

JavaScript (Parte I)

Introducción a JavaScript

En esta unidad aprenderemos sobre un lenguaje de programación interpretado muy importante en el desarrollo web: **JavaScript**.



Imagen 1. Logo de JavaScript.



Imagen 2. Tipos de lenguajes de programación.

- Desde un punto de vista simple, HTML sería el esqueleto del personaje, la estructura.
- Por otro lado, CSS serviría para darle apariencia al personaje.
- Finalmente, JavaScript sería la animación del personaje, el comportamiento o la interacción que éste pueda tener.

En el desarrollo de sitios web la interactividad y el dinamismo es clave. JavaScript nos ayuda a generar interacción con el usuario, efectos de estilo dinámicos, animaciones y mucho más.

Algunos ejemplos de JavaScript

Existen millones de sitios utilizando JavaScript, veamos algunos ejemplos.

1. [filippobello](#)
2. [promoflex](#)
3. [legworkstudio](#): Cada interacción de este sitio activa un efecto que es visible por el usuario, y como te puedes dar cuenta, todo ocurre sin tener que recargar la pagina.
4. Cualquier sitio que contenga **Bootstrap** está utilizando JavaScript. Este framework utiliza una biblioteca llamada **jQuery**, que nos permite entregar el dinamismo y los efectos visuales a sus componentes.

Veamos el siguiente ejemplo:

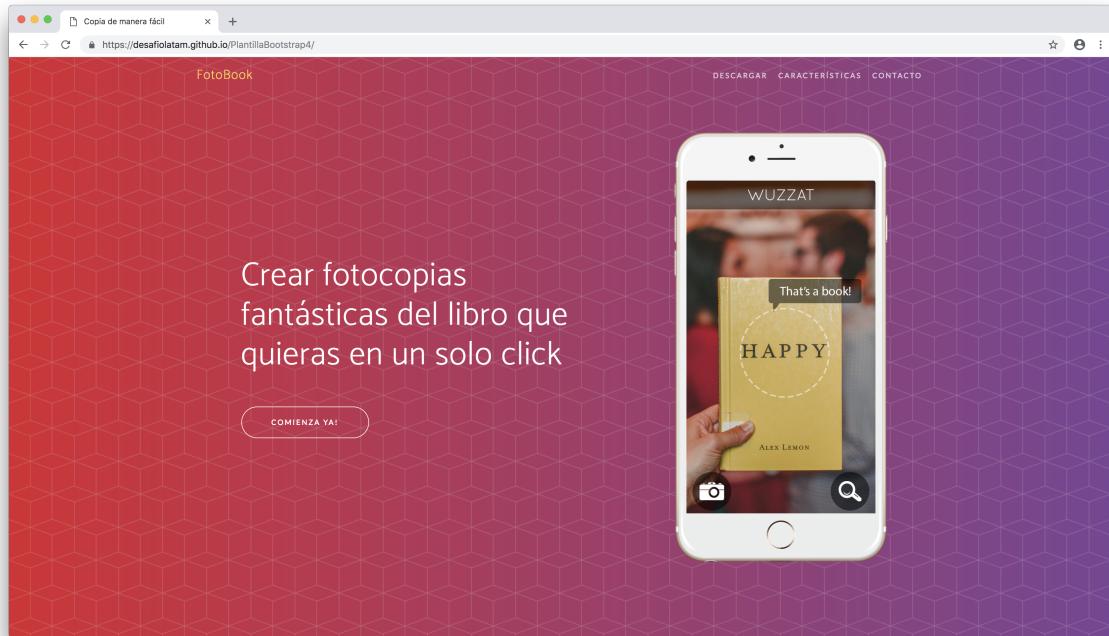


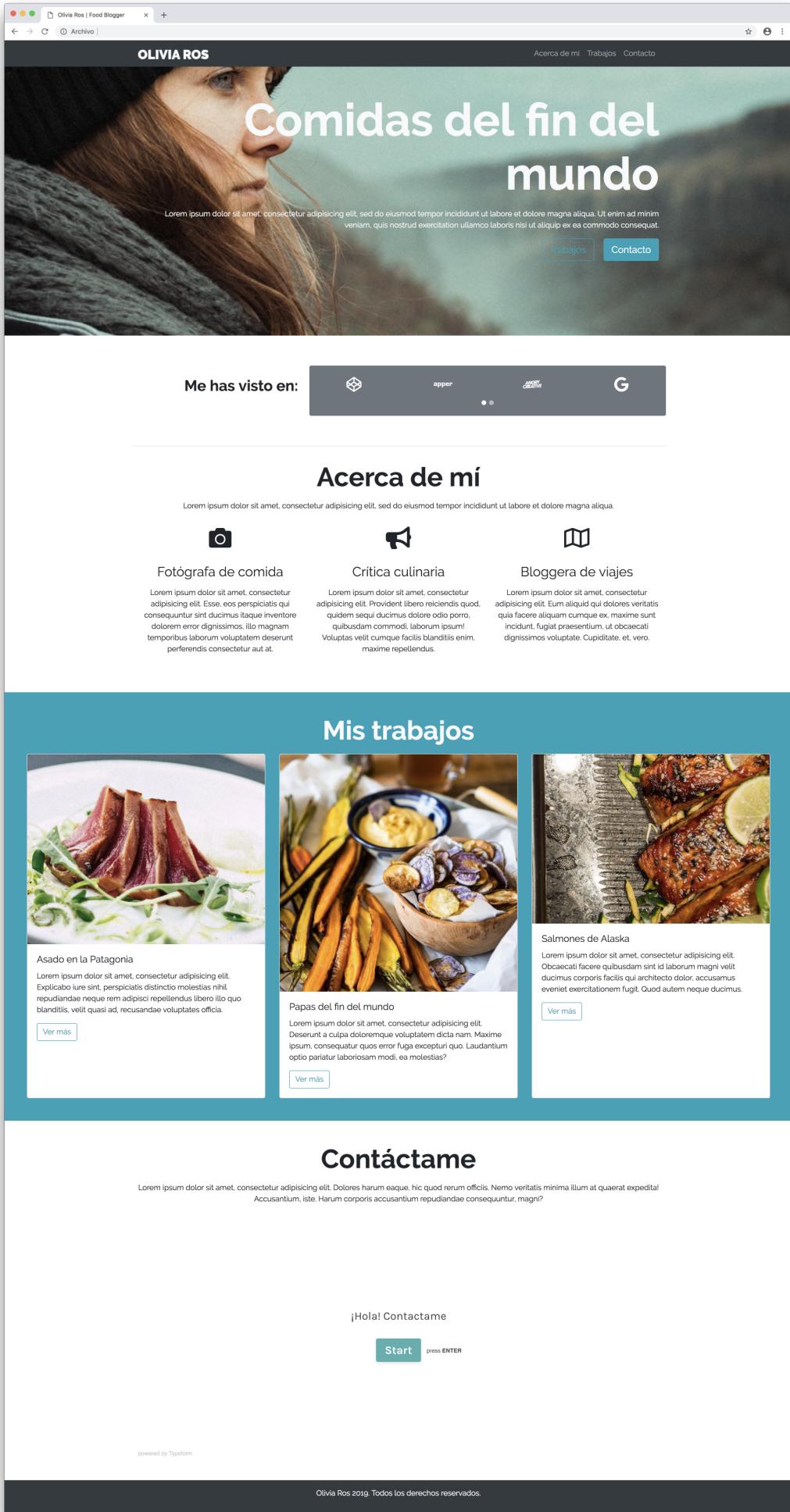
Imagen 3. Ejemplo de Bootstrap con JavaScript y jQuery.

