



***¡LES DAMOS LA  
BIENVENIDA!***

¿Están listos?

***RECUERDA PONER A GRABAR LA  
CLASE***

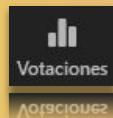


# ***PRESENTACIÓN DE ESTUDIANTES***



Por encuestas de Zoom:

1. País
2. Conocimientos previos en (temática del curso)
3. ¿Por qué elegiste el curso?



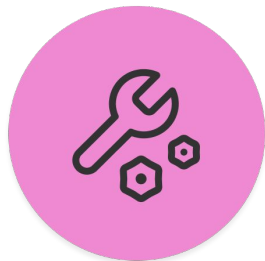


***¿DUDAS DEL ON-BOARDING?***

**MIRALO AQUI**

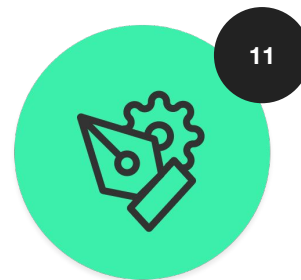
# ***DESAFÍOS Y ENTREGABLES***

Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



## **Desafíos genéricos**

Ayudan a poner en práctica los conceptos y la teoría vista en clase No deben ser subidos a la plataforma.



## **Desafíos entregables**

Relacionados completamente con el **Proyecto Final**. Deben ser subidos obligatoriamente a la plataforma hasta 7 días luego de la clase para que sean corregidos.

# ***DESAFÍOS Y ENTREGABLES***

Son actividades o ejercicios que se realizan durante la cursada, para enfocarse en la práctica.



## **Entregas del Proyecto Final**

Entregas con el estado de avance de tu **proyecto final** que deberás subir a la plataforma a lo largo del curso y hasta 7 días luego de la clase, para ser corregidas por tu docente o tutor/a.



# ***PROYECTO FINAL***

El Proyecto Final se construye a partir de los **desafíos** que se realizan clase a clase. Se va creando a medida que el estudiante sube los desafíos entregables a nuestra plataforma.

El objetivo es que cada estudiante pueda utilizar su Proyecto Final como parte de su portfolio personal.

El **proyecto final** se debe subir a la plataforma la ante-última o última clase del curso. *En caso de no hacerlo tendrás 20 días a partir de la finalización del curso para cargarlo en la plataforma. Pasados esos días el botón de entrega se inhabilitará.*

***¿CUÁL ES NUESTRO PROYECTO FINAL?***



# ***CREA TU PROPIA TIENDA ECOMMERCE***

Proyecto  
Final



Desarrollarás el front end de una tienda online con carrito de compras, utilizando los componentes de React y Firebase como servidor en la nube. Crearás así una experiencia de usuario amigable con actualizaciones visuales instantáneas y código escalable.



# ***PROYECTOS DE NUESTROS ESTUDIANTES***

- A.M. Florist: <https://unruffled-hermann-20d922.netlify.app/>
- Refugio Tienda Deco:  
<https://refugio-tiendadeco-marceloluismoreno.netlify.app/>
- Del Campe: <https://cocky-bose-3d69f9.netlify.app/>



<b><i>ENTREGA</i></b>	<b><i>REQUISITO</i></b>	<b><i>FECHA</i></b>
<b>1° entrega</b>	Durante esta entrega, se hará una revisión integral del estado actual de avance de tu proyecto.	<b>Clase N° 8</b>
<b>Proyecto Final</b>	Deberás subir a la plataforma el link a tu app de e-commerce completamente funcional.	<b>Clase N° 14</b>

# ¡IMPORTANTE!

Los desafíos y entregas de proyecto se deben cargar hasta siete días después de finalizada la clase. Te sugerimos llevarlos al día.

