

JS

+



JAVASCRIPT & JQUERY



INTRO A JAVASCRIPT

¿QUÉ ES JAVASCRIPT?



- Es un lenguaje de programación que fue diseñado para darle interacción a las páginas web.
- Es posible usarlo para hacer en múltiples ámbitos, aplicaciones de servidor e incluso de escritorio.

¿POR QUÉ DEBEMOS APRENDER JAVASCRIPT?



- Es uno de los pilares del desarrollo front-end.
- Es uno de los lenguajes de programación más populares en la actualidad.

¿QUÉ PODEMOS HACER CON JAVASCRIPT?



Agregar interactividad
entre nuestro sitio
web y el usuario

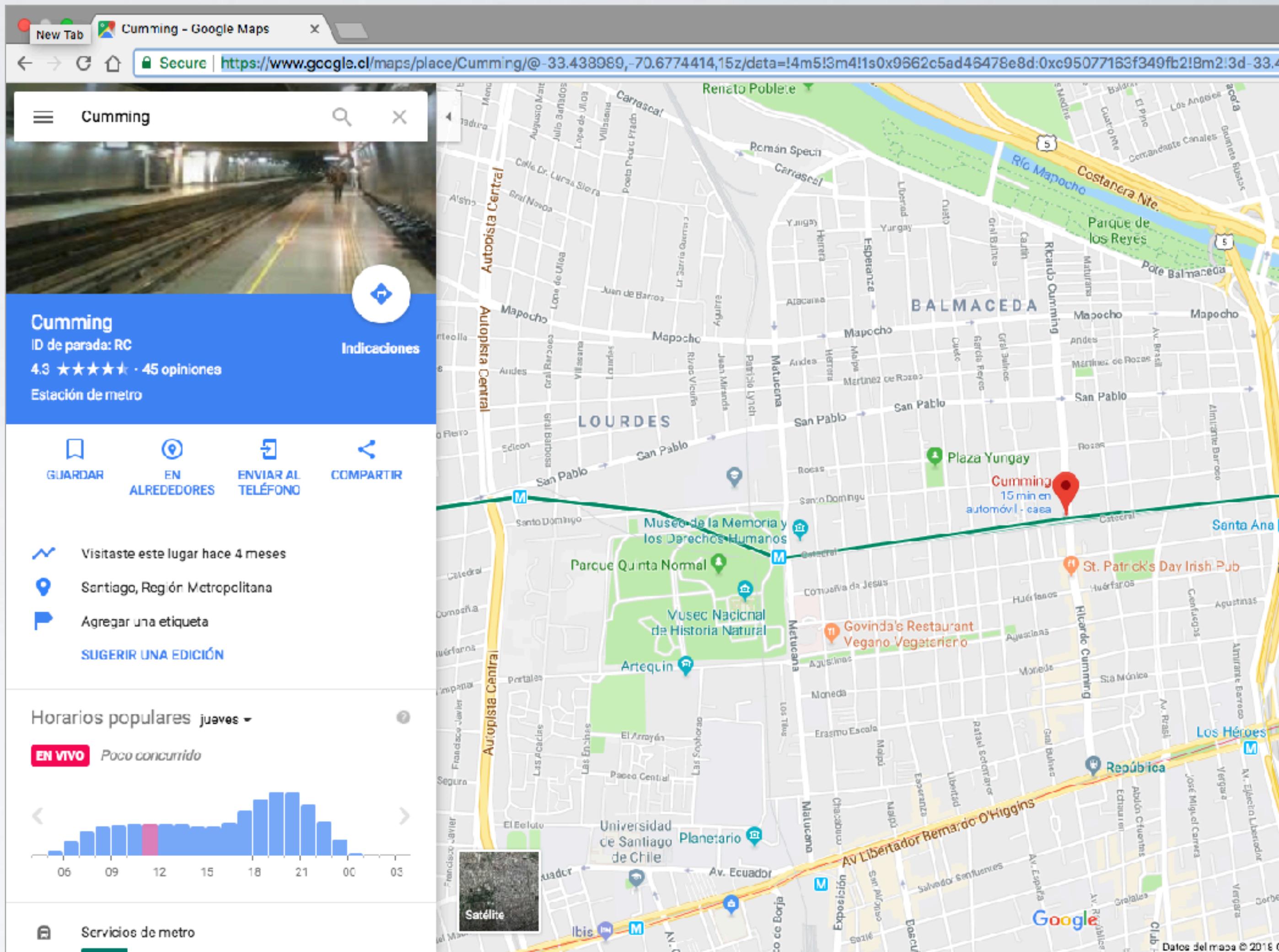


Integrar servicios que
nos entregan nuevas
funcionalidades



Actualizar información
sin necesidad de
recargar nuestra página

Páginas que visitamos a diario usan Javascript para hacer más fácil la interacción del usuario con la web.
Un buen ejemplo de esto es la página web de Google Maps.



Antes de comenzar es importante saber que Java no es lo mismo que Javascript

JS

Es un lenguaje de
programación interpretado
por el navegador



Es un lenguaje de
programación compilado
e interpretado por el JVM

Si quieres saber más visita el siguiente [link](#)

Comencemos creando nuestras primera
página web con **Javascript**

1.- CREAR EL ARCHIVO HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Mi primer script</title>
</head>
<body>

</body>
</html>
```

- Crearemos un nuevo archivo HTML.

2.- AGREGAR LA ETIQUETA <SCRIPT>

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Mi primer script</title>
</head>
<body>
  <script></script>
</body>
</html>
```

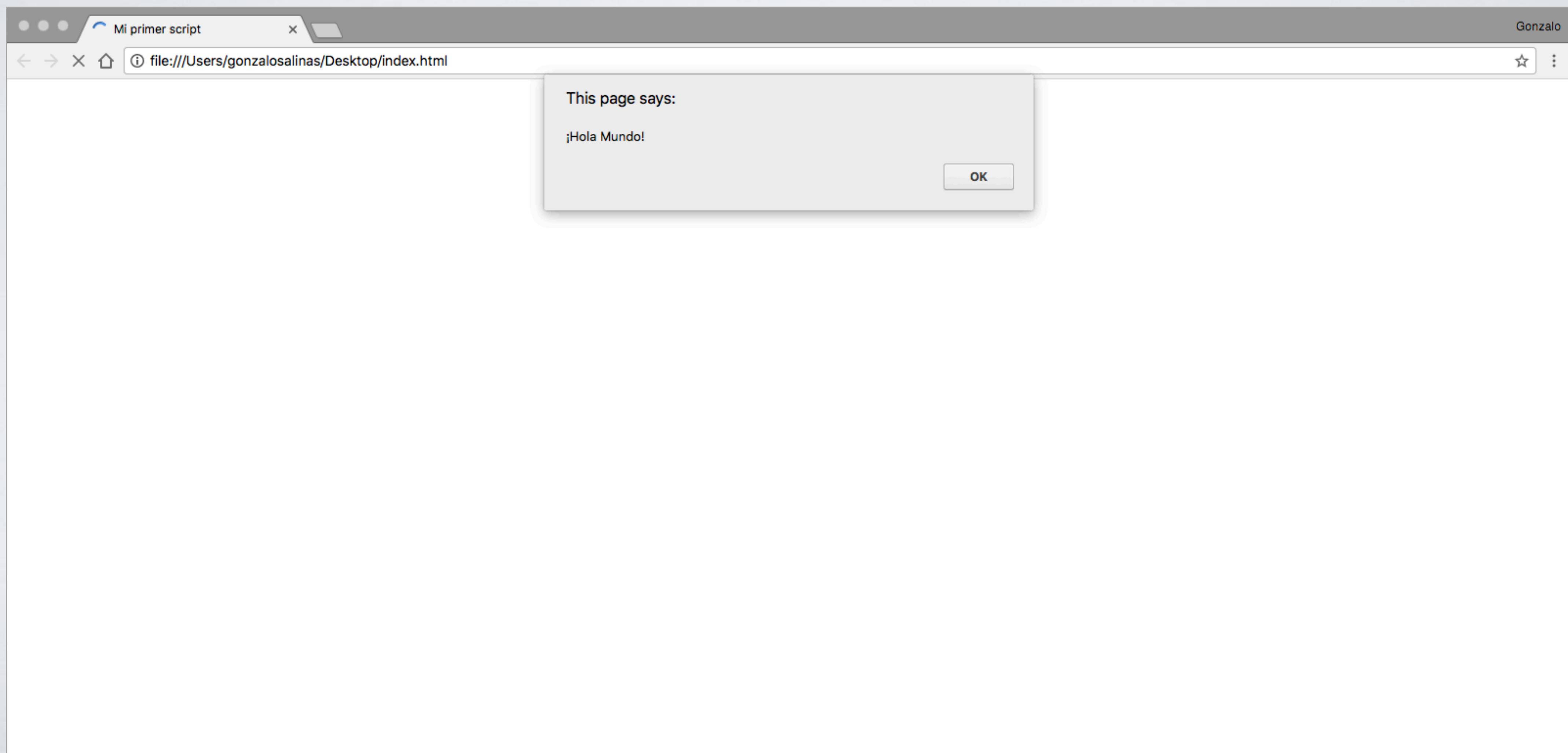
- Luego, escribiremos dentro de <body> la etiqueta <script>.
- Dentro de la etiqueta script tenemos que escribir Javascript y no HTML

3.- ESCRIBIR EL SCRIPT

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Mi primer script</title>
</head>
<body>
<script>
  alert("¡Hola Mundo!");
</script>
</body>
</html>
```

- Dentro de la etiqueta `<script>` escribiremos:
alert("¡Hola Mundo!");

Si todo salió bien, al abrir la página debiese aparecer un mensaje diciendo:



Existen otras dos formas para agregar
Javascript dentro de nuestro HTML

AGREGAR JS DE MANERA INLINE

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>JS inline</title>
</head>
<body>
  <h1>El botón con Javascript inline</h1>
  <button onclick="alert('hola mundo')">Saludo</button>
</body>
</html>
```

AGREGAR JS CON ARCHIVO EXTERNO

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Agregando Javascript en un archivo externo</title>
</head>
<body>
  <h1>El botón con Javascript inline</h1>
  <script src="script.js"></script>
</body>
</html>
```

Cuando entendamos más sobre javascript
discutiremos las buenas prácticas y las mejores
formas de incorporarlo.

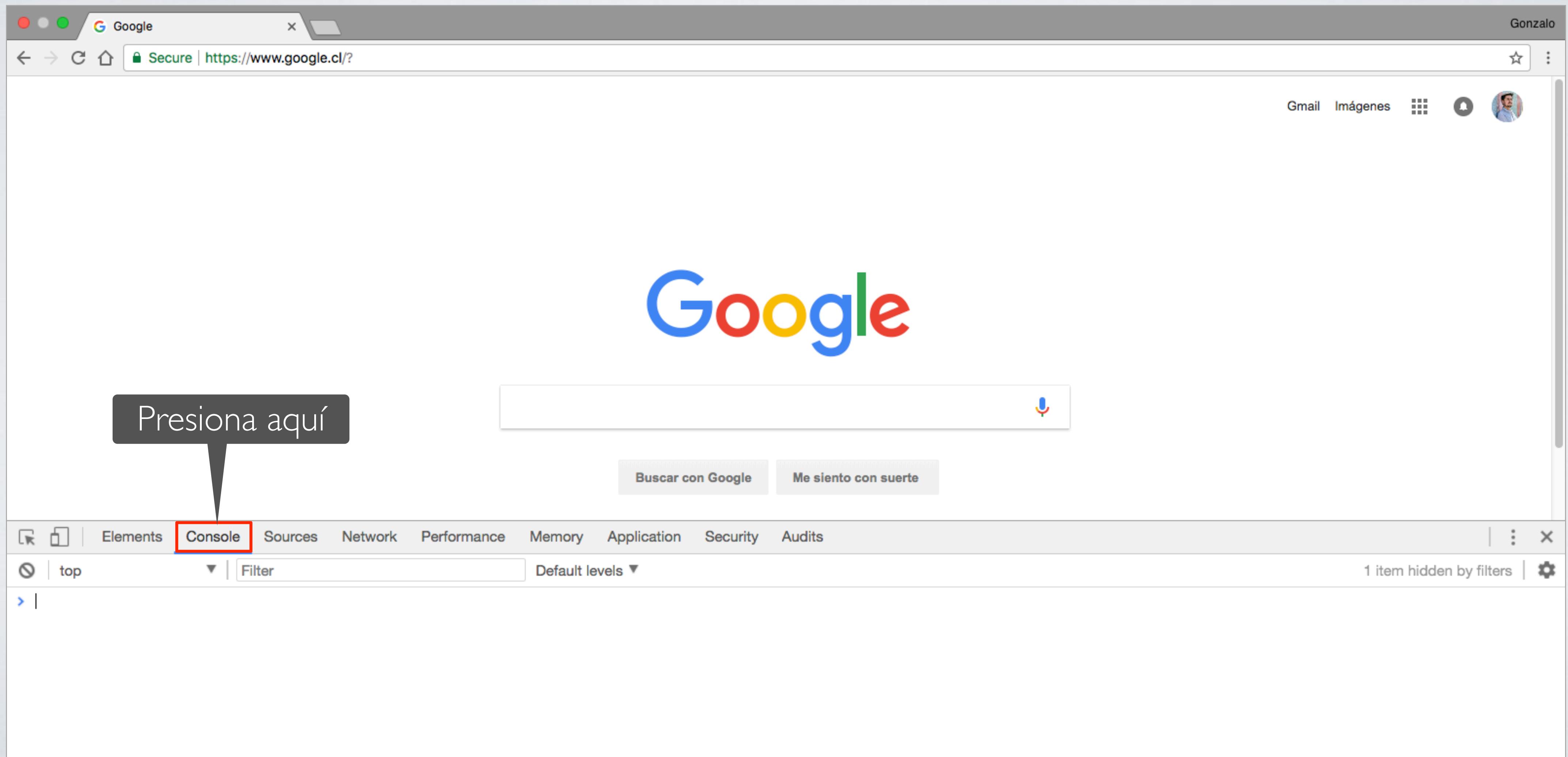
El aviso que viste en la página que creamos es una de las formas que tiene Javascript para interactuar con el usuario. Existen otras formas usando:

CONSOLE.LOG()