En esta página todos los efectos, cambios de color en los botones y los desplazamientos son gracias a los elementos de **JavaScript** y **jQuery** que contiene **Bootstrap**.

¿Qué faremos en esta unidad?

En esta unidad vamos a aprender conceptos básicos de JavaScript, luego, comenzaremos a aprender utilizar la biblioteca jQuery y también aprovecharemos algunos componentes de Bootstrap que utilizan estos elementos para darle vida al proyecto de Olivia Ros.

Lo que lograremos será lo siguiente:

Olivia Ros | Food Blogger

OLIVIA ROS

Acerca de mí Trabajos Contacto

Comidas del fin del mundo

Trabajos Contacto

Me has visto en:

apprer ANDY Creative G

Acerca de mí

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Fotógrafa de comida

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Esse, eos perspicatis qui consequuntur sint ducimus itaque inventore dolorem error dignissimos, illo magnam temporibus laborum voluptatem deserunt perferendis consectetur aut at.

Critica culinaria

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident libero recilendis quod, quidem sequi ducimus dolore odio porro, quibusdam commodi, laborum ipsum! Voluptas velit cumque facilis blanditiis enim, maxime repellendus.

Bloggera de viajes

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum aliquid qui dolores veritatis quia facere aliquam cumque ex, maxime sunt incident, fugiat praesentium, ut obcaecati dignissimos voluptate. Cupiditate, et, vero.

Mis trabajos


Asado en la Patagonia
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Explicabo iure sint, perspicatis distinctio molestias nihil repudiandae neque rem adipisci repellendus libero illo quo blanditiis, velit quasi ad, recusandae voluptates officia.
[Ver más](#)


Papas del fin del mundo
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Deserunt a culpa doloremque voluptatem dicta nam. Maxime ipsum, consequatur quos error fuga excepturi quo. Laudantium optio paratur laboriosam modi, ea molestias?
[Ver más](#)


Salmones de Alaska
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Obcaecati facere quibusdam sint id laborum magni velit ducimus corporis facilis qui architecto dolor, accusamus eveniet exercitationem fugit. Quod autem neque ducimus.
[Ver más](#)

Contáctame

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolores harum eaque, hic quod rerum officiis. Nemo veritatis minima illum at quiaerat expedital Accusantium, iste. Harum corporis accusantium repudiandae consequuntur, magni?

¡Hola! Contactame

Start press ENTER

powered by Typeform

Olivia Ros 2019. Todos los derechos reservados.

Imagen 4. Vista Final de Proyecto.

Haremos que al clickear los ítems del menú se haga un scroll suave.

Le incorporaremos ventanas modales.

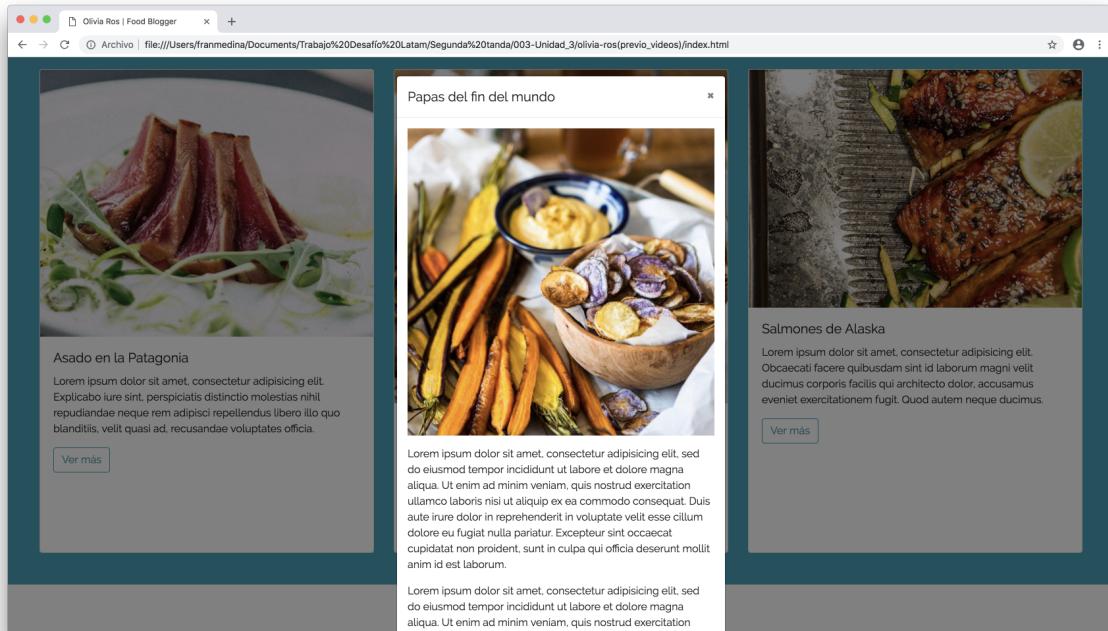


Imagen 5. Vista final de Proyecto - Ventanas Modales.

También un carousel de íconos.