LUNES 16/03 20:30HS

10. Estrategia de contenido para Twitter y LinkedIn

DESAFÍO - EXPIRA EL 23/03/2020

Crear publicaciones para Twitter

👍

● HOY 20:30

📁

Tenes tiempo hasta el  
23/03/2020

↑ ENTREGAR

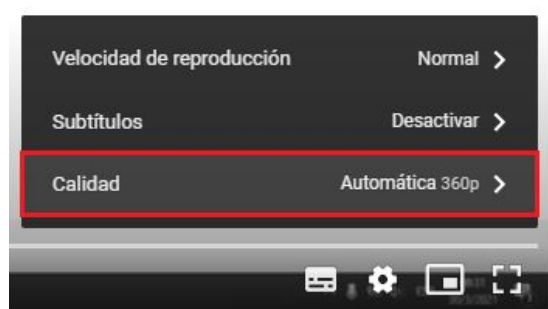
# ***TUTORIALES PARA INSTALACIONES***

# ¿CÓMO INSTALAR PROGRAMAS Y SOFTWARES?

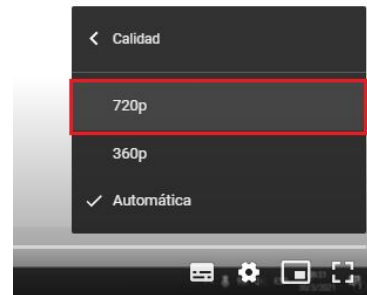
Dentro de la carpeta [“Tutoriales para instalaciones”](#) encontrarás videos donde te explicamos cómo instalar los softwares y programas que utilizarás a lo largo del curso. Para optimizar la calidad de imagen de los mismos al reproducirlos, deberás seguir los siguientes pasos:



Haz clic en configuración.



Luego en calidad.



Y por último selecciona 720p.



# Repaso

- ✓ El objetivo de esta dinámica es que los estudiantes puedan repasar algunos aspectos fundamentales de JavaScript antes de comenzar este curso.

Te proponemos realizar la siguiente actividad de repaso.

[Actividad de repaso](#)



Clase 01. REACT JS

***NIVELACIÓN***





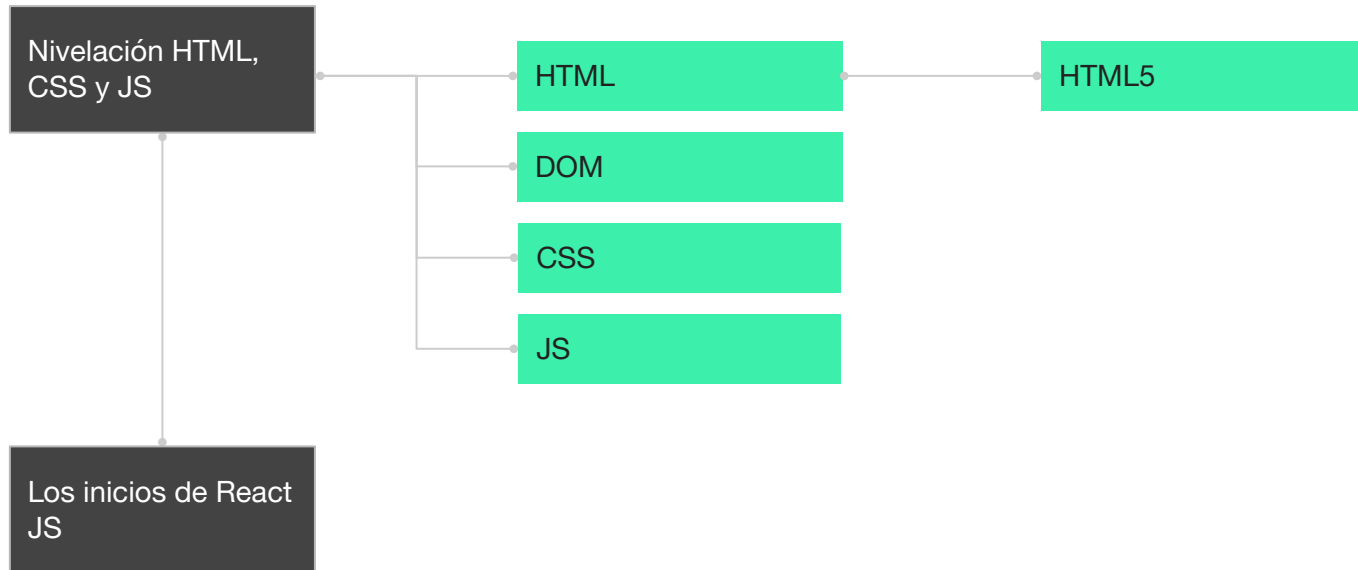
## ***OBJETIVOS DE LA CLASE***

- Repasar HTML, CSS y fundamentos JavaScript.
- Conocer la sintaxis de JavaScript y ECMAScript 6.
- Realizar una introducción a React JS.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE 1

