario)

var num = 10;

num >= 15 ? console.log('Es mayor o igual que 15') : console.log('Es menor que 15');

# Ciclos - while

var i = 0;

while (i < 850) {

console.log("Hola mundo");

i = i + 1;

}

while (true) {

console.log("Hola Mundo");

}

while (false) {

console.log("Hola mundo");

}

var i = 0;

while (i < 850) {

console.log("Hola mundo");

i++;

}

**for**

for (var i = 0; i < 850; i ++) {

console.log("Hola mundo");

}

**while = for**

var i = 10; // el inicializador

while (i <= 20) { // la condición

console.log(i);

i = i + 2; // el incrementador

}

for (var i=10; i <= 20; i = i + 2) {

console.log(i);

}

var i = 100;

while (i > 0) {

console.log(i);

i--;

}

for (var i=100; i > 0; i --) {

console.log(i);

}

# Arreglos

var array = [1, "Pedro", true, false, "Juan"]

## Obteniendo elementos del arreglo

array[0]

## Recorriendo un arreglo

var array = [1, "Pedro", true, false, "Juan"];

for (var i = 0; i < array.length; i ++) {

console.log(array[i]);

}

## Reemplazando un elemento

var array = [1, "Pedro", true, false, "Juan"];

array[1] = "Germán"; // reemplazamos el elemento en la posición 1

// [1, "Germán", true, false, "Juan"]

## Insertando nuevos elementos

* + El método push te permite agregar un elemento al final de la lista.

var array = ["Pedro"];

array.push("Germán"); // ["Pedro", "Germán"]

array.push("Diana"); // ["Pedro", "Germán", "Diana"]

* + El método splice se utiliza tanto para insertar como eliminar elementos.

var array = ["Pedro", "Germán", "Diana"];

array.splice(0, 0, "Juan") // ["Juan", "Pedro", "Germán", "Diana"]

var array = ["Pedro", "Germán", "Diana"];

array.splice(1, 1); // ["Pedro", "Diana"]

# Más cadenas de texto

## Obteniendo la longitud de una cadena

Abre la consola de Node.js y escribe "hola".length

$ node

> "hola".length

4

## Recorriendo cadenas

**En consola**

$ node

> var str = "Hola Mundo"

undefined

> str[0]

'H'

> str[1]

'o'

> str[2]

`l'

> str[3]

'a'

**Con Archivo**

var str = "Hola Mundo";

for (var i = 0; i < str.length; i ++) {

console.log(str[i]);

}

## Partiendo y uniendo cadenas

**Partiendo - .split(“ ”)**

$ node

> "prueba,separar,cadenas".split(",")

[ 'prueba', 'separar', 'cadenas' ]

**Uniendo - .join(“ ”)**

$ node

> ["Hola", "mundo"].join(" ")

'Hola mundo'

# Funciones

**Sintaxis**

function hello() {

console.log("Hola Mundo");

}

**Invocación**

function hello() {

console.log("Hola Mundo");

}

hello(); // acá la estamos invocamos

# Argumentos

function hello(name) {

console.log("Hola " + name);

}

hello("Germán");

hello("David");

## Retornando un valor

function hello(name) {

return "Hola " + name;

}

var g1 = hello("Germán"); // podemos asignar el valor de retorno a una variable

console.log(g1);

// podemos llamar la función directamente en el parámetro de otra función.

console.log(hello("David"));

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

function hello(name) {

return "Hola " + name;

console.log("Esto nunca se va a imprimir");

}

console.log(hello("Pedro"));

## Cajas negras

En muchas ocasiones es bueno pensar en funciones como cajas negras que reciben unos parámetros de entrada y genera un valor de salida (el valor de retorno).

function bmi(weight, height) {

return weight / height \*\* 2

}

console.log("Tu IMC es: " + bmi(80, 1.8));