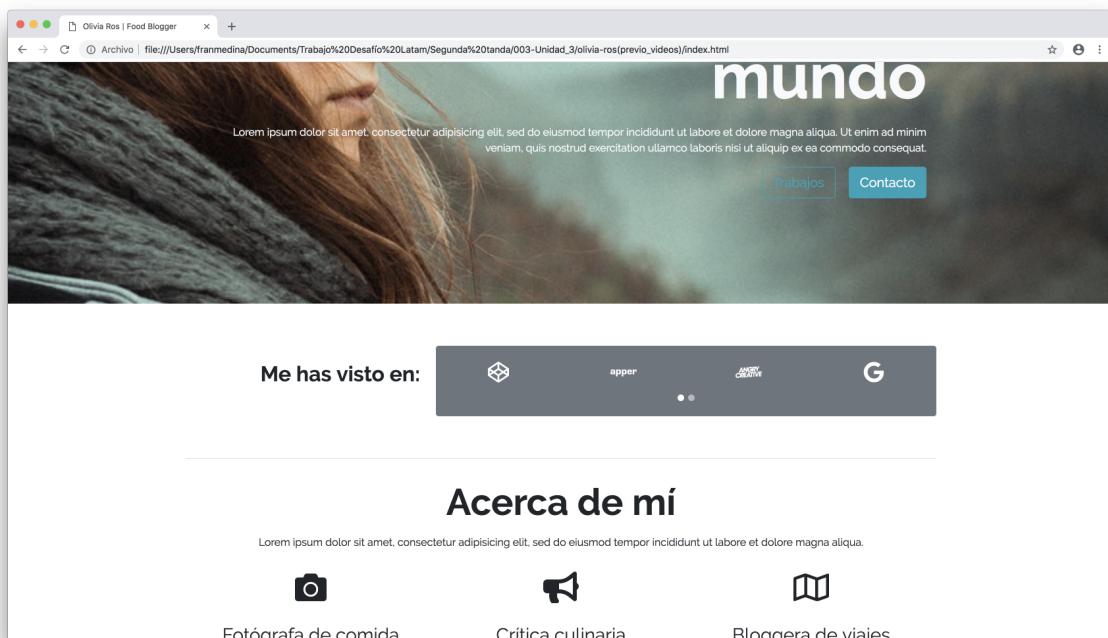


Imagen 6. Vista final de Proyecto - Carrusel de íconos.

A esos íconos le implementaremos popovers.

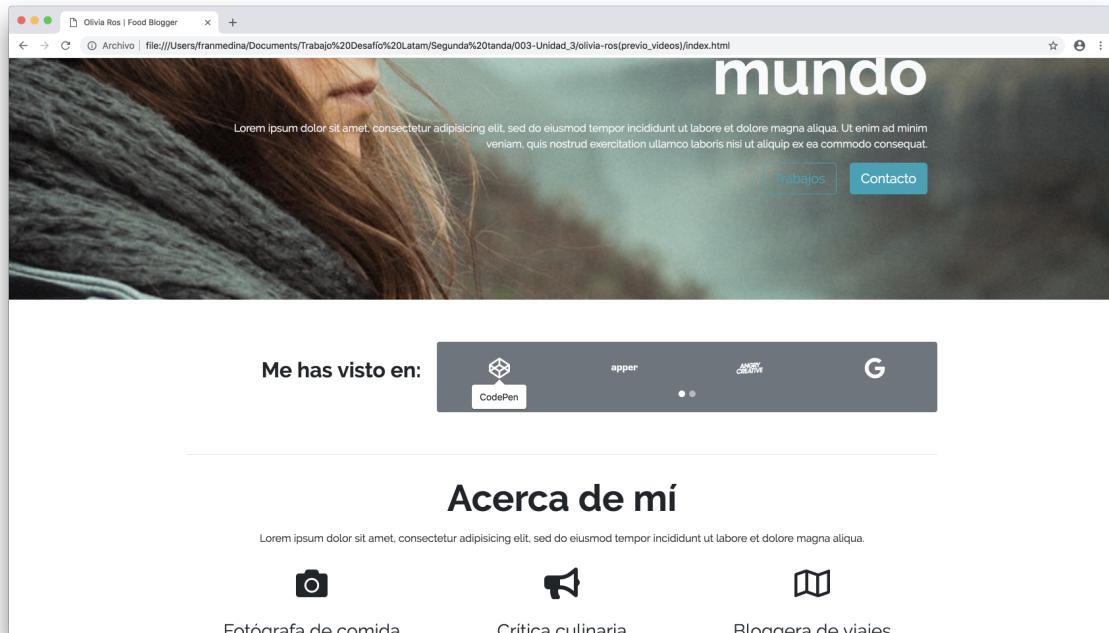


Imagen 7. Vista final de Proyecto - Íconos con popovers.

Y, finalmente, construiremos un formulario de contacto dinámico con [Typeform](#)

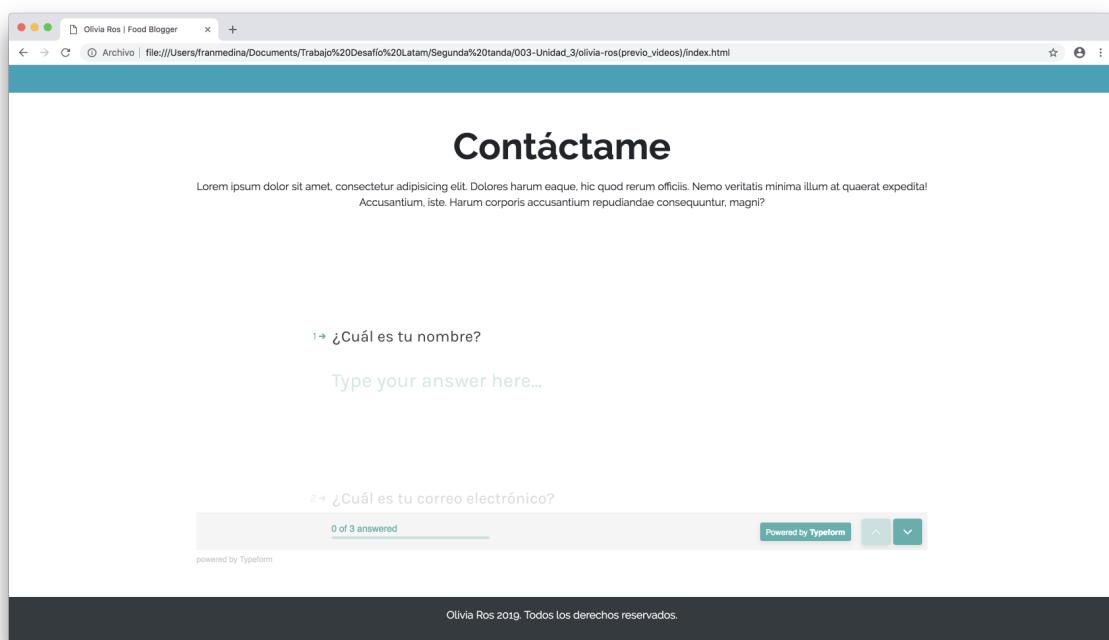


Imagen 8. Vista final de Proyecto - Formulario de contacto dinámico.

Avisos alert y console log

Nueva carpeta para probar

Antes de comenzar a realizar los scripts para mejorar nuestro proyecto Olivia Ros, vamos a crear otra página en una carpeta diferente, donde realizaremos las pruebas pertinentes para conocer el lenguaje de JavaScript, paso a paso.

Como ya lo hemos hecho durante el módulo crearemos un directorio de carpetas, con sus archivos ordenados y clasificados.

1. Comencemos creando una nueva carpeta donde más nos acomode. Ésta se llamará `probando.js`.
2. El siguiente paso será entrar a Atom.
3. Crearemos el `index.html`, en la raíz de la carpeta `probando-js`.
4. Creamos la carpeta `assets` presionando "botón derecho" y luego nueva carpeta.
5. Dentro de assets crearemos la carpeta `js`.

Hacer este ejercicio por cada proyecto nos ayudará a ordenarnos.