```
console.log("¡Hola  
Mundo!");
```

- Esta instrucción nos ayudará a probar nuestros scripts.
- Se usa para mostrar valores de Javascript dentro de la consola del inspector de elementos.

Primero vamos al inspector de elementos y luego a presionemos la pestaña “console”



Escribamos ahora un ¡Hola mundo! para probar

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output window displays the following:

```
> console.log("¡Hola Mundo!");
¡Hola Mundo!
< undefined
```

The output is displayed in blue for the command and gray for the result. The status bar at the bottom of the DevTools window shows the message "1 item hidden by filters".

PREGUNTA

¿Cuál es la diferencia entre console.log y alert?

¿QUÉ HEMOS APRENDIDO HASTA AHORA?

- Hemos aprendido tres formas de agregar javascript:
 1. Con la etiqueta script
 2. De manera inline
 3. Con un script externo
- Hemos aprendido a ocupar alert y console.log

PROMPT()

```
prompt("¿Cuál es tu nombre?");
```

Este método muestra un diálogo con un mensaje,
que solicita al usuario que escriba un texto

Provemos prompt() en el inspector de elementos...

The screenshot shows the Chrome DevTools interface with the "Console" tab selected. The console output is as follows:

```
> prompt("¿Cuál es tu nombre?");
"Gonzalo"
< undefined
```

A modal dialog box is displayed over the DevTools window, titled "www.google.cl says:". It contains the question "¿Cuál es tu nombre?" and a text input field containing the value "Gonzalo". There are "Cancel" and "OK" buttons at the bottom right of the dialog.

On the left side of the main window, there is a dark gray callout box with the text:

El resultado se verá como
"Gonzalo".

PROMPT()

```
nombre = prompt("¿Cuál es tu nombre?");  
console.log(nombre);
```

Podemos guardar el valor para utilizarlo después ocupando una variable

ACTIVIDAD

Escribamos el código que previamente escribimos en el inspector dentro de nuestra página

SOLUCIÓN

1.- CREAR UN ARCHIVO HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Jugando con prompt()</title>
</head>
<body>
    <script></script>
</body>
</html>
```

- Crearemos un nuevo archivo HTML.
- Dentro del <body> agregaremos la etiqueta <script>.

2.- AGREGAMOS EL CÓDIGO DENTRO DE SCRIPT

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Jugando con prompt()</title>
</head>
<body>
  <script>
    nombre = prompt("¿Cuál es tu nombre?");
    console.log(nombre);
  </script>
</body>
</html>
```

- agregamos el prompt pidiendo el nombre y lo mostramos con **console.log()** o **alert()**.

El concepto clave que tenemos que dominar
es el de variables

VARIABLES

```
nombre = prompt("¿Cuál es tu nombre?");
```

Las variables son como un contenedor, pueden almacenar los valores que nosotros queramos

VARIABLES

Con `=` podemos asignar un valor a una variable

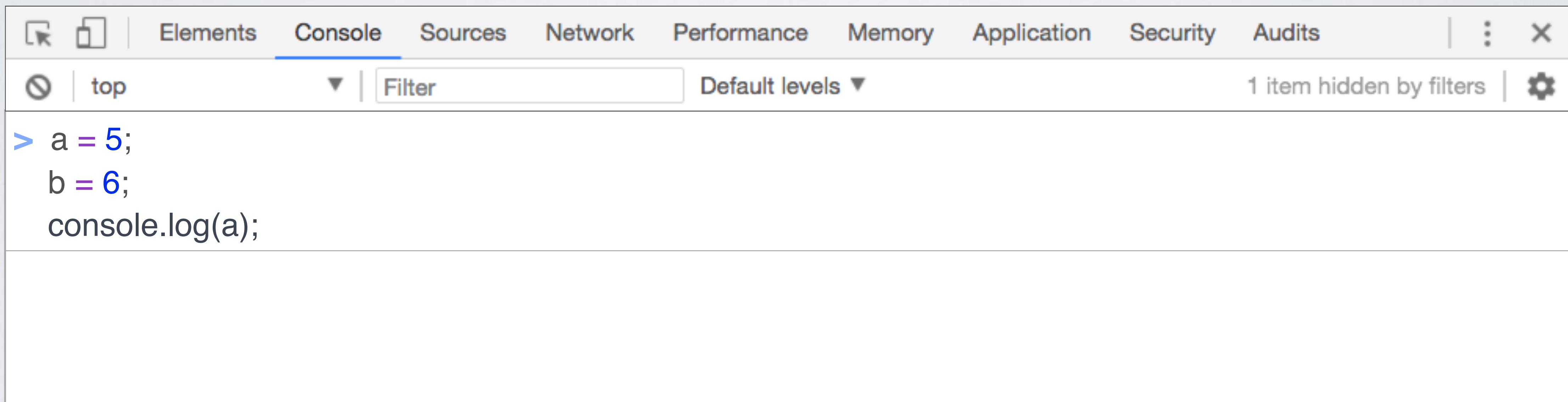
```
a = 6;  
alert(a); // 6
```

Luego, podemos cambiar el valor de un momento a otro, por eso se llaman variables

```
a = 20;  
alert(a); // 20
```

PREGUNTA

¿Qué valor se muestra en el inspector de elementos?



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following code:

```
> a = 5;  
b = 6;  
console.log(a);
```

The output area is currently empty, indicating that the code has not been run.

Para sacarle provecho a las variables tenemos
que aprender sobre operaciones

INTRODUCCIÓN A OPERACIONES

¿QUÉ SUCEDE SI ESCRIBIMOS?

```
console.log(2 + 2);
```

Aquí el símbolo `+` funciona exactamente igual como aprendimos en matemáticas, este es un ejemplo de operador

¿Y SI LO HACEMOS CON VARIABLES?

```
a = 2;  
b = 2;  
console.log(a + b); // 4
```

EXISTEN OTRAS OPERACIONES

```
a = 3;  
b = 3;  
console.log(a * b); // 9
```

¿QUÉ HEMOS APRENDIDO HASTA AHORA?

- Las variables pueden almacenar valores.
- Podemos operar sobre esos valores.

```
a = 6;  
alert(a); // 6  
  
a = 20;  
alert(a); // 20
```

Lo siguiente que aprenderemos es que existen distintos tipos de valores que podemos usar. Estos tipos de valores reciben el nombre de:

TIPOS DE DATOS

LOS TIPOS DE DATOS MÁS COMUNES SON NÚMEROS Y STRINGS

```
number = 120;
```

Number: Cualquier número entero o decimal; positivo o negativo.

```
string = "Hola";
```

String: Cualquier texto encerrado entre comillas dobles o simples.

ACTIVIDAD

Identificar el tipo dato:

a = "Hola";

b = "27";

c = "2.0";

d = 2.0;

RESPUESTAS

```
a = "Hola"; // string
```

```
b = "27"; // string
```

```
c = "2.0"; // string
```

```
d = 2.0; // number
```

Si queremos saber el tipo de dato podemos usar:

TYPEOF

Ejemplos:

```
typeof(2); // "number"
```

```
typeof("2"); // "string"
```

En javascript las operaciones dependen de los tipos de datos

STRING + STRING = STRING

```
"10" + "10"; // 1010  
"hola" + " mundo"; // hola mundo
```

STRING + NUMBER = STRING

```
"Mi nota es un " + 10; // "Mi nota es un 10"  
"10" + 2; // "102"
```

ops !!

Ahora, veremos un ejemplo donde esto es muy importante:

Crearemos un programa donde el usuario ingresa dos valores con un prompt y mostraremos la suma

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Jugando con prompt()</title>
</head>
<body>
<script>
    a = prompt("Ingresar número 1");
    b = prompt("Ingresar número 2");
    console.log(a + b);
</script>
</body>
</html>
```

¿CÓMO PODEMOS HACER PARA SUMAR LOS NÚMEROS?

Respuesta: Fácil transformamos el string a número y luego lo sumamos

PARSEINT()

```
parseInt("365"); // 365
```

transforma un string en un número

Ahora juntemos lo anterior:

```
<script>
  a = prompt("Ingresar número 1");
  a = parseInt(a);

  b = prompt("Ingresar número 2");
  b = parseInt(b);

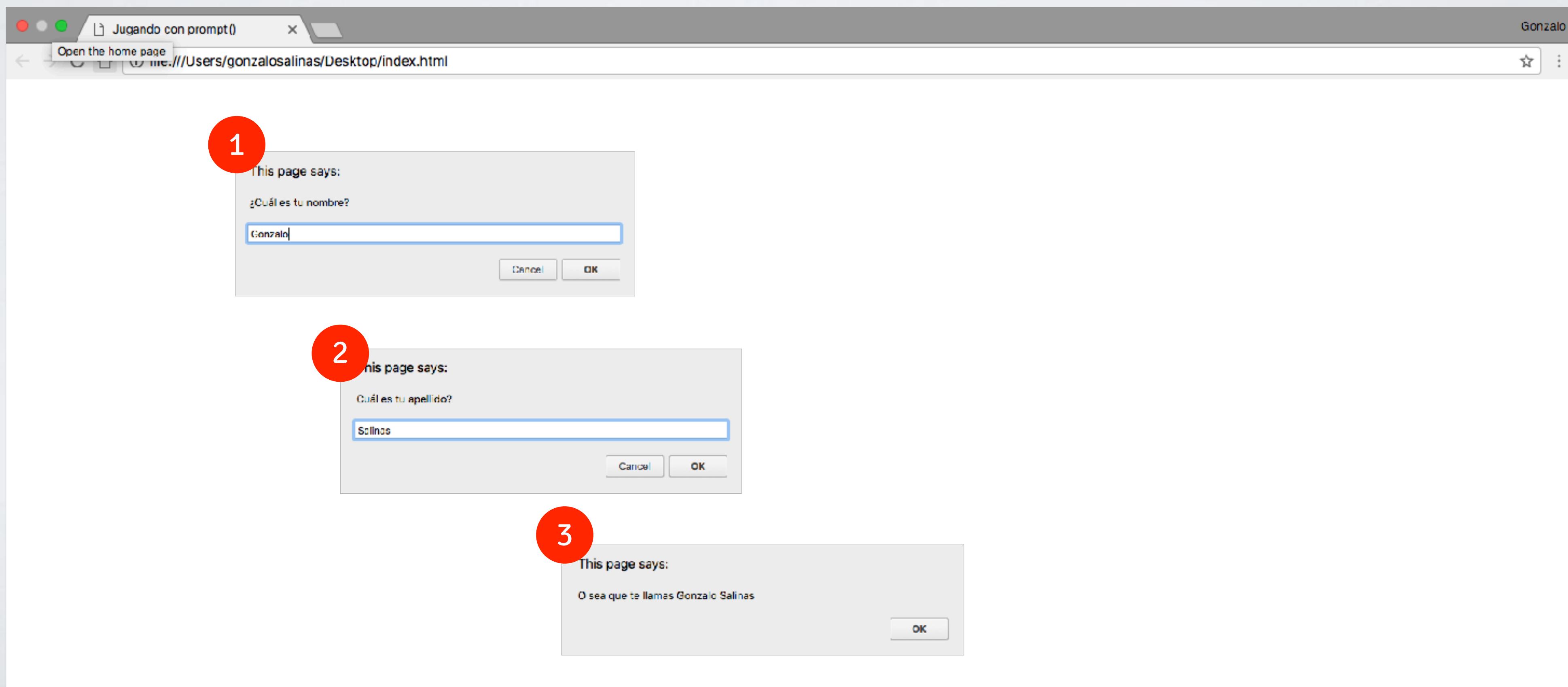
  console.log(a + b);
</script>
```

```
<script>  
  a = prompt("Ingresar número 1");  
  a = parseInt(a);  
  
  b = prompt("Ingresar número 2");  
  b = parseInt(b);  
  
  console.log(a + b);  
</script>
```

- Se ingresa el string 37 (**prompt siempre funciona con strings**) y se guarda en la variable a
- Transformamos el valor anterior (el string 37) a tipo número y lo volvemos a guardar en la variable a.
- Realizamos el paso 1 y 2 con el segundo número.
- Mostramos la suma.