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 1



### Nivelación



EJEMPLO EN VIVO



FORMULARIO

## Clase 2



### Instalación y configuración del entorno



EJEMPLO EN VIVO



CREAR LA APP UTILIZANDO EL CLI

## Clase 3



### JSX y Transpiling

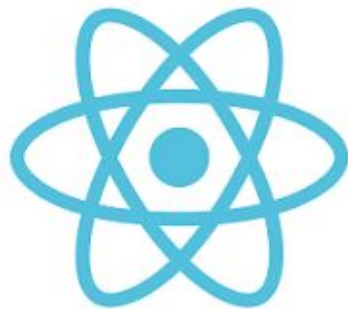


EJEMPLOS EN VIVO



MENÚ E-COMMERCE

# ***NIVELACIÓN HTML, CSS Y JS***



**React JS** es una biblioteca para **desarrollo web**, por lo cual debemos contar con conocimientos mínimos sobre los lenguajes que el navegador web interpreta.

***HTML***

***CODER HOUSE***

**HTML** es un **lenguaje de etiquetas**,  
el cual dará la estructura para  
nuestras páginas web.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Curso de React</title>
</head>
<body>
  <p>Esta es la primera clase</p>
</body>
</html>
```



# ***HTML 5***

HTML 5 es una nueva versión de diversas especificaciones, entre las que se encuentran:

- HTML 4.
- XHTML 1.
- CSS Nivel 2.
- DOM Nivel 2 (Document Object Model).



# ¿CUÁLES SON LAS NOVEDADES DE HTML5?

HTML 5 incluye novedades significativas en diversos ámbitos. Este nuevo estándar supone mejoras en áreas que hasta ahora quedaban fuera del lenguaje y para las que se necesitaba utilizar otras tecnologías:

- Estructura del `<body>`
- Etiquetas para contenido específico
- Canvas
- Bases de datos locales
- Web Workers
- Aplicaciones web Offline
- Geolocalización

# UN POCO MÁS DE HTML

Ahora que ya entendemos qué son HTML y HTML 5, comencemos a ver un poco más de estructuración en HTML:

```
<div class="site-logo col-6">
  <a href="index.html">
    
  </a>
</div>
<nav class="site-navigation">
  <ul class="site-menu js-clone-nav d-none d-lg-block">
    <li><a href="#acerca" class="nav-link">ACERCA DEL ARTISTA</a></li>
    <li><a href="#music">MÚSICA</a></li>
    <li><a href="#video">COMPOSICIONES AUDIOVISUALES <span class="ml-1 mr-1">/>
    <li><a href="#prensa">PRENSA</a></li>
  </ul>
</nav>
```

# ***DOCTYPE***

DOCTYPE no es una etiqueta, sino una instrucción para **indicar al navegador qué versión de HTML vamos a utilizar.**

El DOCTYPE mostrado en nuestro ejemplo anterior es del estándar HTML 5.

```
<!DOCTYPE html>
```

# ***METADATOS DEL DOCUMENTO***

- **<title>**: define el título del documento, el cual se muestra en la barra de título del navegador o en las pestañas de página.
- **<base>**: define la URL base para las URLs relativas en la página.
- **<link>**: utilizada para enlazar JavaScript y CSS externos.
- **<meta>**: sirve para aportar información sobre el documento. Entre **otras cosas** con esta etiqueta definimos la codificación que tendrá nuestro archivo, los mismos pueden ser:
  - UTF-8
  - ANSI: es el formato estándar de codificación de archivos utilizados en el Bloc de notas.

# ***SCRIPTING***

- **<script>**: define tanto un script interno como un enlace hacia un script externo. El lenguaje de programación es JavaScript
- **<noscript>**: define un contenido alternativo a mostrar cuando el navegador no soporta scripting.

# NOSCRIPT

El lenguaje HTML define la etiqueta `<noscript>` para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. El siguiente código muestra un ejemplo del uso de la etiqueta

`<noscript>`:

```
<head> ... </head>
```

```
<body>
```

```
<noscript>
```

```
  <p>Bienvenido a Mi Sitio</p>
```

```
  <p>La página que estás viendo requiere para su funcionamiento el uso de  
  JavaScript.
```

```
Si lo has deshabilitado intencionadamente, por favor vuelve a activarlo.</p>
```