Ahora al `index.html` le daremos la estructura base de un archivo HTML.

```
<!DOCTYPE html>
<html lang="es" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Probando JS</title>
  </head>
  <body>

  </body>
</html>
```

Agregando JS

Inspector de elementos

Comenzaremos integrando JavaScript de la forma más simple, desde nuestro navegador. Abriremos el `index.html` en el navegador Chrome e ingresaremos al inspector de elementos.

Ingresaremos a la pestaña `console`, desde acá podremos ejecutar código JavaScript. Realicemos la siguiente prueba. Vamos a escribir en la consola lo siguiente. `alert("Esta prueba es desde la consola");` presionaremos `enter` para ejecutarlo, podremos observar que nos aparece la alerta.

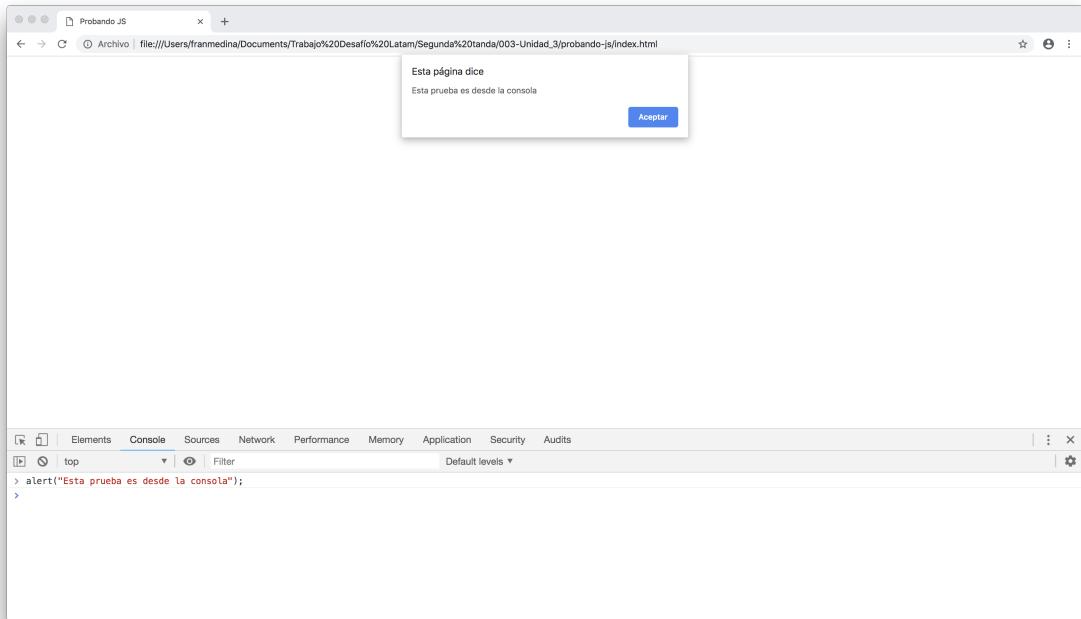


Imagen 9. Mensaje de alerta en navegador.

Como veremos, esto arrojará un mensaje de alerta en nuestro navegador con el mensaje descrito. Durante esta prueba estamos usando la instrucción `alert()`. Felicidades, ejecutaste tu primer script en una página.

Aunque recuerda que como utilizamos el inspector de elementos esta instrucción no se guardará dentro de nuestro archivo.

Desde el mismo archivo HTML

Ahora que ya sabemos como probar JavaScript desde el inspector de elementos, vamos a integrarlo directamente a un archivo HTML.

La etiqueta correspondiente para escribir JavaScript dentro de HTML es la etiqueta `<script> ... </script>`. Ésta podemos colocarla tanto dentro de la etiqueta `<head>` como de la etiqueta `<body>`. Como vimos en la unidad de Bootstrap, el documento se carga desde arriba hacia abajo, por lo tanto, nos será conveniente poner esta etiqueta al final del `<body>` para que la página en total se pueda cargar más rápido.

Entonces, hagámoslo. Escribamos nuestro script al final de la etiqueta `<body>` del `index.html` con la instrucción `alert();`.

```
<!DOCTYPE html>
<html lang="es" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Probando JS</title>
  </head>
  <body>

    <script>
      alert("Esta prueba es desde la etiqueta script");
    </script>
  </body>
</html>
```

Recarguemos el navegador y veamos el resultado.

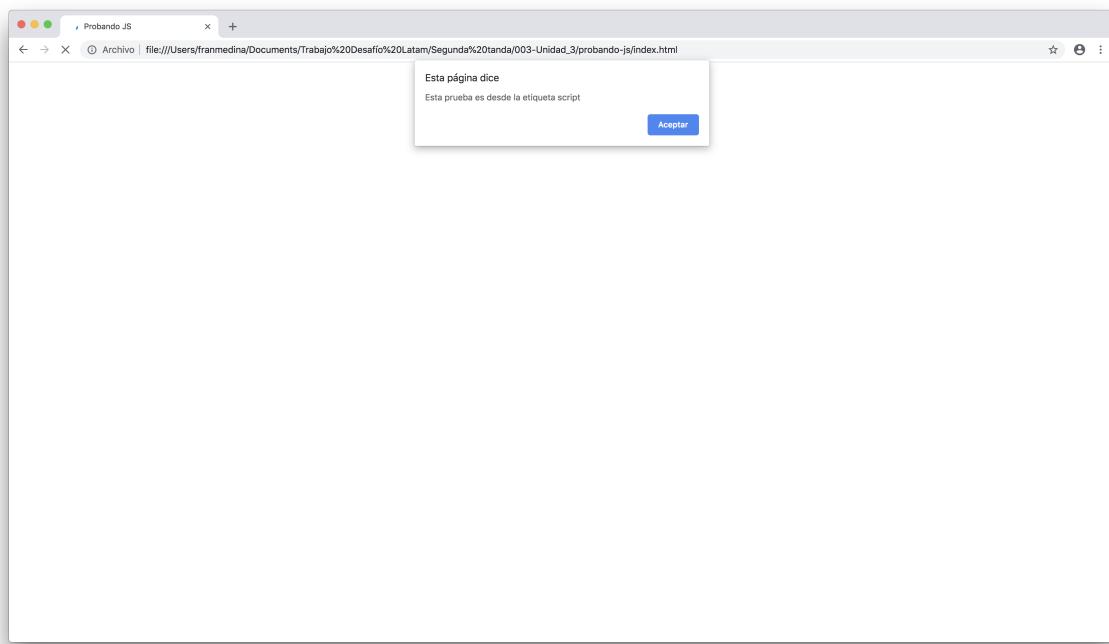


Imagen 10. Mensaje de alerta en archivo HTML.

Desde un archivo JS

También podemos escribir JavaScript en un archivo externo al HTML. Para vincularlo a la etiqueta `<script>` le debemos añadir el atributo `src` con valor de la ruta del archivo. Éste puede ser un archivo que tengamos dentro de la carpeta de nuestro proyecto, como también un archivo llamado desde CDN.