EJERCICIO

Crear un programa javascript que pida el nombre al usuario y lo almacene en una variable, luego, que pida el apellido y lo almacene en otra, y finalmente, debe saludar al usuario ocupando su nombre y apellido.



SOLUCIÓN

```
<script>
    nombre = prompt("¿Cuál es tu nombre?");
    apellido = prompt("¿Cuál es tu apellido?");
    alert("O sea que te llamas " + nombre + " " + apellido);
</script>
```

STRING - NUMBER = NUMBER

Es importante saber que con otros operadores Javascript no convierte los valores a string

```
"10" + 5; // 105  
"10" - 5; // 5
```

Es bueno leer la documentación y hacer pruebas para ir conociendo cada caso.

¿QUÉ HEMOS APRENDIDO HASTA AHORA?

- En las variables podemos guardar distintos tipos de datos
- El resultado de las operaciones dependerá de estos tipos de datos.

EJERCICIOS

¿Cuál es el resultado de las siguientes operaciones?

1

"2" + 3;

2

"2" - 2;

3

5 / 2;

4

4.5 * 2;

5

"2" * 2;

ANATOMÍA DE UN PROGRAMA EN JAVASCRIPT

UNA SENTENCIA

```
miNombre = "Gonzalo";
```

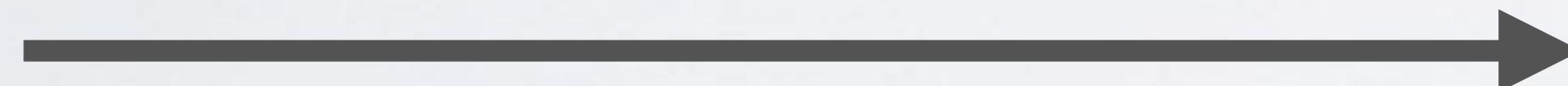
```
└
```

▲
Sentencia

- Las instrucciones son llamadas sentencias.
- Cada sentencia es terminada con punto y coma (**el punto y coma en algunas sentencias es opcional, nosotros lo agregaremos siempre para evitar problemas**).

ANATOMÍA DE JAVASCRIPT

```
miNombre = "Gonzalo" ;
```



- Javascript escanea de izquierda a derecha la sentencia.
- Este transforma la sentencia en una secuencia de elementos que pueden ser tokens, controles de caracteres, espacios en blanco, comentarios, terminaciones entre otras cosas.

ANÁLISIS LÉXICO

O, como un computador lee un lenguaje de programación

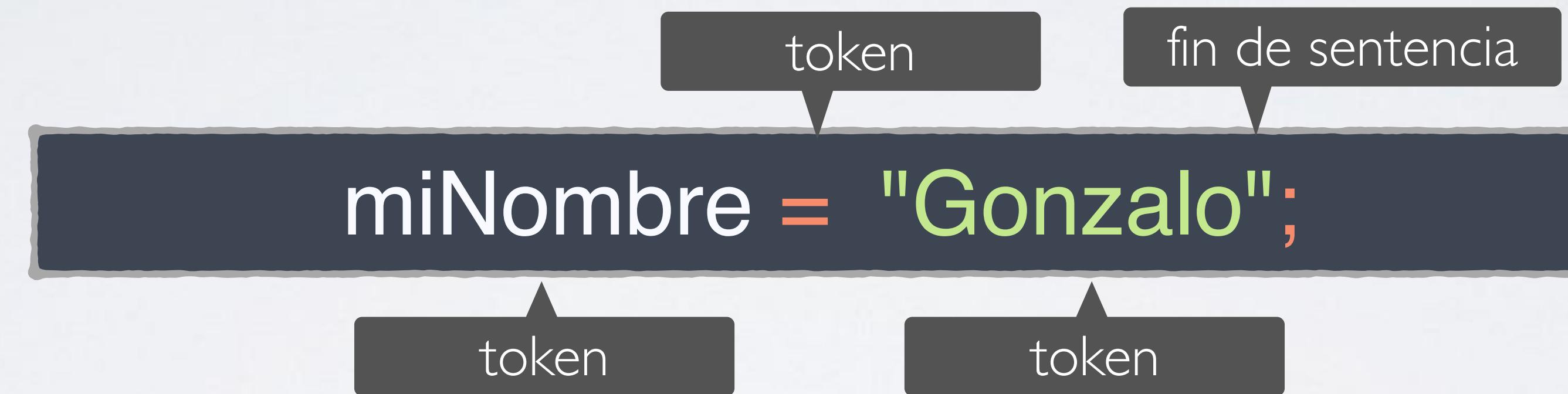
"No es necesario saber esto para programar, pero nos ayudará a entender la sintaxis"

LOS TOKENS

- Son los elementos sintácticos de Javascript.
- Pueden ser palabras reservadas, identificadores, valores literales, puntuadores entre otros.
- **Cosas que no son tokens:** espacios en blanco y comentarios.

TOKENIZANDO

!Esto lo hace javascript de forma automática!



TIPOS DE TOKEN

- Identificadores
- Puntuadores (operadores)
- Literales
- Palabras reservadas
- Valores predefinidos

Los **identificadores** son los nombres de las variables (y de las funciones, lo veremos mas adelante)

```
miNombre = "Gonzalo";
```

Identificador

operador de
asignación

Valor literal

REGLAS DE IDENTIFICADORES

- Deben empezar con una letra, guión bajo (_) o un símbolo dólar (\$).
- No se debe comenzar con números (aunque se pueden agregar en los valores subsiguiente).
- No puede empezar con comillas (ni simples ni dobles)
 - Esto es para que se pueda distinguir una variable de un string
- Se puede comenzar con letras mayúsculas o minúsculas.
- Se pueden usar letras Unicode como ñ o å como identificador.

EJERCICIO

¿Son identificadores válidos o no?

1

40personas = 40;

2

\$Nombre = "Juana Doe";

3

_asbjørn = "nombre noruego";

4

π = 3.1415;

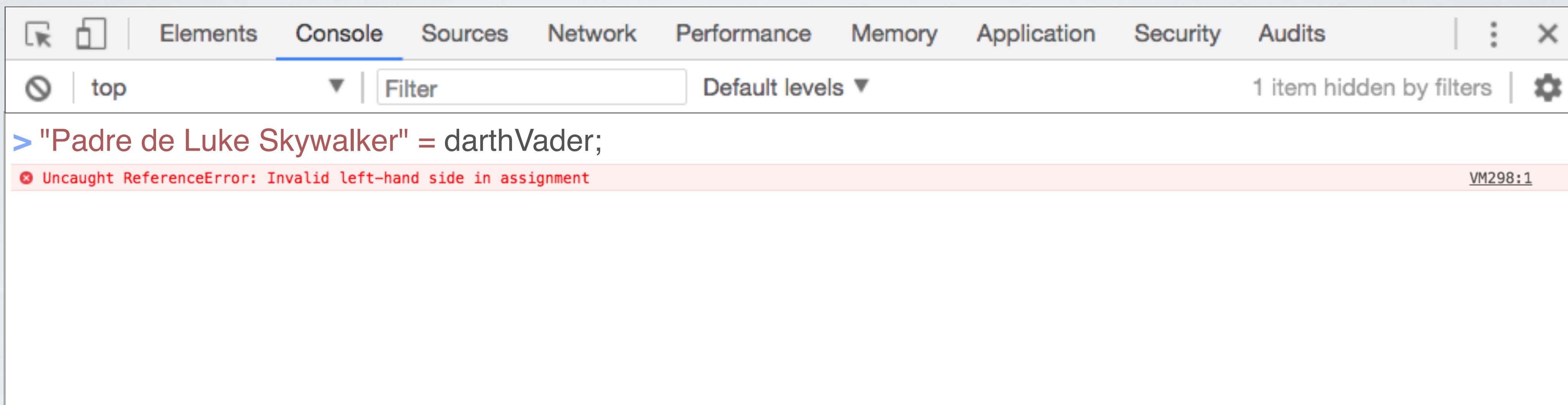
UNA REGLA BÁSICA:

La asignación de identificadores es de izquierda a derecha

```
"Gonzalo" = miNombre ;
```

"Este sería un error ya que el identificador estaría a la derecha"

Probemos en el inspector de elementos:



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output window displays the following content:

```
> "Padre de Luke Skywalker" = darthVader;
✖ Uncaught ReferenceError: Invalid left-hand side in assignment
```

The error message is highlighted in red. The status bar at the bottom right indicates the error occurred at VM298:1.

OTRO DETALLE IMPORTANTE DE JAVASCRIPT:

Javascript es **case-sensitive** (distingue entre mayúsculas y minúsculas).

no es lo mismo:

```
Minombre = "Juan Pérez";
```

```
miNombre = "Juan Pérez";
```

EJERCICIO

¿Por qué esto sería un error?

```
A = 2;  
console.log(a);
```

TIPOS DE TOKEN

- ⦿ Identificadores
- ⦿ **Puntuadores (operadores)**
- ⦿ Literales
- ⦿ Palabras reservadas
- ⦿ Valores predefinidos

TOKEN PUNTUADOR (O DE OPERACIÓN)

```
miNombre = "Gonzalo" ;
```

token de puntuador (operación de asignación)

OTROS TIPOS DE TOKEN PUNTUADORES



TIPOS DE TOKEN

- ☑ Identificadores
- ☑ Puntuadores (operadores)
- **Literales**
 - Palabras reservadas
 - Valores predefinidos

LITERALES

- Los literales son un valor que debe ser leído "**literalmente**" o sea, **cuando decimos 2 es el número 2.**
- Todos los valores que asignamos, o sea, **números o strings son literales.**

¿COMO SE DIFERENCIA UN STRING DE UNA VARIABLE ?

"Gonzalo"

miNombre

RESPUESTA: Los strings siempre están envueltos en comillas (simples o dobles)

¿POR QUÉ ES IMPORTANTE SABER ESTO?

no es lo mismo

```
a = "2";  
console.log(a + 2);
```

que ...

```
a = 2;  
console.log("a" + 2)
```

¿CÓMO SE DIFERENCIA UNA VARIABLE DE UN NÚMERO?

RESPUESTA: Las variables no pueden empezar con un número, esto es para distinguir el token identificador del literal

TIPOS DE TOKEN

- Ⓐ Identificadores
- Ⓐ Puntuadores (operadores)
- Ⓐ Literales
- **Palabras reservadas**
- Valores predefinidos

PALABRAS RESERVADAS

Son palabras claves que realizan funciones propias de javascript. No se pueden utilizar como variables, funciones métodos o identificadores

break	do	instanceof	true
case	else	new	try
catch	false	null	typeof
continue	finally	return	var
debugger	for	switch	void
default	function	this	while
delete	if	throw	with

PALABRAS RESERVADAS OBJETO WINDOW

alert	innerHeight	outerWidth
blur	innerWidth	screen
closed	length	screenX
document	location	screenY
focus	navigator	statusbar
frames	open	window
history	outerHeight	

Existen otros elementos básicos que debemos
conocer como:

COMENTARIOS

Lineas que nos ayudarán a comentar el código

```
// comentario de linea
```

```
/*
Comentario de bloque o
multilinea
*/
```

Existen:

- **Comentarios de línea:** Se usa para comentar en una sola línea.
- **Comentarios de bloque:** Son útiles para comentar código.

Es importante saber que los comentarios no se pueden anidar.