```
</noscript>
```

```
</body>
```

La etiqueta `<noscript>` se debe incluir en el interior de la etiqueta `<body>` (normalmente se incluye al principio de `<body>`). El mensaje que muestra `<noscript>` puede incluir cualquier elemento o etiqueta XHTML.

***DOM***

***CODER HOUSE***

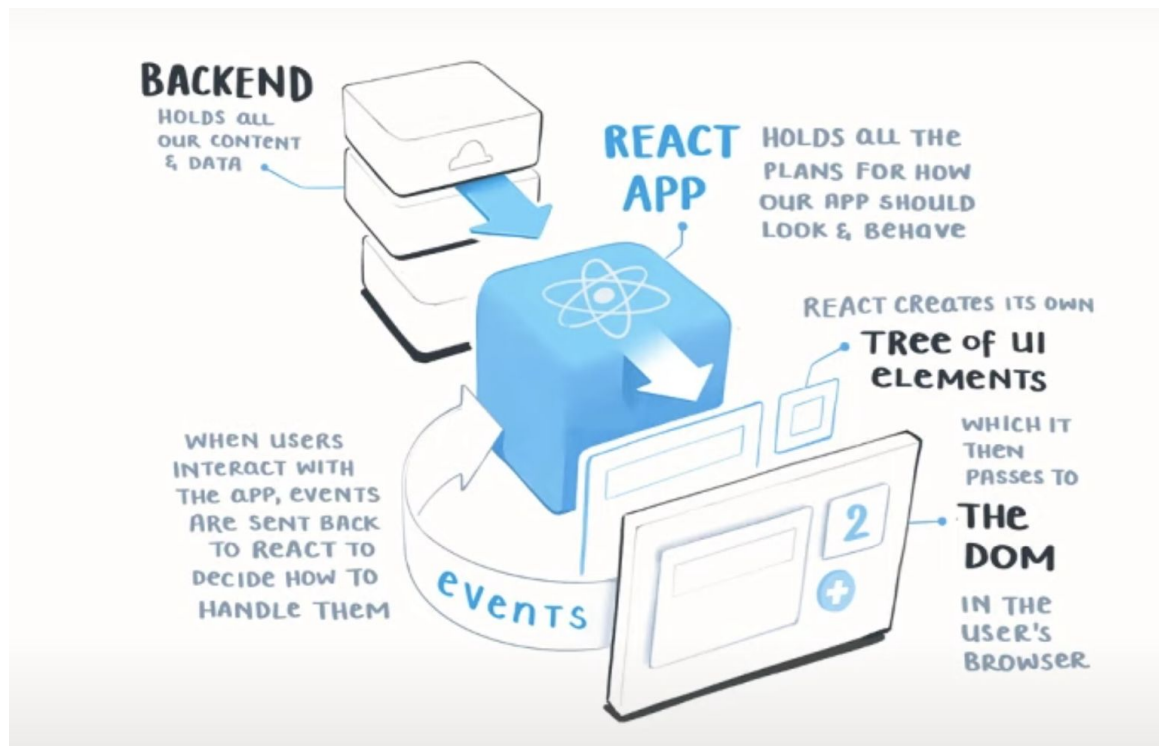


# ***DOM***

DOM (Document Object Model o modelo de objetos del documento) nos sirve para **acceder** a cualquiera de los **componentes** que hay dentro de una página.

Por medio del DOM podremos **controlar al navegador** en general y a los distintos elementos que se encuentran en la página.

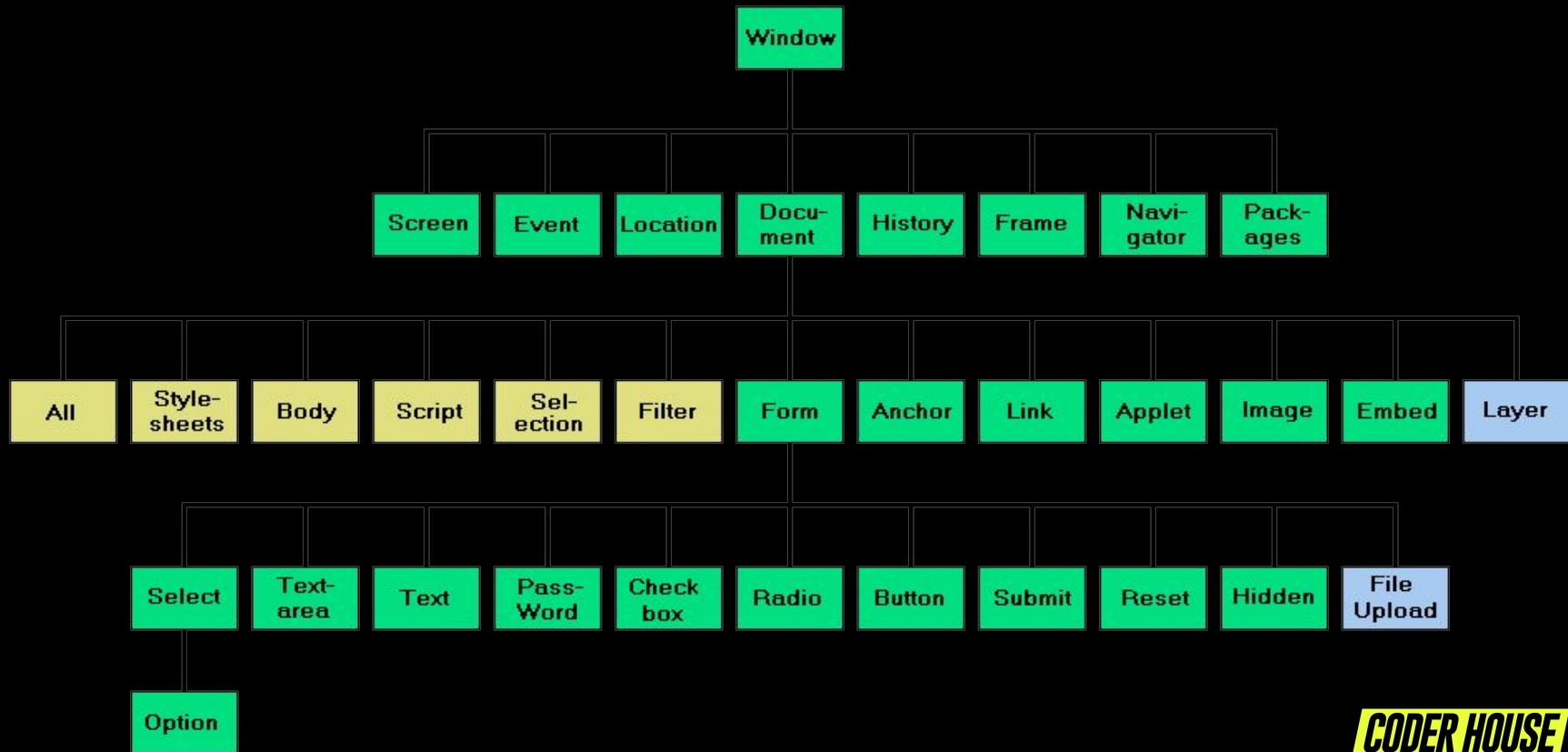
# DOM COMO METÁFORA



Ilustradora: [Maggie Appleton](#) @ Woman of React 2020

**CODER HOUSE**


# JERARQUÍA DEL DOM




# UN MUNDO DE API'S

Siempre es recomendable **avanzar con criterio** y ver qué funcionalidades usamos del **Navegador**, ya que hay centenas de API's, pero:

- No todas están listas para **producción**.
- No todas tienen soporte en todos los navegadores.

Especificación	Estado	Comentario
Geolocation API	 REC Recommendation	Especificación inicial

Specification	Status	Comment