Esta vez escribiremos la misma línea de código que escribimos dentro de la etiqueta `<script>` en el HTML, pero en un archivo nuevo de JavaScript. Para ello, dentro de la carpeta `assets/js` crearemos un archivo con extensión `.js`. Lo podremos llamar como queramos, llamémoslo `script.js`.

Dentro de `script.js` escribiremos lo siguiente:

```
alert("Esta prueba es desde el archivo script.js");
```

Como habrás notado, **no** estamos usando la etiqueta `<script> ... </script>` debido a que estamos escribiendo al interior de un archivo de JavaScript donde no son requeridas.

Ahora en el HTML vamos a borrar la alerta desde la etiqueta y dejaremos su contenido vacío. Lo que sí haremos será escribir el atributo `src` con el valor de la ruta del archivo:

```

<!DOCTYPE html>
<html lang="es" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Probando JS</title>
  </head>
  <body>

    <script src="assets/js/script.js"></script>
  </body>
</html>

```

Recarguemos el navegador y veamos el resultado.

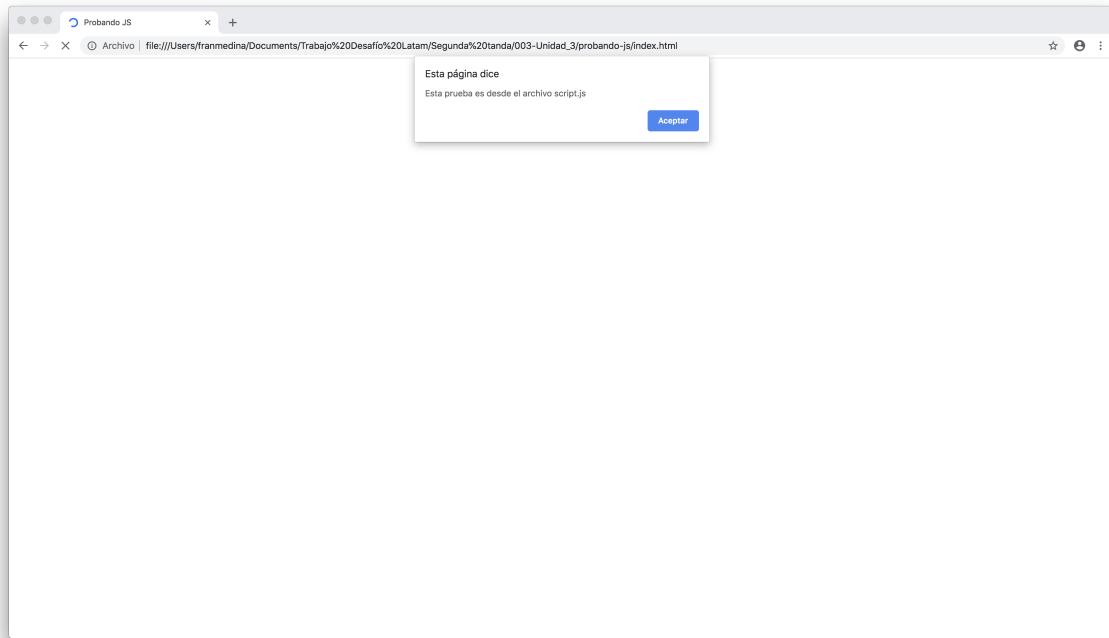


Imagen 11. Mensaje de alerta vinculado a etiqueta script.

Muy bien. Ya probamos las distintas formas de incorporar JavaScript a nuestra página.

La forma que más utilizaremos es esta última, ya que hacerlo en un archivo externo al HTML nos permitirá reutilizarlo en otros archivos HTML del sitio sin tener que escribirlo una y otra vez en la etiqueta `<script>`.

Bueno, la verdad nosotros probamos nuestras primeras líneas de código con una instrucción llamada `alert()`, que es una función de JavaScript que muestra un cuadro de alerta con un mensaje específico y un botón Aceptar. En términos de usabilidad no nos será muy conveniente utilizarlo, ya que puede ser molesto para el usuario y abandonar nuestro sitio. Pero como ejercicio práctico es un buen comienzo.

Console log

Existe otra función que aprenderemos, el llamado `console.log();`. Con él se puede imprimir lo que necesitemos, pero en la consola. Probemos.

```

alert("Esta prueba es desde el archivo script.js");
console.log("Estamos probando mensajes en la consola");

```

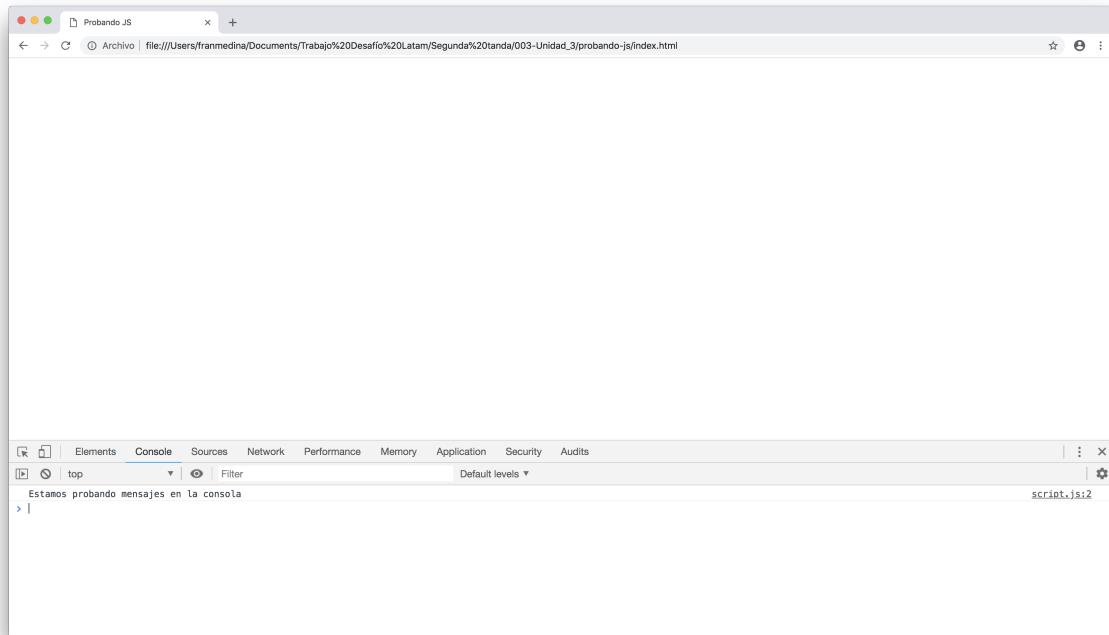


Imagen 12. Utilizando la función `console.log()`.

La función `console.log();` nos servirá más adelante para hacer debug. Por mientras nos ayudará a mostrar diversos resultados.

Resumen

Lo que hicimos en este video fue conocer las distintas formas de implementar JavaScript.

- A través del inspector de elementos.
- Desde el mismo HTML.
- Desde un archivo externo al HTML.