TIPOS DE OPERACIONES

OPERADORES ARITMÉTICOS

Usados para operaciones aritméticas en números

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (es el resto de la división)
++	Incremento
--	Decremento

OPERADORES DE ASIGNACIÓN

Usados para asignar valores a variables

Operador	Ejemplo	Es igual a
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%/	$x %= y$	$x = x \% y$

OPERADORES DE COMPARACIÓN

Usados en declaraciones lógicas para determinar igualdad o diferencia entre valores

Operador	Ejemplo
<code>==</code>	Igual a
<code>==≡</code>	Igual valor e igual tipo
<code>!=</code>	No es igual a
<code>!≡</code>	No es igual a un valor o no es de igual tipo
<code>></code>	Mayor que
<code><</code>	Menor que
<code>>=</code>	Mayor que o igual a
<code><=</code>	Menor que o igual a

OPERADORES LÓGICOS

Usados para determinar la lógica detrás de valores y operaciones

Operador	Descripción
<code>&&</code>	Si ambos operandos devuelven verdadero será verdadero
<code> </code>	Si uno de los dos operandos es verdadero será verdadero
<code>!</code>	Transforma la operación en verdadera si esta es falsa y viceversa

Repasemos lo aprendido, haciendo el primer desafío que se encuentra en la plataforma: “**Actividad: Intro a Javascript**”



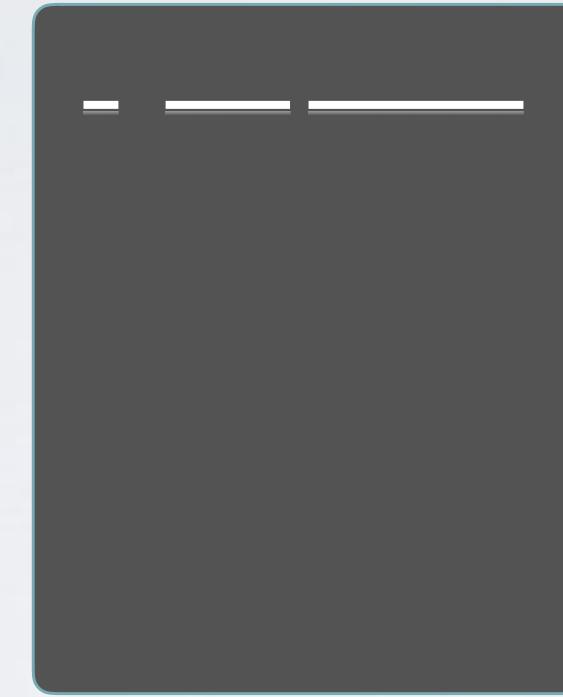
INTRO A JQUERY

¿QUÉ ES JQUERY?



Es una biblioteca que hace más fácil el uso de Javascript en nuestros sitios web.

¿QUÉ NOS PUEDE AYUDAR A HACER?



jQuery toma funcionalidades comunes que requieren varias líneas de código JavaScript y las empaqueta en funciones que podemos utilizar con muy pocas líneas de código



Nos permite manejar de forma sencilla los llamados AJAX y la captura y manipulación de elementos del DOM

Con JQuery puedes hacer páginas web interactivas como esta:



Mas ejemplos en: <https://www.awwwards.com/websites/jquery/>

Para poder hacer cosas geniales con JQuery, primero necesitamos integrarlo a nuestra página web

1.- CREAR EL ARCHIVO HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Usando Jquery</title>
</head>
<body>
  <h1>Sin Jquery</h1>
</body>
</html>
```

- Crearemos un nuevo archivo HTML.
- Escribiremos un título que diga: "Sin Jquery"

2.- INTEGRAR JQUERY

Para integrar JQuery debemos saber:

- Existen dos maneras de integrarlo en nuestros proyectos:
 1. Descargando la biblioteca desde <https://jquery.com/download>
 2. Integrando el CDN de jQuery desde <https://code.jquery.com>
- Nosotros usaremos en este ejemplo el CDN de JQuery

1. Vamos a la página <https://code.jquery.com>.

2. Dentro de la página buscaremos la última versión de JQuery (3.2.1), en su versión minimizada.

3. Copiaremos el código

The screenshot shows a web browser displaying the [jQuery CDN](https://code.jquery.com) website. The address bar at the top indicates a secure connection to <https://code.jquery.com>. The page features the jQuery logo and the tagline "write less, do more." Below the logo, a navigation bar includes links for [jQuery Core](#), [jQuery UI](#), [jQuery Mobile](#), [jQuery Color](#), [QUnit](#), and [PEP](#). The main content area is titled "jQuery CDN - Latest Stable Version" and is powered by [StackPath](#). The "jQuery Core" section shows the latest stable release for each major branch. The "jQuery 3.x" section is highlighted with a red circle labeled "2" and lists "jQuery Core 3.2.1" with links for [uncompressed](#), [minified](#), [slim](#), and [slim minified](#). A "Code Integration" modal window is open, containing the copied code:

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha256-k2WSCIexGzOj3Euiig+TlR8gA0EmPjuc790EeY5L45g=" crossorigin="anonymous"></script>
```

 A red box labeled "3" highlights this code. To the right of the modal, a dark gray box labeled "Versión Minimizada" contains the text "jQuery 3.x" and a list item for "jQuery Core 2.2.4" with links for [uncompressed](#) and [minified](#). The bottom of the page includes sections for "jQuery 2.x", "jQuery 1.x", "jQuery Migrate", and a footer with links to "GitHub", "Issues", "Pull Requests", and "About".

2.- INTEGRAR JQUERY

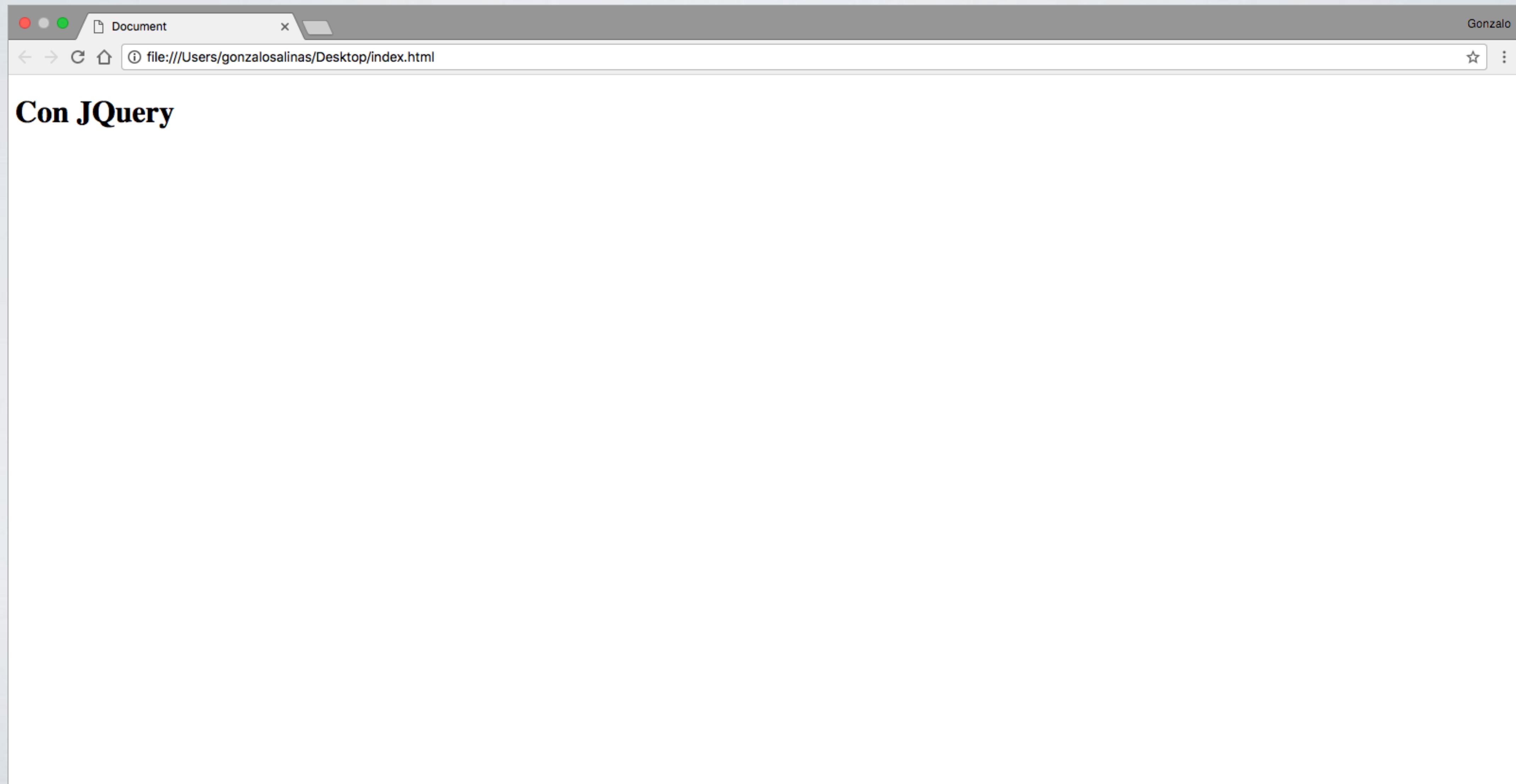
Pega el código debajo del título que escribimos hace un rato

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Usando JQuery</title>
</head>
<body>
  <h1>Sin Jquery</h1>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
</body>
</html>
```

3.- ESCRIBIR EL SCRIPT

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Usando JQuery</title>
</head>
<body>
    <h1>Sin Jquery</h1>
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script>
        $("h1").text("Con JQuery");
    </script>
</body>
</html>
```

4.- VER EL RESULTADO EN EL NAVEGADOR



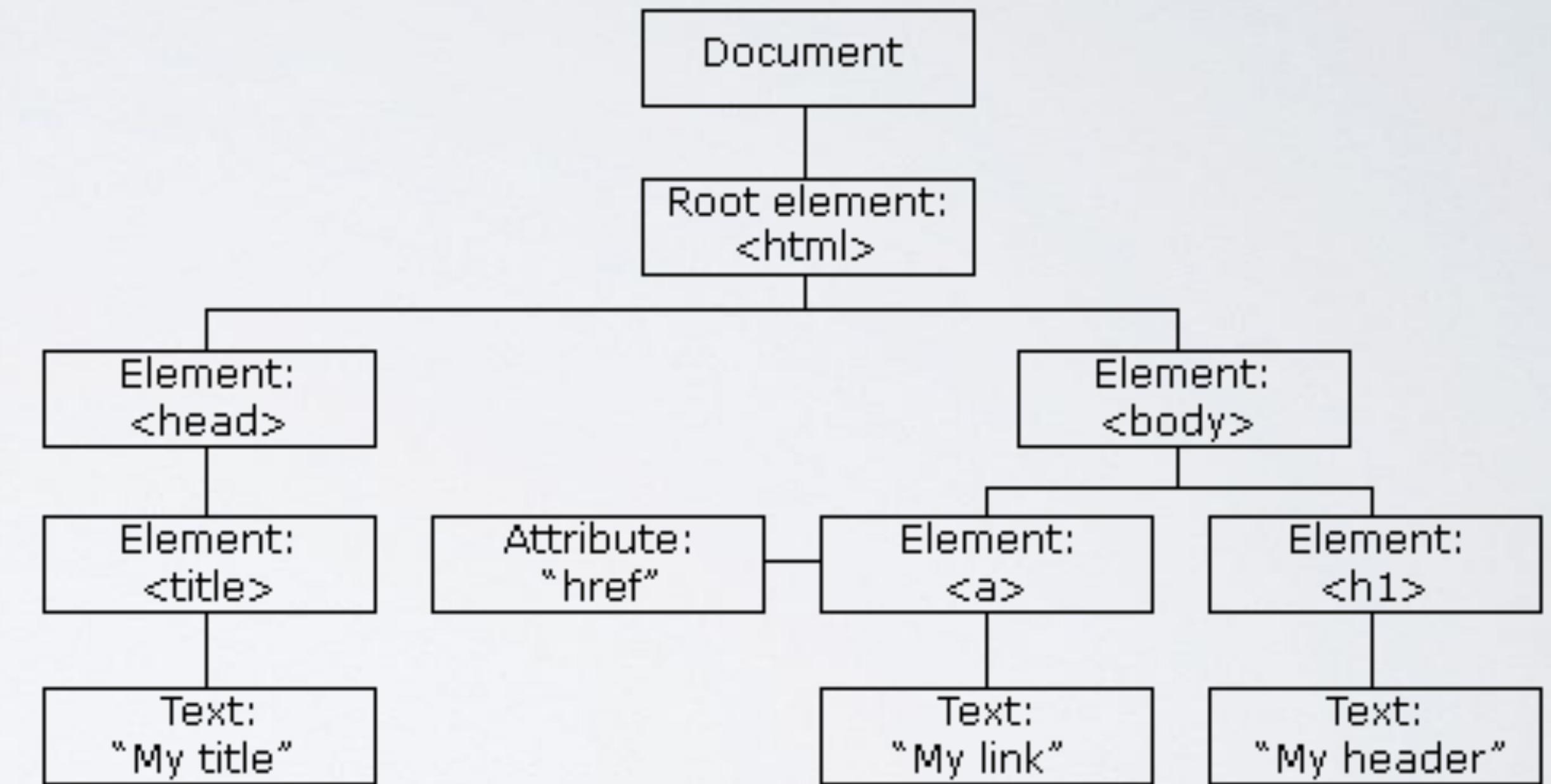
¿QUÉ OCURRIÓ EN LA PÁGINA?