Web Animations The definition of 'AnimationEffect' in that specification.	 WD Working Draft	Editor's draft.

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


# UN MUNDO DE API'S

Todo depende del **contexto** del trabajo y el **target** de usuarios / plataformas.

Puedo emocionarme para luego enterarme que algunos usuarios no pueden utilizar la experiencia.

Sugerencia: <https://caniuse.com>

Intenta chequear si puedes utilizar ES6 en cualquier navegador

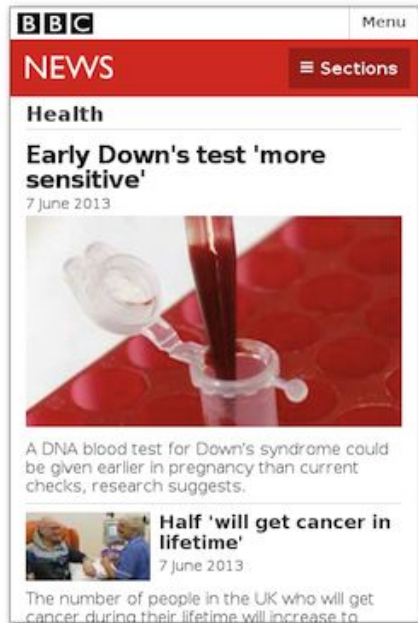


Desktop						Mobile					
Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0

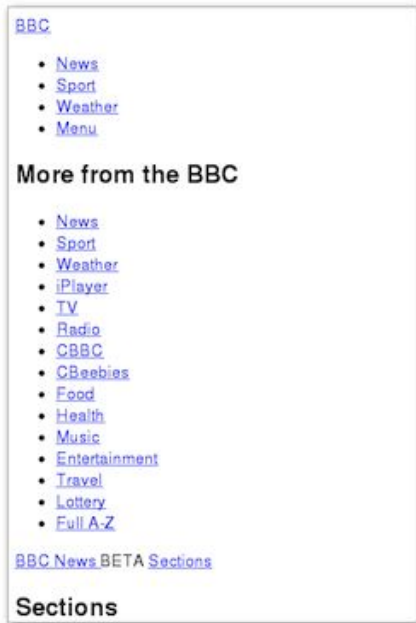
Fuente: <https://developer.mozilla.org/es/docs/Web/API>

***CSS***

***CODER HOUSE***



Con CSS.



Sin CSS.

CSS (cascading style sheets - hojas de estilo en cascada) es un lenguaje de diseño gráfico con el cual podremos **dar estilos** (diseño, colores, márgenes) a nuestras webs desarrolladas con HTML. Veremos la sintaxis básica de CSS

# ***INCLUIR HOJA DE ESTILO EN HTML***

Para incluir una hoja de estilo en nuestro documento

HTML podemos utilizar el elemento `<link>`.

Dentro del archivo CSS externo incluiremos las sentencias correspondientes, como lo hacemos con un archivo css en línea.

Por ejemplo:

```
<!-- MAIN CSS -->  
<link rel="stylesheet" href="css/style.css">
```



# CLASES CSS

Podemos seleccionar uno o más elementos html a través de su **clase** o su **atributo “id”** en CSS. O todos los elementos de un **tipo**

Modos básicos de “**seleccionar/capturar**” elementos por su:

Tipo de elemento (div, ul, li, nav, input)	div { background: lightgray; }
ID (usando # adelante)	#nombre { background: coral; }
Clase (usando punto inicial)	.apellido

# ***CLASES CSS***

Elemento

Id

Clase

```
<div>@gaearon</div>
```

```
<div id="nombre">Dan</div>
```

```
<div class="apellido">Abramov</div>
```

# CLASES CSS

Error clásico:

Los # y los puntos “.” de los selectores se usan en la hoja de estilos, no en el elemento

```
X  
<div id="#nombre">Dan</div>  
X  
<div class=".apellido">Abramov</div>
```



***BREAK***

**¡5/10 MINUTOS Y VOLVEMOS!**

# ***JAVASCRIPT***

***CODER HOUSE***

# ¿QUÉ ES?

JavaScript es el **lenguaje de programación web** por excelencia.

Decimos que se trata de un lenguaje de programación **interpretado**.

Su uso más conocido es del lado del cliente (**client-side**), corriendo en el navegador web, permite mejoras en la interfaz de usuario.

React está desarrollado en JS, por eso lo llamamos **reactjs**.



# ***¡PREGUNTA RÁPIDA!***

*“En una propuesta laboral decía que necesitaba saber Vanilla JS, ¿Dónde lo puedo aprender?”*

¡Tal como el helado más **común**, el saborizante más **conocido**, o el clásico de los clásicos!

¡JS sin aditivos ni conservantes!

**(mejor dicho, sin librerías externas)**



# ***INCLUIR EL ARCHIVO JS EN NUESTRO HTML***

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

Esto debe ser incluido en el head del html. También podemos introducir nuestro código javascript dentro de la etiqueta

`<script></script>`. Por ejemplo:

```
<script>
  function mi_funcion() {
    |   document.getElementById("mi_funcion").innerHTML = "Ejemplo de mi funcion"
  }
</script>
```



# ***TIPOS DE EJECUCIÓN***

## **Ejecución directa**

Es el método de ejecución de scripts más básico. En este caso se incluyen las **instrucciones dentro de la etiqueta <script>**, y cuando el navegador lee la página y encuentra un script va interpretando las líneas de código y las va **ejecutando una después de otra.**

# ***TIPOS DE EJECUCIÓN***

## **Respuesta a un evento**

Los eventos son **acciones que realiza el usuario**. Los programas como Javascript están preparados para atrapar determinadas acciones realizadas, en este caso sobre la página, y **realizar acciones como respuesta**.  
Por ejemplo la pulsación de un botón, el movimiento del ratón o la selección de texto de la página.

# ***VARIABLES***

La manera estándar de JS de **almacenar valores** en memoria es a través del uso de variables.