Nos apoyamos en las funciones `alert();` y `console.log();` para realizar las pruebas.

Declaraciones y sintaxis

Introducción a las declaraciones

Antes de seguir escribiendo código JavaScript debemos entender lo que estamos haciendo.

- Un programa es una lista de "instrucciones" para ser "ejecutadas" por un computador.
- En un lenguaje de programación, estas instrucciones de programación se denominan declaraciones.
- Un script de JavaScript es una lista de declaraciones.

Es decir, en nuestro archivo `script.js` hicimos dos declaraciones que el navegador ejecutó.

1. Realizar una alerta al usuario con el texto "Esta prueba es desde el archivo script.js".
2. Imprimir un texto en consola con el texto "Estamos probando mensajes en la consola".

Las declaraciones de JavaScript se componen de:

- Valores.
- Operadores.
- Expresiones.
- Palabras clave.
- Comentarios.

Sintaxis JavaScript

La sintaxis de JavaScript tiene sus propias reglas, pero dependerá del tipo de declaración que corresponda.

Los **punto y coma** sirven para separar las declaraciones de JavaScript. Si bien ahora ya no son 100% obligatorios nos ayudarán a ordenar nuestro propio código y a evitar errores.

Es importante tener en cuenta que JavaScript:

- Es sensible a las mayúsculas y minúsculas.
- No toma en cuenta los espacios en blanco ni los saltos de línea.

Valores

Respecto a la declaración de tipo "Valor" existen dos tipos:

1. Literales.
2. Variables.

Los valores de tipo literales sólo se escriben:

`100`

`0.5`

`"Hola a todos"`

Cómo se escriban dependerá del tipo de dato, que veremos más en profundidad en el próximo video. Pero es necesario asimilar que los números se pueden escribir enteros o en decimales y que los textos (strings) están siempre entre comillas, éstas pueden ser simples o dobles.

Los valores de tipo variable se escriben así:

```
var numero = 100;  
var numero2 = 0.5;  
var frase = "Hola a todos";
```

Una variable se emplea para almacenar y hacer referencia a un valor. Se compone por la palabra clave `var`, seguida del nombre de la variable (sin tildes, caracteres especiales o espacios), con un símbolo `=` para la asignación y finalmente su valor (cómo se escriba también dependerá del tipo de dato).

Operadores

Los operadores son símbolos con los que se pueden realizar operaciones con variables, valores literales o variables y valores literales.

Veremos 3 tipos

1. Operadores de asignación.
2. Operadores aritméticos.
3. Operadores de comparación.

Éstos operadores los veremos en dos videos más, en mayor profundidad.

Expresiones

Una expresión es una combinación de valores, variables y operadores, que computa un valor.

El cómputo se llama evaluación.

Una expresión puede contener los distintos tipos de valores (literales o variables) y éstos pueden ser de distintos tipos de datos.

Palabras Clave

Las palabras clave son tokens que tienen un significado especial en JavaScript.

Las declaraciones de JavaScript a menudo comienzan con una palabra clave para identificar la acción de JavaScript que se realizará.

Muchas de las palabras clave están reservadas, lo que significa que no pueden ser utilizadas como variables, etiquetas o nombres de funciones.

Al final de este video te dejaremos una lectura con una lista de palabras clave.

Comentarios

No todas las declaraciones de JavaScript son "ejecutadas".

El código después de `//` o entre `/ *` y `* /` se trata como un comentario.

Los comentarios son ignorados, y no serán ejecutados.

Los comentarios son muy útiles para documentar el código.

Tipos de datos y variables

Qué es una variable

Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. Gracias a las variables es posible crear "programas genéricos", es decir, programas que funcionan siempre igual independientemente de los valores concretos utilizados.

Sintaxis de una variable

Se compone por la palabra clave `var`, seguida del nombre o identificador de la variable, con un símbolo `=` para la asignación y finalmente su valor (cómo se escriba también dependerá del tipo de dato).

El identificador debe cumplir las siguientes normas:

- Sólo puede estar formado por letras, números y los símbolos `$` (dólar) y `_` (guión bajo).
- El primer carácter **no** puede ser un número.

Aunque todas las variables de JavaScript se crean de la misma forma, la forma en la que se les asigna un valor depende del tipo de dato que se quiere almacenar.

Aunque existen más tipos de datos en este video veremos:

1. Números.
2. Texto.
3. Booleanos.

Tipo de dato numérico (integer y float)

Se utilizan para almacenar valores numéricos enteros (llamados `integer` en inglés) o decimales (llamados `float` en inglés). En este caso, el valor se asigna indicando directamente el número entero o decimal. Los números decimales utilizan el carácter `.` (punto) para separar la parte entera y la parte decimal.

```
var entero = 10;  
var decimal = 2.5;
```

Tipo de dato cadena de texto (string)

Se utilizan para almacenar caracteres, palabras y/o frases de texto. Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples, para delimitar su comienzo y su final:

```
var frase = "Hola ¿Cómo estás?";  
var letra = "d";
```

Si el texto requiere comillas dentro de él se deberá usar la comilla contraria a la que se está usando para englobar el texto, por ejemplo:

```
var comillas = "Hola, esto es un texto y 'este está entre comillas'"
```

Para caracteres que son difíciles de incluir en esta cadena de texto existe el backslash.

Si se quiere incluir...	Se debe incluir...
Un salto de línea	\n
Un tabulador	\t
Una comilla simple	\ '
Una comilla doble	\ "
Un backslash	\ \

Hagamos una prueba en el inspector de elementos:

```
var texto = "Estoy escribiendo un texto con \nsalto de línea"  
alert(texto);
```

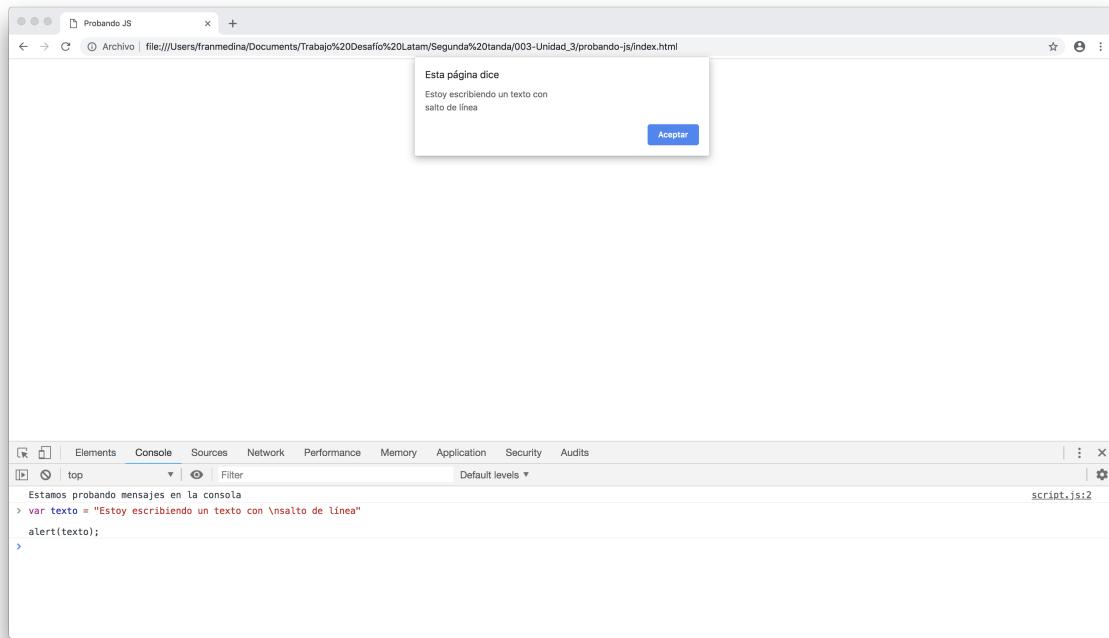


Imagen 13. Utilizando salto de línea en mensaje.