- Si recuerdas, en el paso 1 escribimos un título que decía "**Sin JQuery**".
- Ese título se modificó a "**Con Jquery**" debido a que seleccionamos el elemento `<h1>` de nuestro HTML y luego modificamos el texto utilizando el handler llamado `text()` de JQuery.

La etiqueta <h1> que seleccionamos es parte de un árbol de elementos que nos ayudará a manipular nuestra página web. Este árbol se denomina DOM

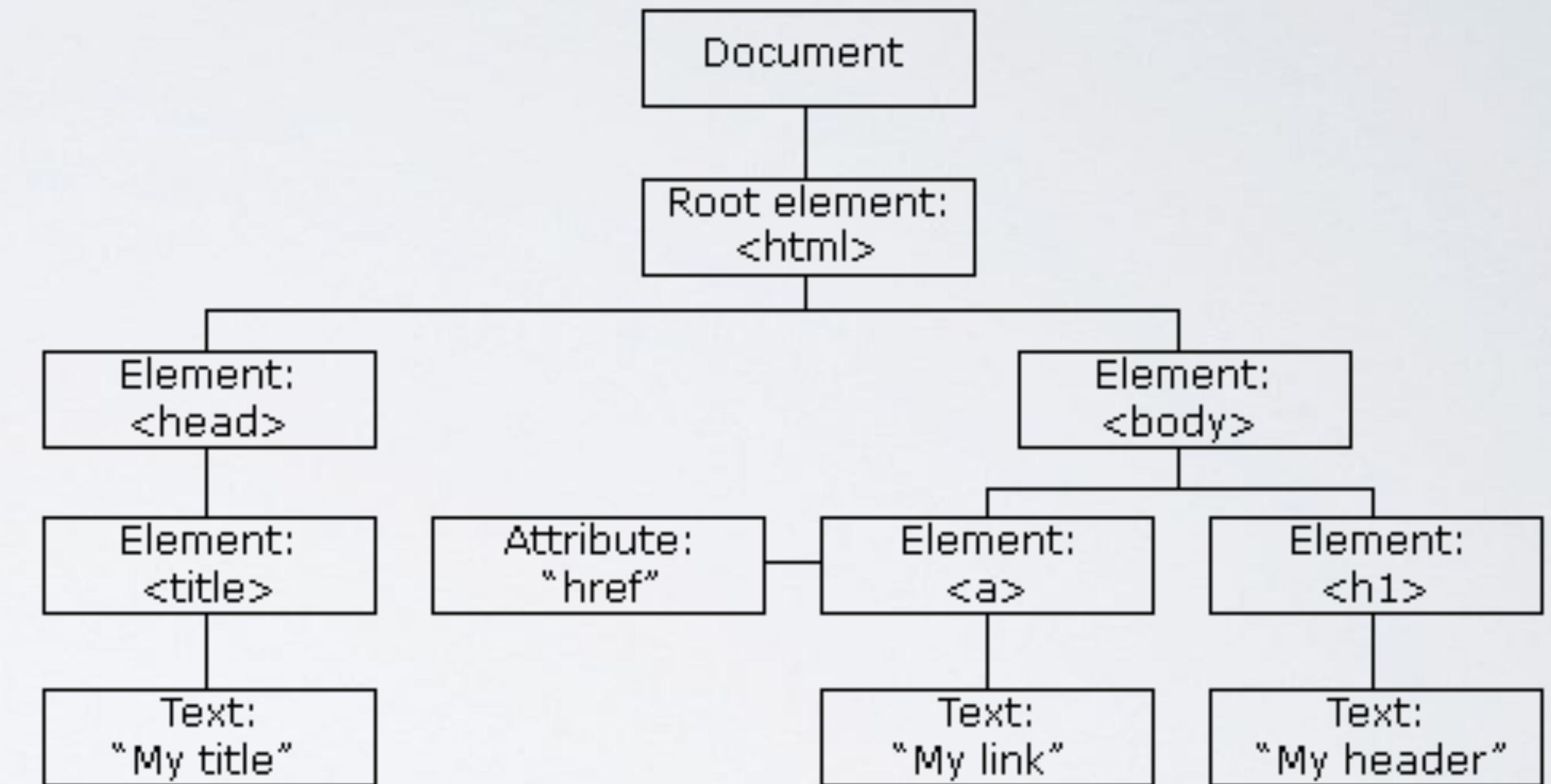
INTRODUCCIÓN AL DOM

- Parte importante del proceso de la carga de una página web es la creación del DOM
- El DOM es como el navegador entiende y relaciona los elementos (las etiquetas) del HTML
- EL DOM puede ser manipulado con javascript. Nos permite añadir, modificar y eliminar contenido de nuestro HTML.



¿QUÉ ES EL DOM?

- Es el acrónimo de Document Object Model (Modelo de Objetos del Documento).
- Es esencialmente una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML.



¿QUÉ PODEMOS LOGRAR MANIPULANDO EL DOM?

- Podemos obtener el valor de los elementos HTML
- Crear nuevos elementos HTML y añadirlos a la página
- Modificar atributos
- Aplicar animaciones a los elementos HTML

Se puede manipular el DOM con javascript sin jquery, pero requiere de más líneas de código y es ligeramente mas complejo

RESUMEN

- El DOM nos permite modificar el contenido de una página dinámicamente con Javascript
- jQuery es una biblioteca para Javascript que nos permite modificar el DOM y crear animaciones con muy poco código
- Para poder ocupar jQuery primero tenemos que integrarlo a nuestro proyecto

MODIFICANDO EL DOM CON JQUERY

LÓGICA PARA RESOLVER PROBLEMAS CON JQUERY

- Utilizar el selector para encontrar el elemento que se desea modificar
- Modificar el elemento

Antes de continuar apliquemos lo aprendido hasta ahora
con JQuery con el siguiente ejercicio:

EJERCICIO

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>¿Qué es JQuery?</title>
</head>
<body>
    <p>¿Qué es Jquery?</p>
</body>
</html>
```

Dado el siguiente código:

1. Integra JQuery vía CDN.
2. Selecciona el párrafo y luego cambia el texto de este por una pequeña reseña de qué es JQuery.

Sigamos conociendo más sobre Jquery

ANATOMÍA DE JQUERY

Aliases

Acción de JQuery
que se realizará en el
elemento.

```
$("selector").handler();
```

Selecciona elementos desde
el DOM

ANATOMÍA DE JQUERY

🕒 Aliases

- Selectores
- Handler

LOS ALIASES SON LAS FORMAS EN LA QUE PODEMOS OCUPAR JQUERY

```
$("selector").handler();
```

```
$()
```

```
jQuery("selector").handler();
```

```
jQuery()
```

ALIASES DE JQUERY

- '\$' suele ser la opción favorita, sin embargo, existen otras bibliotecas que utilizan el espacio de nombre '\$'.
- Si utilizamos una de ellas, junto con jQuery, debemos optar por otro alias para no generar conflictos.

ANATOMÍA DE JQUERY

- Aliases
- Selectores**
- Handler

SELECTORES

- Es una de las funcionalidades más importantes de jQuery.
- Sirven para 'buscar' o 'seleccionar' elementos HTML en función del **id, clase, nombre, atributos, valores, etc.**

SELECTORES

Podemos tener acceso y manipular cualquier elemento del DOM mediante selectores utilizando la sintaxis:

Donde:

```
$("selector").action();
```

- \$ es el signo que define la llamada a jQuery.
- Selector es la 'consulta' que busca uno o mas elementos HTML.
- action() es la acción que se realizará sobre el elemento seleccionado

TIPOS DE SELECTORES

Existen diferentes tipos de selectores que podremos usar para manipular el DOM

Entre ellos están:

1.- Elemento:

```
$("h1").action();
```

2.- #ID:

```
$("#contact").action();
```

3.- .clase:

```
$(".section").action();
```

4.- Atributo:

```
$("[href]").action();
```

SELECTOR DE ELEMENTOS

Selecciona elementos en función del nombre del elemento (**<etiqueta>**)

```
$("p").hide();
```

Por ejemplo: Podemos esconder todos los párrafos **<p>**

SELECTOR DE #ID

- Selecciona un elemento HTML específico en función al **atributo id**.
- Para ello debemos incluir un carácter '#' seguido del **id del elemento**.

```
$("#test").hide();
```

Por ejemplo: Podemos esconder el elemento con ***id="test"***

SELECTOR DE .CLASE

- Selecciona elementos HTML en función a una **clase específica**.
- Para ello debemos incluir un carácter ':', seguido del **nombre de la clase**.

```
$(".awesome").hide();
```

Por ejemplo: Podemos esconder el elemento con **class="awesome"**

SELECTOR DE ATRIBUTO

Para seleccionar un elemento que contenga un atributo debemos incluir el atributo entre **corchetes []**.

```
$("[href]").hide();
```

Podemos esconder todos los elementos que tengan un atributo **href**.

```
$("[href='#']").hide();
```

Podemos esconder todos los elementos que tengan un atributo **href='#'**.

```
$("#a[href='#']").hide();
```

Podemos esconder solo los elementos **<a>** que tengan un atributo **href='#'**.

OTROS SELECTORES

Seletor	Ejemplo	¿Qué hace?
window	<code>\$(window)</code>	Obtiene el objeto principal de javascript, el cual es el viewport.
document	<code>\$(document)</code>	Obtiene el elemento document, el cual es hijo directo de window, de el se puede extraer información como el largo del documento
.	<code>\$(".")</code>	Obtiene todos los elementos
#id	<code>\$("#identificador")</code>	Obtiene al elemento con id="identificador"
.class	<code>\$(".intro")</code>	Obtiene todos los elementos con class="intro"
.class, .class	<code>\$(".bonus, .demo")</code>	Obtiene todos los elementos con class="bonus" o class="demo"
element	<code> \$("p")</code>	Obtiene todos los elementos <p>

EJERCICIO

Modifica el siguiente formulario usando JQuery

Ejercicio: Seleccionando elementos de un formulario
A PEN BY Gonzalo Salinas

First name:

Last name:

Male
 Female

Favorite day of the week:

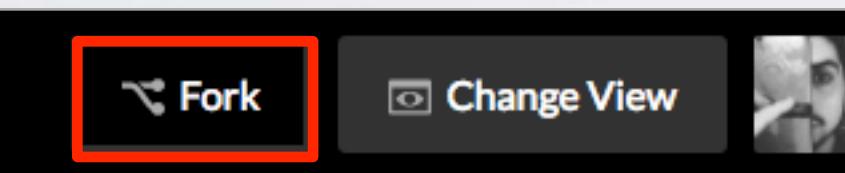
What's your major:

What's your job title:

Do you like challenges:
 Yes
 No

Fork Change View 

¿QUÉ NECESITAS PARA REALIZARLO?



- Ingresa al ejercicio en codepen.io.
- Haz un fork del ejercicio a tu cuenta de Codepen.
- Sigue las siguientes instrucciones:

EJERCICIOS:

1. Pon tu nombre al atributo value del campo **first name**.
2. Pon el valor a la pregunta Favorite **Day of Week** a **Monday**.
3. Cambia la etiqueta de First name a '**Tu nombre**'.
4. Obtén el valor del atributo '**name**' del campo **Favorite Day of The Week**.
5. Escoge la opción **Female** de la pregunta de género.
6. Encuentra la primera form del documento y pon el atributo **name = personal_info**
7. Encuentra la primera form del documento y pon el **atributo name = job_info**

ANATOMÍA DE JQUERY

- 🕒 Aliases
- 🕒 Selectores
- 🕒 Handler

Ahora que sabemos seleccionar elementos del DOM, podemos utilizar handlers para modificarlos

¿QUÉ SON LOS HANDLERS?

Los handlers son métodos que, como su nombre lo indica, nos permiten manipular elementos del DOM.

HANDLER: TEXT()

Podemos **setear/reemplazar el texto** de un elemento del DOM pasando como argumento un string:

```
$("body").text('Este es el texto que estamos añadiendo al elemento  
<body>');
```

Podemos **obtener el texto** de un elemento del DOM al no pasar argumento:

```
$("body").text();  
// 'Este es el texto que estamos añadiendo al elemento <body>'
```

¿Cómo lo hacemos?

Tenemos dos párrafos uno con **id="p1"** y el otro con **id="p2"**.

Queremos copiar el texto del **id="p1"** en **id="p2"**.

RESPUESTA

```
textoCopia = $("#p1").text('Usando el handler text()');  
$("#p2").text(textoCopia);
```

HANDLER: HTML()

Podemos **setear/reemplazar** el contenido HTML de un elemento del DOM pasando como argumento un string:

```
$("body").text('<p>El body ahora contiene este texto dentro de un <span style="font-weight:900">párrafo</span></p>');
```

Podemos **obtener** el HTML contenido en un elemento del DOM al no pasar argumento:

```
$("body").text();  
// "<p>El body ahora contiene este texto dentro de un <span style="font-weight:900">párrafo</span></p>'
```

HANDLER: APPEND()

Si en lugar de reemplazar queremos **agregar contenido**, podemos utilizar el método `.append()`:

```
$("p").append("Estamos agregando un <a href='#'>LINK</a> al  
<strong>body</strong>!");
```

HANDLER: PREPEND()

Si en lugar de agregar contenido al final, queremos **agregar al principio del elemento**, podemos utilizar el método `.prepend()`:

```
$("p").prepend("<h1><u>Título</u> al principio de cada elemento  
párrafo del DOM</h1>");
```

HANDLER: ADDCLASS()

El método `addClass()` nos permite **agregar una o varias clases** (CSS) a los elementos seleccionado.