```
var nombre = 'dan';  
  
let apellido = 'abramov';
```

# ***DEFINIR UNA CONSTANTE***

Desde ES6 (ECMA 2015) podemos definir una constante con la palabra reservada `const`

```
const PI = 3.141593;  
alert(PI > 3.0);
```

Las **const** no pueden cambiar la referencia con la que fueron inicializadas.

# ***¡RECUERDEN!***

Puede ser poco intuitivo, pero...  
Que una constante mantenga para siempre  
la misma referencia no significa que esa  
referencia tenga el mismo contenido, de  
hecho, **puedo cambiarlo.**

# ***DOM***

DOM (Document Object Model o modelo de objetos del documento) nos sirve para **acceder** a cualquiera de los **componentes** que hay dentro de una página.

Por medio del DOM podremos **controlar al navegador** en general y a los distintos elementos que se encuentran en la página.

# LET

Podemos utilizar `let` en lugar de `var` a la hora de declarar una variable, cuando queremos que ésta sólo sea accedida de manera local en determinado ámbito. Por ejemplo:

```
let lenguaje = "Javascript";  
let adjetivo = "awesome";  
  
console.log(`Solo quiero decir que ${lenguaje} es ${adjetivo}`);  
// Solo quiero decir que Javascript es awesome
```

# ***FUNCIÓN ARROW***

```
//ES6
var miFuncion = (num) => num + num;

//ES6
var data = [{...}, {...}, {...}];
data.forEach(elem => {
  console.log(elem);
})
```

En ambos ejemplos se sustituye el uso de la palabra reservada **function**, dando simplicidad al código.

*(parametro1,parametro2,...,parámetro n) => {Definición de la función}*




# UN MUNDO DE API'S

Todo depende del **contexto** del trabajo y el **target** de usuarios / plataformas.

Puedo emocionarme para luego enterarme que algunos usuarios no pueden utilizar la experiencia.

Sugerencia: <https://caniuse.com>

Intenta chequear si puedes utilizar ES6 en cualquier navegador



Desktop						Mobile					
Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0
75	79	63	No	62	13.1	75	75	63	54	13.4	11.0

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

# ***PARA REPASAR...***

- Operador condicional ternario
- Desestructuración
- Operador Spread
- Parámetros Rest





# ***VAMOS AL CÓDIGO***



***CODER HOUSE***

# TEMPLATE STRINGS

Antes de ES6, creábamos mensajes compuestos de la siguiente manera:

```
var nombre = 'dan';  
  
let apellido = 'abramov';  
  
const cuenta = '@gaearon'  
  
console.log('Mi nombre es ' + nombre + ' ' +  
apellido + '\n encuéntrenme en ' + cuenta)
```

console was cleared

Mi nombre es dan abramov  
encuéntrenme en @gaearon

>

Problemas: ¿cuáles detectan?

Agregar espacios manuales entre concatenaciones '+', dificultad para modificar texto, multiline explícito \n, poca legibilidad.

# TEMPLATE STRINGS

Con **ES6** podemos interpolar **strings** de una forma más sencilla, a como estábamos haciendo hasta ahora. Miremos el ejemplo:

```
var nombre = 'dan';  
  
let apellido = 'abramov';  
  
const cuenta = '@gaearon'  
  
console.log(`Mi nombre es ${nombre} ${apellido}  
encuéntrenme en ${cuenta}`);
```

 Preview (local)  ☒ Clear

Console was cleared

Mi nombre es dan abramov  
encuéntrenme en @gaearon

>

# ***VALORES POR DEFECTO: ANTES Y HOY***

Antes teníamos que comprobar si la variable ya tenía un valor, para dar una respuesta por defecto.

En cambio hoy, podemos usar otra sintaxis, aunque tenga algunas diferencias de comportamiento.

```
function saludaloA(nombre) {  
  console.log(nombre || 'coder');  
}
```

# VALORES POR DEFECTO: ¡CUIDADO!

```
const saludaloA = (nombre = 'coder') => {  
  console.log(nombre);  
};  
  
saludaloA('dan');  
saludaloA(null);  
saludaloA(undefined);  
saludaloA();
```



Preview (local)



Clear console on reload

Console was cleared

dan

null

coder

coder

>

En muchas convenciones **null** es considerado como algo explícito de un valor que **es la nada**. Mientras que al **undefined** se lo considera como un valor que aún no se puede inferir que es, cuando no pasamos un parámetro será **undefined** (indefinido)

Dicho esto, si le paso **null** a **function**, ¡el valor por defecto no se aplicará!

# MÓDULOS

Ahora JavaScript se empieza a parecer a lenguajes como Python o Ruby. **Llamamos a las funciones desde los propios scripts**, sin tener que importarlos en el HTML si usamos JavaScript en el navegador.

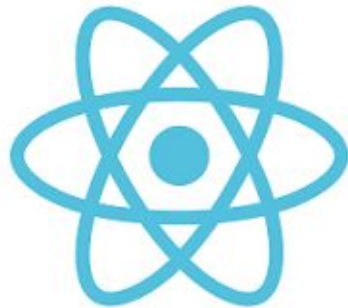
```
module 'person' {  
  export function hello(nombre) {  
    return nombre;  
  }  
}
```

Y para importar en otro fichero:

```
import { hello } from 'person';  
var app = {  
  foo: function() {  
    hello('Niko');  
  }  
}  
  
export app;
```



# ***REACT JS: LOS INICIOS***



React JS fue creada por Jordan Walke, un ingeniero de software en Facebook, inspirado por los problemas que tenía la compañía con el mantenimiento del código de los anuncios dentro de su plataforma. React intenta ayudar a los desarrolladores a **construir aplicaciones que usan datos que cambian todo el tiempo**. Su objetivo es ser **sencilla, declarativa y fácil de combinar**.

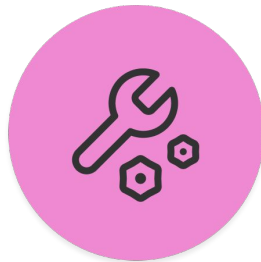
# ***SITIOS BASADOS EN REACT JS***



**NETFLIX**

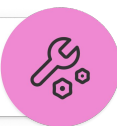


y más...



# ***FORMULARIO***

Con lo visto hasta ahora, te proponemos crear un formulario de contacto.



## ***¡A PRACTICAR!***

Crear un formulario de contacto utilizando HTML5, CSS, y añade JS para simular su funcionamiento.

Cuentas con 15 minutos para hacer la actividad.

# ***CONCLUSIONES***

***¿PREGUNTAS?***





***TE INVITAMOS A QUE COMPLEMENTES  
LA CLASE CON LOS SIGUIENTES  
CODERTIPS***





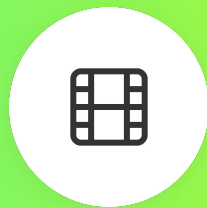
# ***VIDEOS Y PODCASTS***

- [Aprende Programación Web y construye el futuro de nuestra humanidad](#) | **Coderhouse**
- [Desarrollo freelance](#) | **Coderhouse**
- [Desarrollo profesional](#) | **Coderhouse**



# ***VIDEOS Y PODCASTS***

- [CoderNews](#) | **Coderhouse**
- [Serie de Branding](#) | **Coderhouse**
- [Serie para Emprendedores](#) | **Coderhouse**
- [Serie Aprende a Usar TikTok](#) | **Coderhouse**
- [Serie Finanzas Personales](#) | **Coderhouse**
- [CoderConf](#) | **Coderhouse**



***¿QUIERES SABER MÁS? TE DEJAMOS  
MATERIAL AMPLIADO DE LA CLASE***



- <https://es.wikipedia.org/wiki/HTML> | **Wikipedia**
- <http://www.w3schools.com/html/> | **w3schools.com**
- <http://www.w3schools.com/css> | **w3schools.com**
- [https://developer.mozilla.org/es/docs/Tools/Page\\_Inspector](https://developer.mozilla.org/es/docs/Tools/Page_Inspector) | **MDN web docs**
- <https://developer.mozilla.org/es/docs/Web/API> | **MDN web docs**



***¿YA CONOCES LOS BENEFICIOS QUE TIENES  
POR SER ESTUDIANTE DE CODERHOUSE?***

***CODER HOUSE***

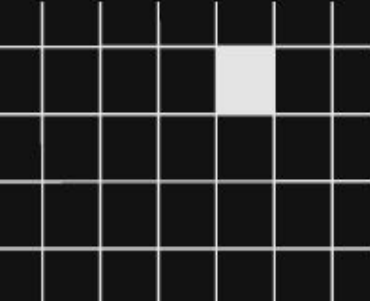
# ***BENEFICIOS*** 🧐

Haz clic [aquí](#) y conoce todos nuestros beneficios exclusivos para estudiantes de Coderhouse.



# ***¡MUCHAS GRACIAS!***

Resumen de lo visto en clase hoy:

- Nivelación HTML, CSS, y JS.
  - Introducción a ReactJS.
- 



***OPINA Y VALORA ESTA CLASE***