Tipo de dato verdadero o falso (boolean)

Una variable de tipo booleano almacena un tipo especial de valor que solamente puede tomar dos valores: `true` (verdadero) o `false` (falso). No se puede utilizar para almacenar números y tampoco permite guardar cadenas de texto.

Profundizaremos mucho más en este tipo de dato en el siguiente video, de operadores y asignación.

Operadores y asignación

Como ya lo vimos, los operadores permiten manipular el valor de las variables, realizar operaciones matemáticas con sus valores y comparar diferentes variables. De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

Hace dos videos vimos que veremos 3 tipos en profundidad:

1. Operadores de asignación.
2. Operadores aritméticos
3. Operadores de comparación.

Operadores de asignación

Los operadores de asignación son muchos, existen de:

- Asignación básica.
- Asignación de adición.
- Asignación de sustracción.
- Asignación de multiplicación.
- Asignación de división.
- Asignación de resto.
- Entre otros...

Pero nosotros nos vamos a concentrar de momento en la asignación básica.

El operador de asignación básico es el igual (`=`), el cual asigna el valor del operando derecho al operando izquierdo.

Es decir, `x = y` asigna el valor de `y` a `x`.

Probémoslo en el inspector de elementos:

```
var valor1 = 2;  
var valor2 = 5;  
  
valor1 = valor2;  
  
alert(valor1);
```

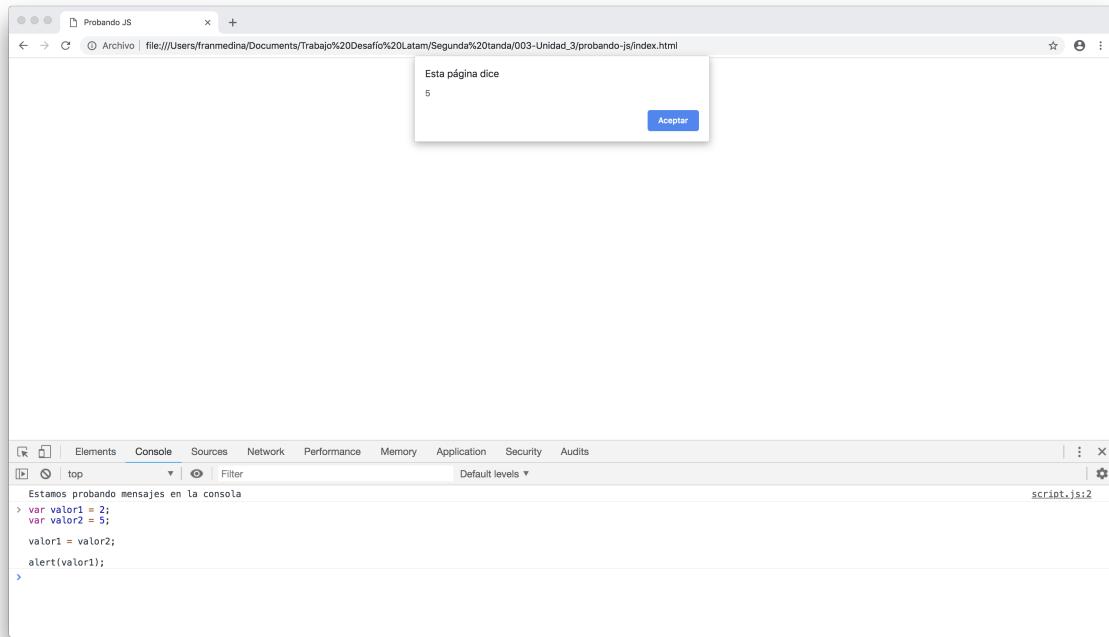


Imagen 14. Probando operador de asignación básico.

Muy bien.

Operadores aritméticos

Los operadores aritméticos son usados para realizar operaciones matemáticas.

Éstos son los siguientes:

Operador	Operación	Sintaxis
+	Adición	<code>var sumando1 = 5; var sumando2 = 4; var total_suma = sumando1 + sumando2;</code>
-	Sustracción	<code>var minuendo = 6; var sustraendo = 3; var total_resta = minuendo - sustraendo;</code>
*	Multiplicación	<code>var factor1 = 2; var factor2 = 6; var total_multi = factor1 * factor2;</code>
/	División	<code>var dividendo = 9; var divisor = 3; var total_divi = dividendo / divisor;</code>
%	Módulo de división	<code>var dividendo = 9; var divisor = 3; var total_modulo = dividendo % divisor;</code>

Probemos el siguiente código en el inspector de elementos:

```

var sumando1 = 5;
var sumando2 = 4;

var total_suma = sumando1 + sumando2;

alert(total_suma);

```

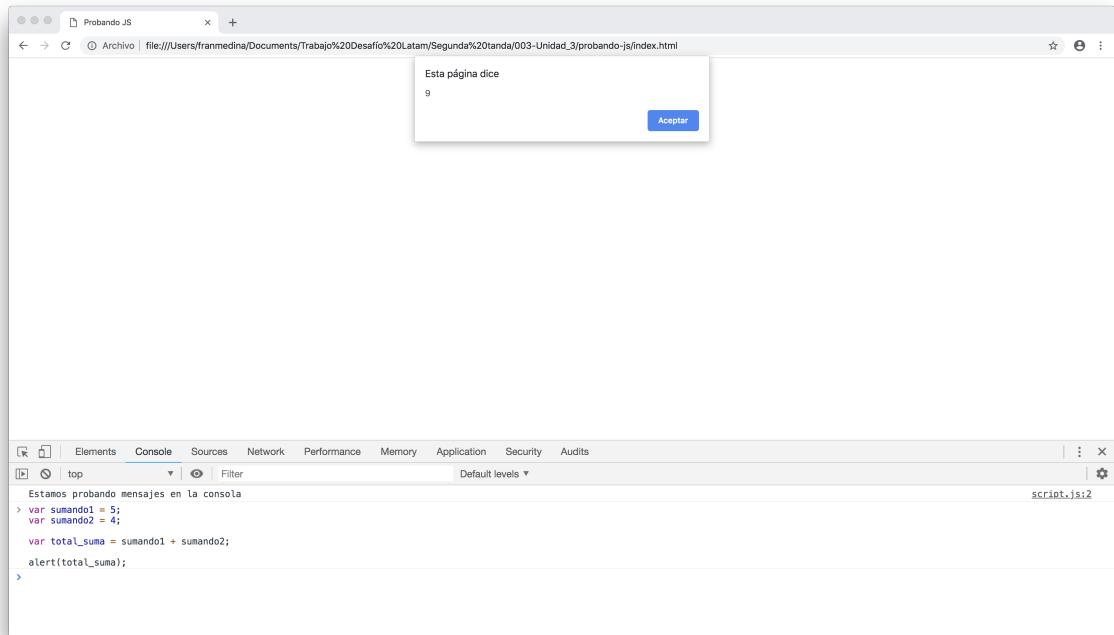


Imagen 15. Probando operador aritméticos.