```
$(".texto").addClass("destacar");
```

Por ejemplo podemos **agregar la clase '.destacar'** a todos los elementos con la clase '.texto'

HANDLER: REMOVECLASS()

El método `removeClass()` nos permite eliminar una o **varias clases (CSS)** de los **elementos seleccionados**

```
$(".destacar").removeClass("texto");
```

Por ejemplo podemos **remover la clase '.texto'** a todos los elementos con la clase **'.destacar'**

HANDLER: CSS()

El método **css()** establece o devuelve **una o más propiedades de CSS para los elementos seleccionados**. Para establecer múltiples propiedades de CSS, use la siguiente sintaxis:

```
$("p").css({"background-color": "red",  
"font-size": "40px"});
```

EJEMPLO

Juguemos con los **handlers vistos** con el siguiente ejemplo:

1.- CREAR EL HTML Y AGREGAR JQUERY

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Jugando con handlers</title>
</head>
<body>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
</body>
</html>
```

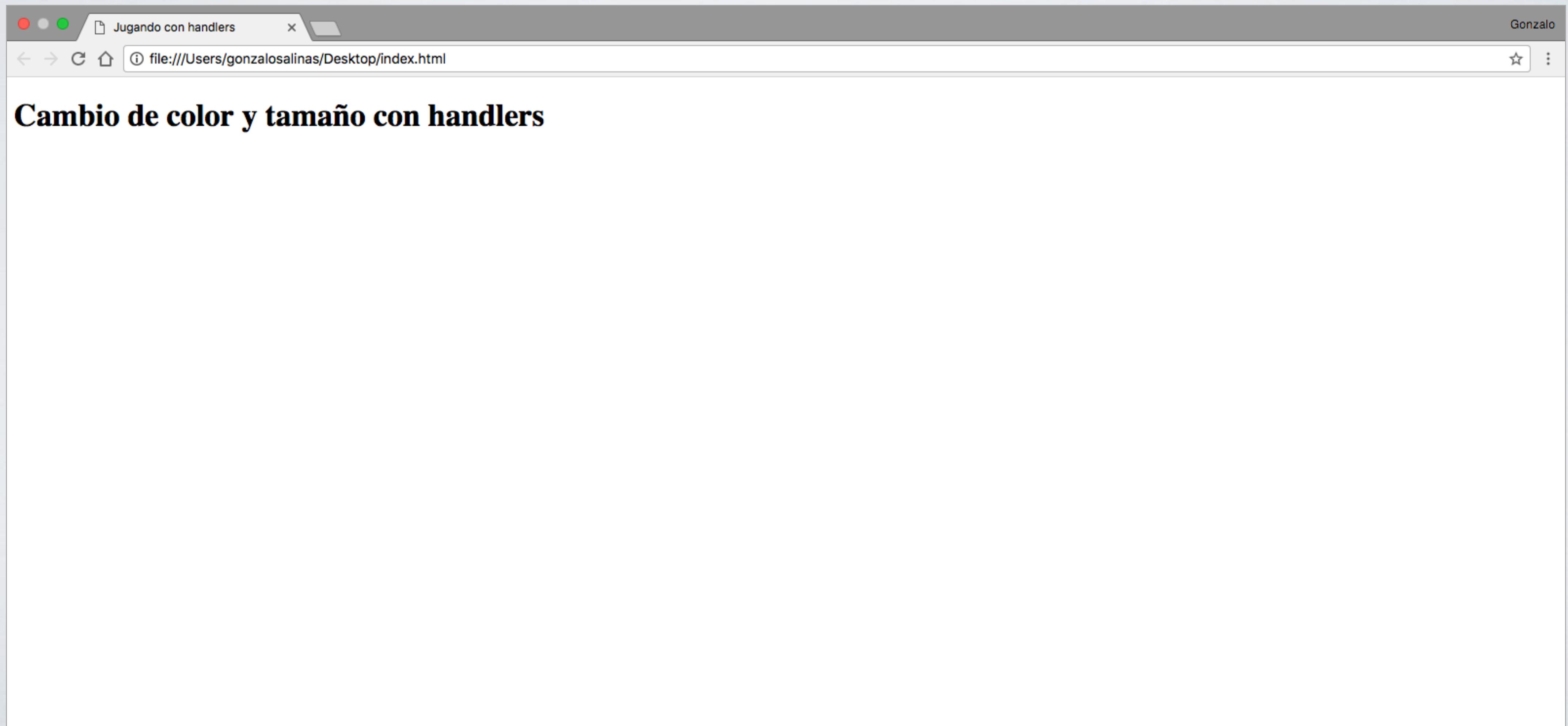
2.- AGREGAR UN POCO DE CSS

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Jugando con handlers</title>
  <style>
    .clase-roja {
      color: red;
      font-family: sans-serif;
      font-size: 1em;
    }
    .clase-azul {
      color: blue;
      font-family: monospace;
      font-size: 2em;
    }
  </style>
</head>
```

3.- ESCRIBIR ALGO DE TEXTO

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Jugando con handlers</title>
</head>
<body>
  <h1>Cambio de color y tamaño con handlers </h1>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
</body>
</html>
```

4.- VER LA PÁGINA WEB EN EL NAVEGADOR

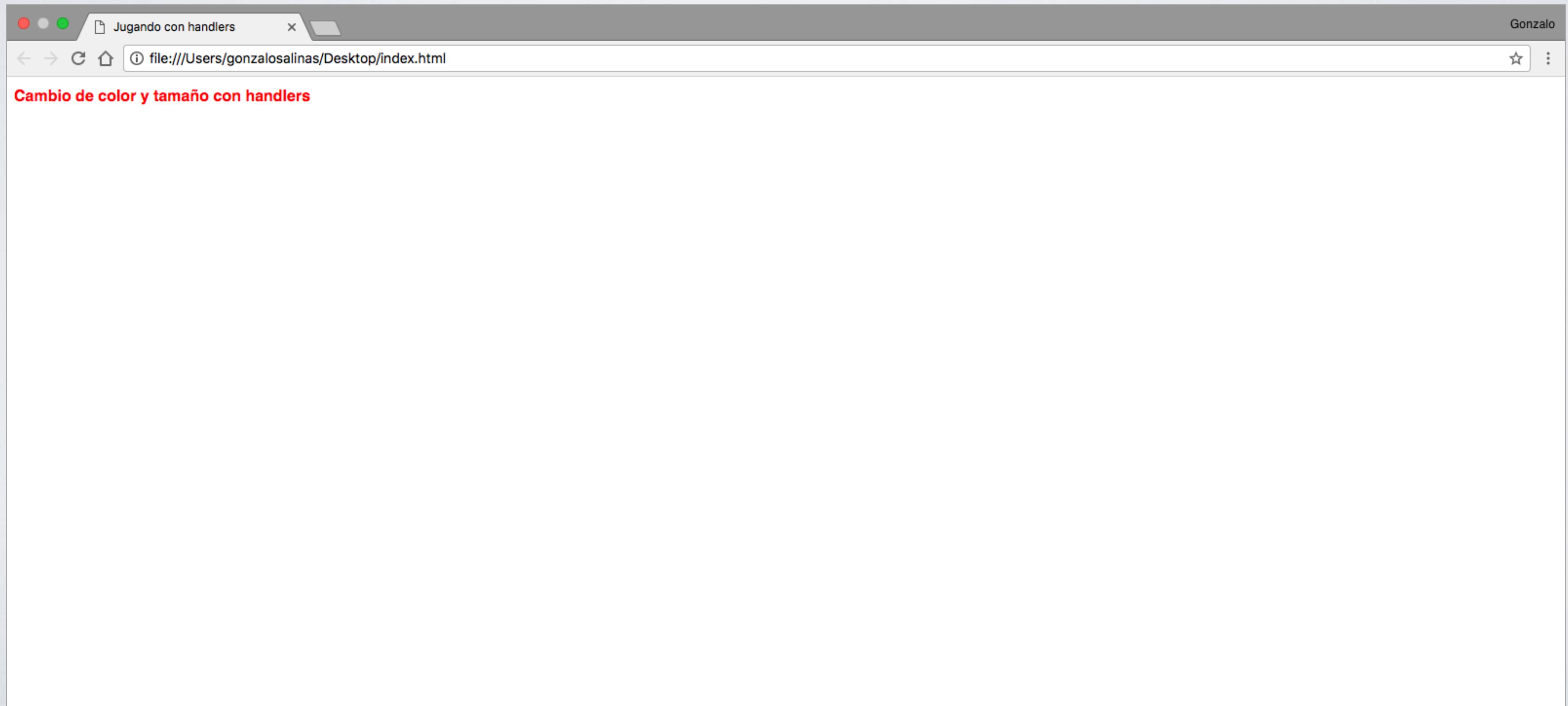


5.- AGREGAR UNA CLASE AL TÍTULO

agreguemos la **clase-roja** al **<h1>** con **addClass()** y veamos que sucede...

```
<body>
  <h1>Cambio de color y tamaño con handlers </h1>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
  <script>
    $("h1").addClass("clase-roja");
  </script>
</body>
```

6.- VER EL CAMBIO EN EL NAVEGADOR



5.- AGREGAR UN SUBTÍTULO CON APPEND()

Ahora creamos un subtítulo debajo del título.

```
<body>
  <h1>Cambio de color y tamaño con handlers </h1>
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
  <script>
    $("h1").addClass("clase-roja");
    $("h1").append('<h3 class="clase-azul">Se creó dinámicamente con
      append()</h3>');
  </script>
</body>
```

6.- VER EL CAMBIO EN EL NAVEGADOR



AHORA TE TOCA A TI

Sigue jugando con los siguientes ejercicios:

1. Cambia el **texto del título** dinámicamente.
2. Modifica la clase del título por "**clase-azul**".
3. Cambia el subtítulo dinámicamente de lugar **dejándolo arriba del título**.

Cabe destacar que **todo lo anterior es realizable mediante JavaScript nativo**. La biblioteca **jQuery sólo nos permite ahorrar código** para poder **programar menos, pero no pensar menos**, es por eso que debemos dominar bien JavaScript para su óptima utilización.

INTRODUCCIÓN A EVENTOS

Jquery además de seleccionar elementos desde el DOM, puede tomar eventos hechos por el usuario dentro de la interfaz como presionar un botón o scrollear

¿QUÉ ES UN EVENTO?

- Todas las diferentes acciones de los visitantes a las que una página web puede responder se llaman eventos.
- Un evento representa el momento preciso en que sucede algo.
 - *Ej: Presionar un botón.*

Existen varios métodos para tomar un evento en JQuery, pero es importante que conozcas los siguientes:

CLICK()

```
$("p").click(function() {  
    alert("esto es un evento");  
};
```

Método usado para obtener un evento al presionar un elemento HTML con el mouse.

\$(DOCUMENT).READY()

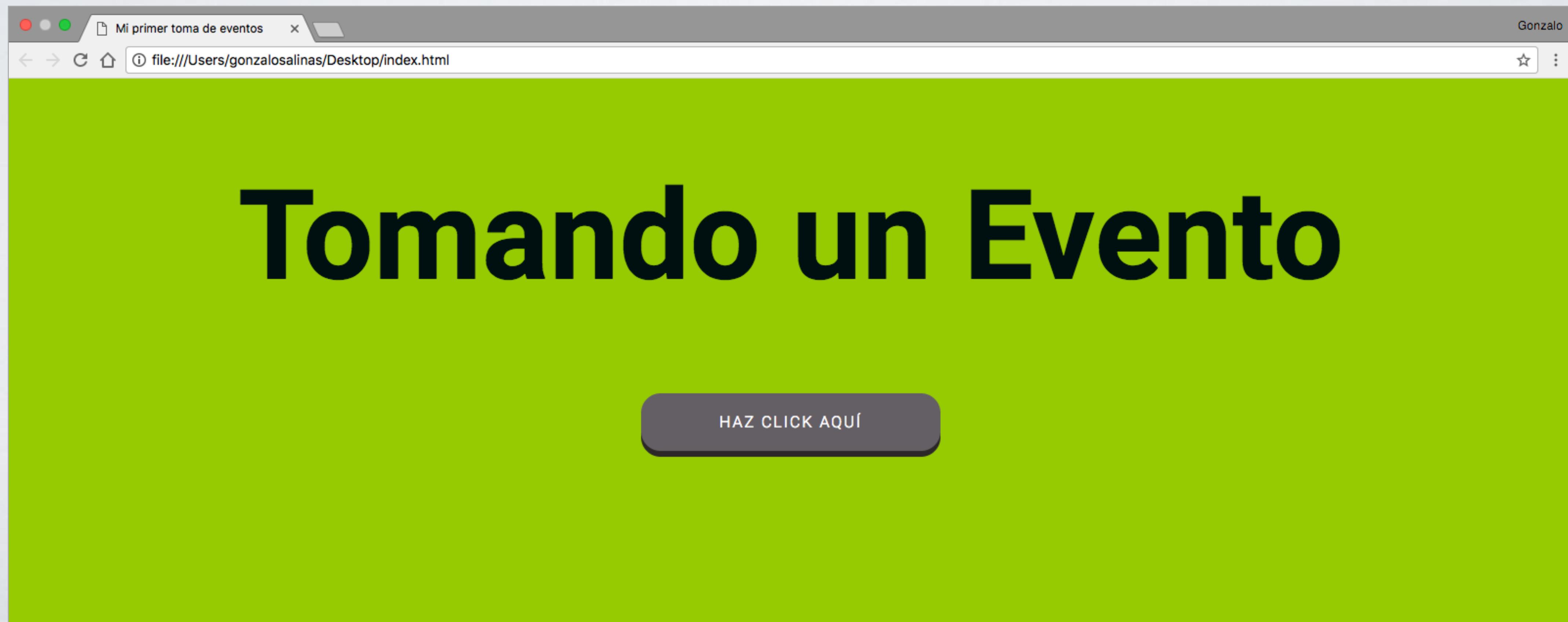
```
$(document).ready(function() {  
    // Métodos de JQuery  
};
```

Este método nos permite ejecutar una función cuando el documento está completamente cargado.

Conozcamos cómo se usan estos métodos con el siguiente ejemplo...

EJEMPLO

Vamos a hacer que el texto crezca al presionar un botón usando JQuery



1.- INGRESA A CODEPEN.IO

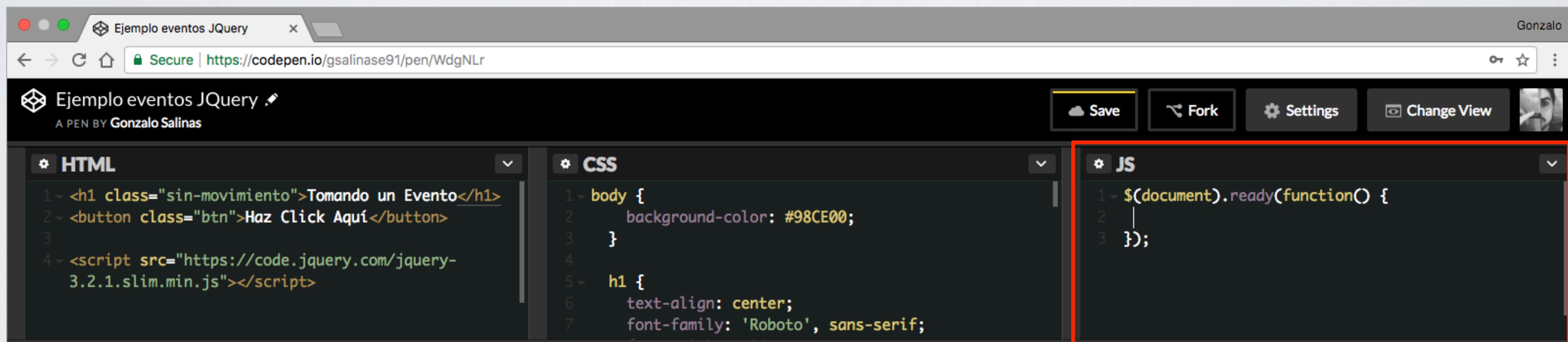
Ingresemos al siguiente link

[codepen.io](#)

En Codepen encontrarás la estructura de nuestra página ya pre diseñada.

2.- AGREGA \$DOCUMENT.READY()

Dentro de Codepen, en la sección "JS" escribiremos el método `$(document).ready()`.



The screenshot shows the CodePen interface with the following details:

- HTML:**

```
1 <h1 class="sin-movimiento">Tomando un Evento</h1>
2 <button class="btn">Haz Click Aquí</button>
3
4 <script src="https://code.jquery.com/jquery-
  3.2.1.slim.min.js"></script>
```
- CSS:**

```
1 body {
2   background-color: #98CE00;
3 }
4
5 h1 {
6   text-align: center;
7   font-family: 'Roboto', sans-serif;
```
- JS:**

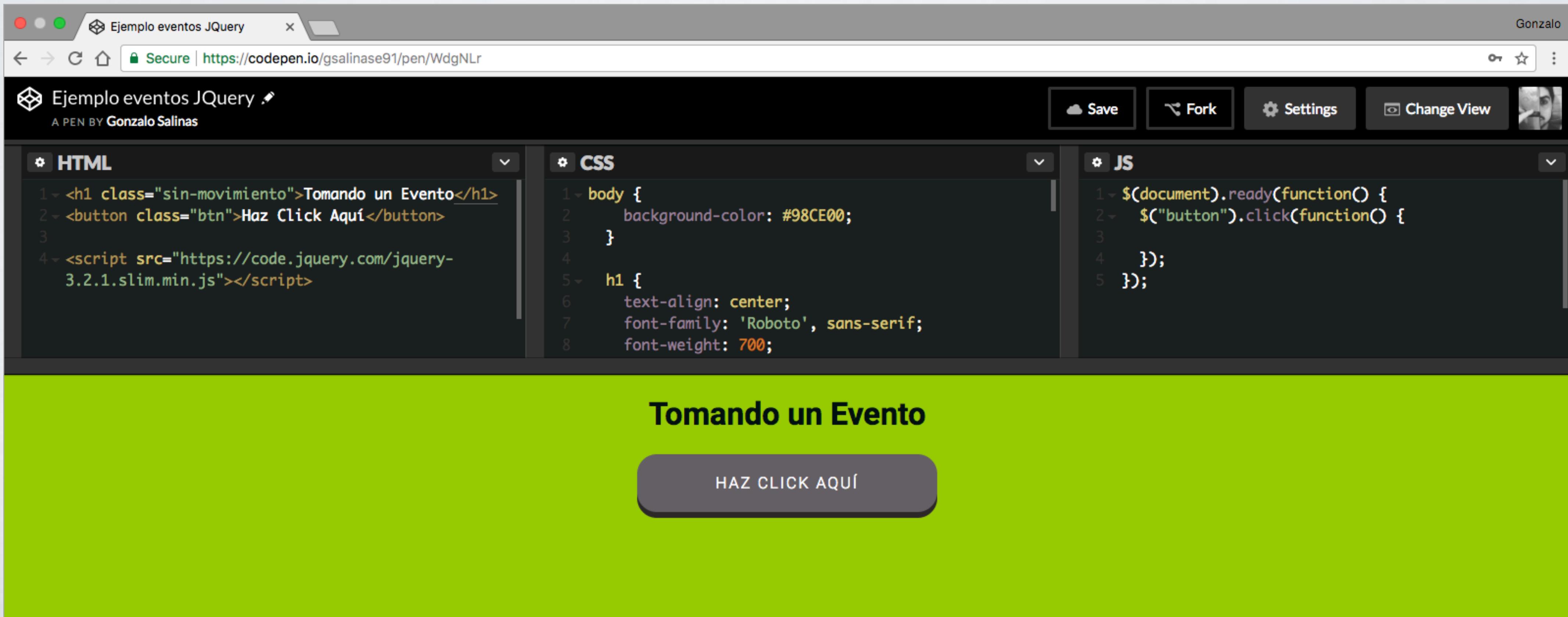
```
1 $(document).ready(function() {
2
3});
```

Tomando un Evento

HAZ CLICK AQUÍ

3.- TOMAR EL EVENTO DEL BOTÓN

Para tomar el evento del click del botón debemos seleccionar el **elemento <button>** y luego **agregar el método click()**



The screenshot shows a CodePen interface with the following details:

- HTML:**

```
1 <h1 class="sin-movimiento">Tomando un Evento</h1>
2 <button class="btn">Haz Click Aquí</button>
3
4 <script src="https://code.jquery.com/jquery-
  3.2.1.slim.min.js"></script>
```
- CSS:**

```
1 body {
2   background-color: #98CE00;
3 }
4
5 h1 {
6   text-align: center;
7   font-family: 'Roboto', sans-serif;
8   font-weight: 700;
```
- JS:**

```
1 $(document).ready(function() {
2   $("button").click(function() {
3
4 });
5});
```

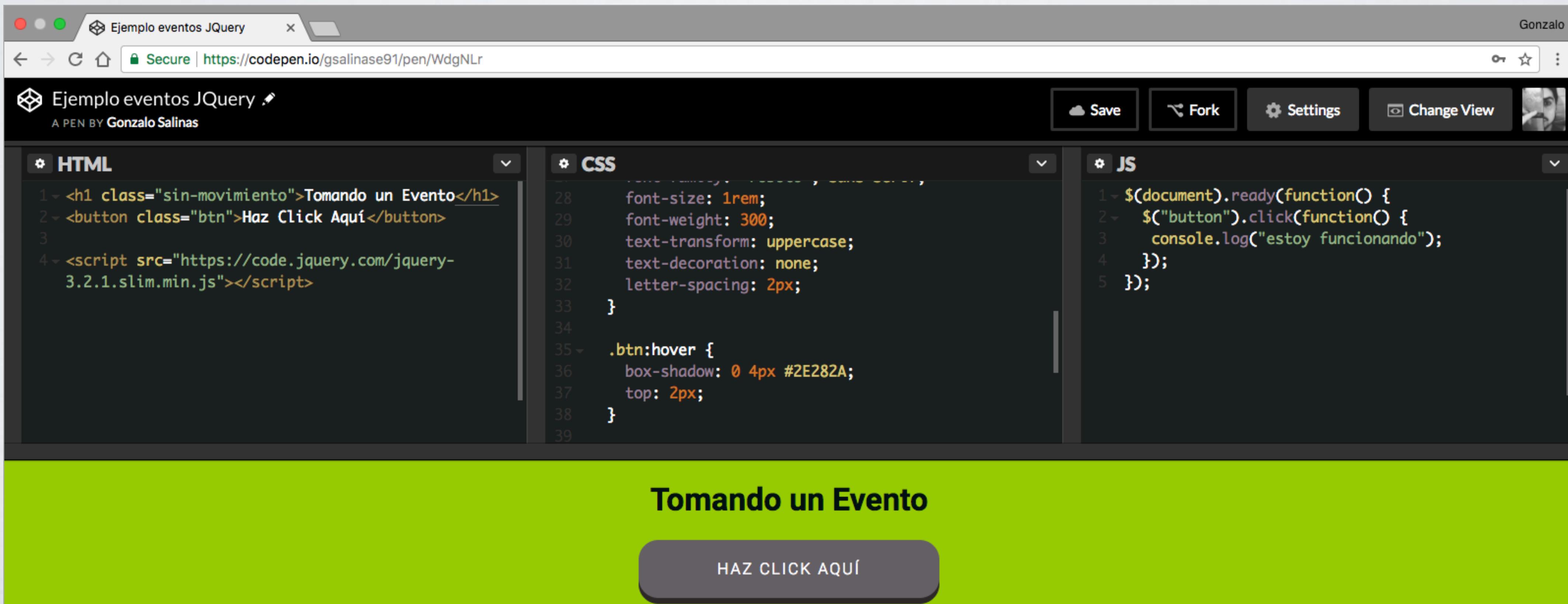
The preview area displays the following content:

Tomando un Evento

HAZ CLICK AQUÍ

4.- PROBAR SI FUNCIONA EL EVENTO

Probemos si funciona el evento agregando dentro del método click() un `console.log` con el siguiente string: "funciona el evento"



The screenshot shows a CodePen interface with the following details:

- Title:** Ejemplo eventos JQuery
- Author:** Gonzalo Salinas
- HTML:**

```
1 <h1 class="sin-movimiento">Tomando un Evento</h1>
2 <button class="btn">Haz Click Aquí</button>
3
4 <script src="https://code.jquery.com/jquery-
  3.2.1.slim.min.js"></script>
```
- CSS:**

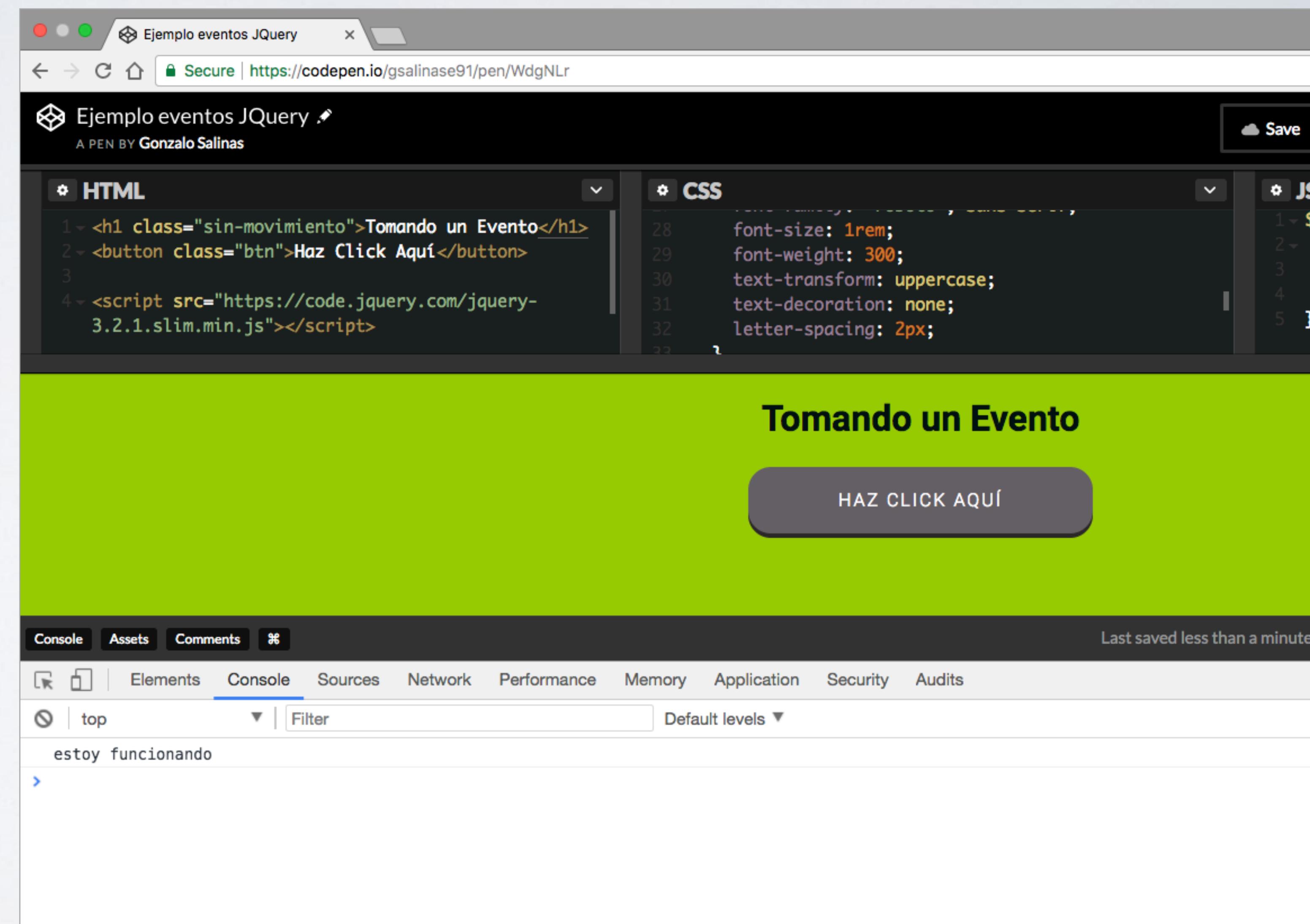
```
28 font-size: 1rem;
29 font-weight: 300;
30 text-transform: uppercase;
31 text-decoration: none;
32 letter-spacing: 2px;
33 }
34
35 .btn:hover {
36   box-shadow: 0 4px #2E282A;
37   top: 2px;
38 }
39
```
- JS:**

```
1 $(document).ready(function() {
2   $("button").click(function() {
3     console.log("estoy funcionando");
4   });
5 });
```

The preview area at the bottom shows the heading "Tomando un Evento" and a button labeled "HAZ CLICK AQUÍ".

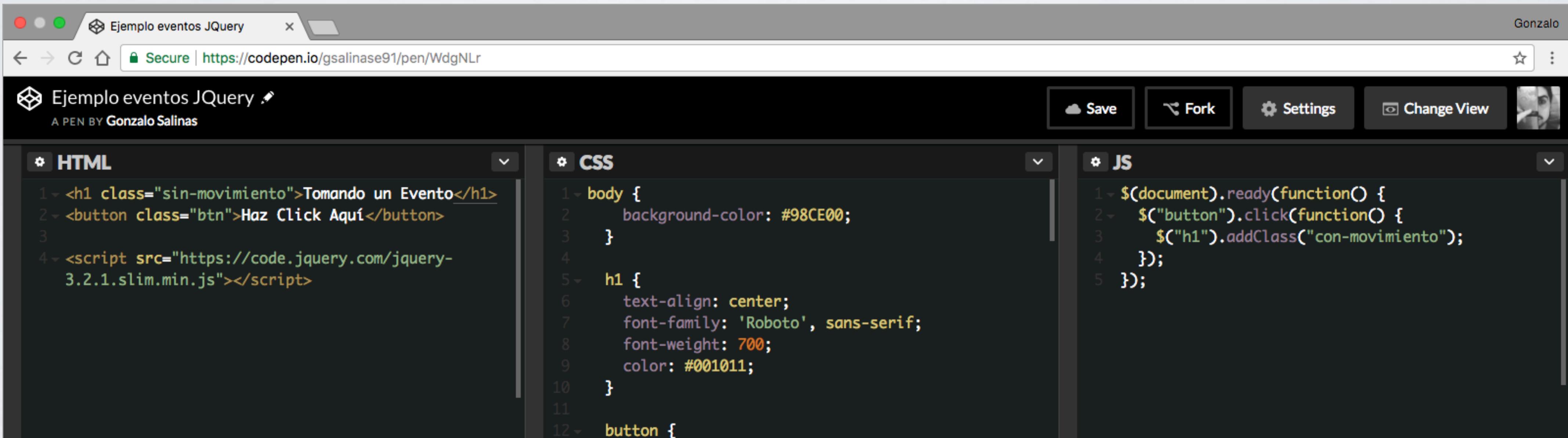
5.- PROBAR EL EVENTO EN EL INSPECTOR DE ELEMENTOS

1. Vamos a la consola del inspector de elementos
2. Presionemos el botón "haz click aquí".
3. Si aparece en la consola el texto "estoy funcionando, quiere decir que vamos muy bien.



6.- AGREGAR LA CLASE CON-MOVIMIENTO

Ahora que sabemos que funciona el botón, eliminaremos el console.log y agregaremos en su lugar un addClass() que afecte directamente al elemento <h1>



The screenshot shows a CodePen interface with the following details:

- HTML:**

```
1 <h1 class="sin-movimiento">Tomando un Evento</h1>
2 <button class="btn">Haz Click Aquí</button>
3
4 <script src="https://code.jquery.com/jquery-
3.2.1.slim.min.js"></script>
```
- CSS:**

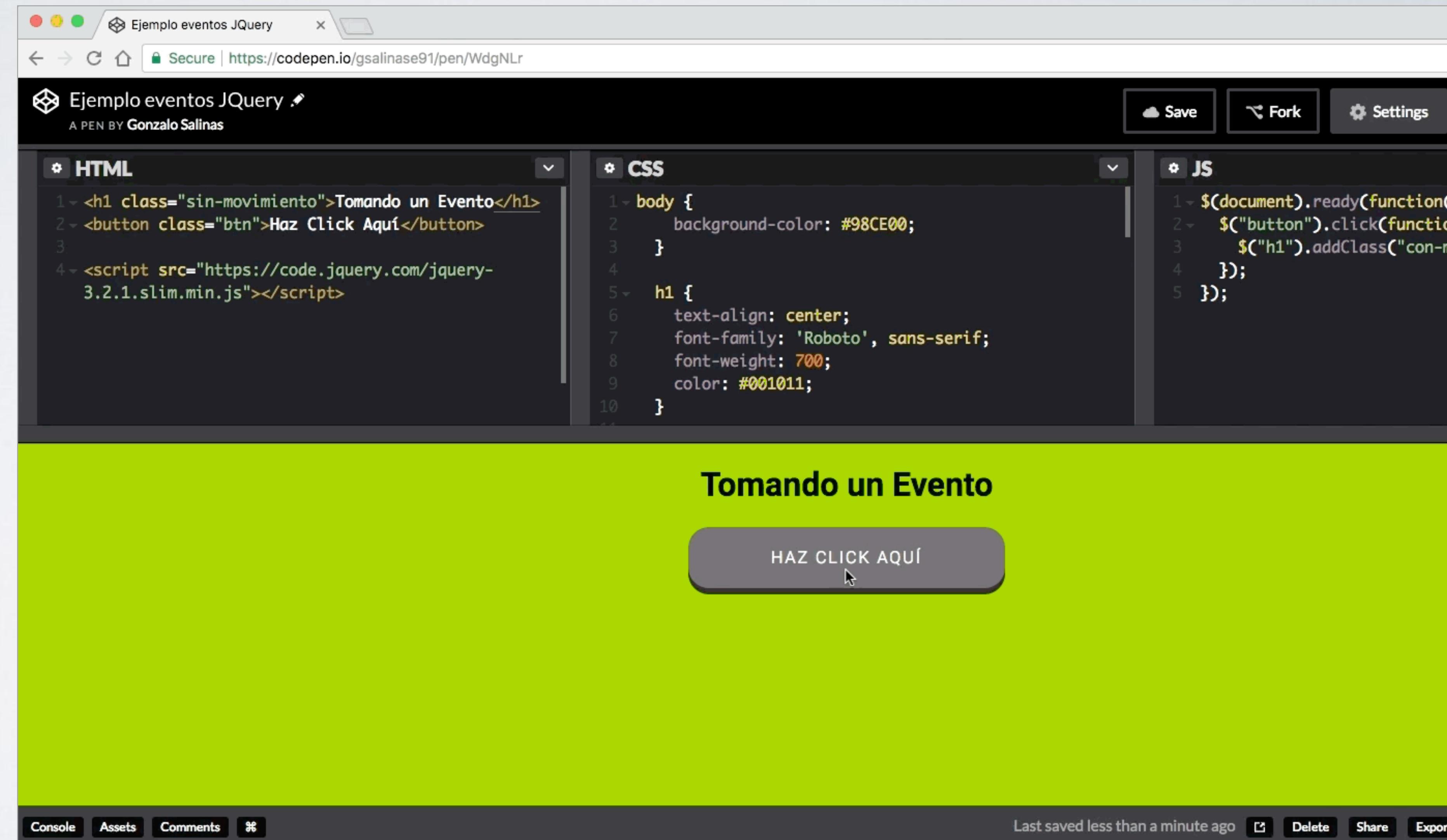
```
1 body {
2   background-color: #98CE00;
3 }
4
5 h1 {
6   text-align: center;
7   font-family: 'Roboto', sans-serif;
8   font-weight: 700;
9   color: #001011;
10 }
11
12 button {
```
- JS:**

```
1 $(document).ready(function() {
2   $("button").click(function() {
3     $("h1").addClass("con-movimiento");
4   });
5 });
```

The preview area at the bottom shows the text "Tomando un Evento" in a large, bold, black font centered on the page. A button below it says "HAZ CLICK AQUÍ".

7.- PROBEMOS SI FUNCIONA EL EJEMPLO

Para finalizar presionemos el botón para ver que pasa... Si el texto creció quiere decir que el ejemplo fue todo un éxito



The screenshot shows a CodePen interface with the title "Ejemplo eventos JQuery" by Gonzalo Salinas. The interface is divided into three main sections: HTML, CSS, and JS.

- HTML:**

```
1 <h1 class="sin-movimiento">Tomando un Evento</h1>
2 <button class="btn">Haz Click Aquí</button>
3
4 <script src="https://code.jquery.com/jquery-
3.2.1.slim.min.js"></script>
```
- CSS:**

```
1 body {
2   background-color: #98CE00;
3 }
4
5 h1 {
6   text-align: center;
7   font-family: 'Roboto', sans-serif;
8   font-weight: 700;
9   color: #001011;
10 }
```
- JS:**

```
1 $(document).ready(function() {
2   $("button").click(function() {
3     $("h1").addClass("con-
4   });
5 });
```

The preview area shows the resulting page. It features a green background. In the center, there is an

element with the text "Tomando un Evento". Below it is a button with the text "HAZ CLICK AQUÍ". The button has a dark gray background and a white border. A cursor arrow is positioned over the button, indicating it is ready to be clicked.

Sigamos repasando con el siguiente desafío de JQuery que se encuentra en la plataforma Empieza.