Para los que no lo conocen el operador "módulo" calcula el resto de la división entera de dos números. Cuando la división es exacta el módulo es `0`, pero cuando la división no es exacta arroja el número restante.

Por ejemplo:

```
var dividendo = 5;
var divisor = 4;

var total_modulo = dividendo % divisor;

alert(total_modulo);
```

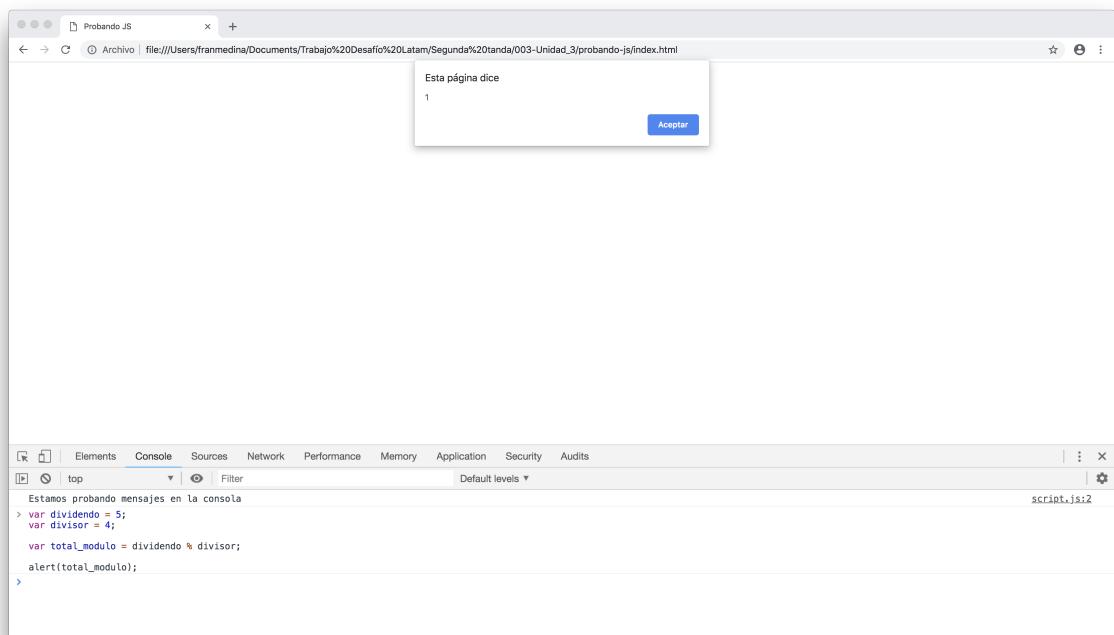


Imagen 16. Probando operador módulo, ejemplo 1.

```

var dividendo = 9;
var divisor = 7;

var total_modulo = dividendo % divisor;

alert(total_modulo);

```

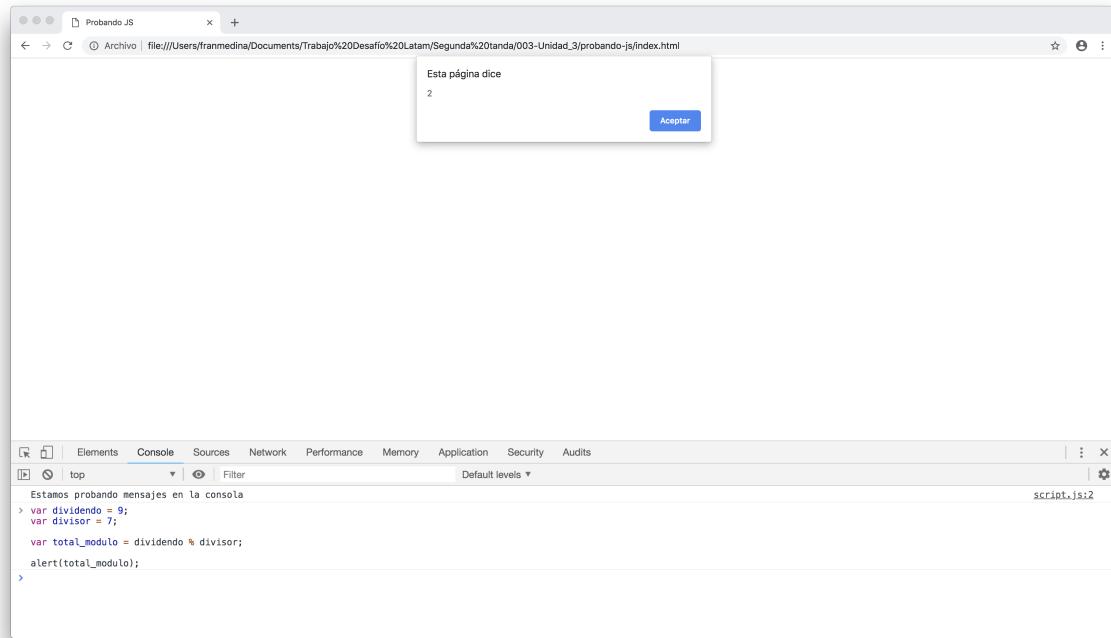


Imagen 17. Probando operador módulo, ejemplo 2.

Operadores de comparación

Los operadores de comparación son usados para comparar entre valores.

Éstos son los siguientes:

Operador	Significado
>	Mayor que
<	Menor que
==	Igual que
!=	Distinto de
>=	Mayor o igual que
<=	Menor o igual que
====	Idéntico que
!==	No idéntico que

La diferencia entre el igual y el idéntico es la siguiente:

Cuando se compara con el operador igual que es verdadero si variable1 y variable2 son iguales, en cambio cuando se compara con el operador idéntico que es verdadero si variable1 y variable2 son idénticas, es decir, sus valores coinciden y son del mismo tipo de dato.

Hagamos un ejemplo en el inspector de elementos:

```
var variable1 = 5;  
var variable2 = "5";  
  
alert(variable1 == variable2);  
alert(variable1 === variable2);
```

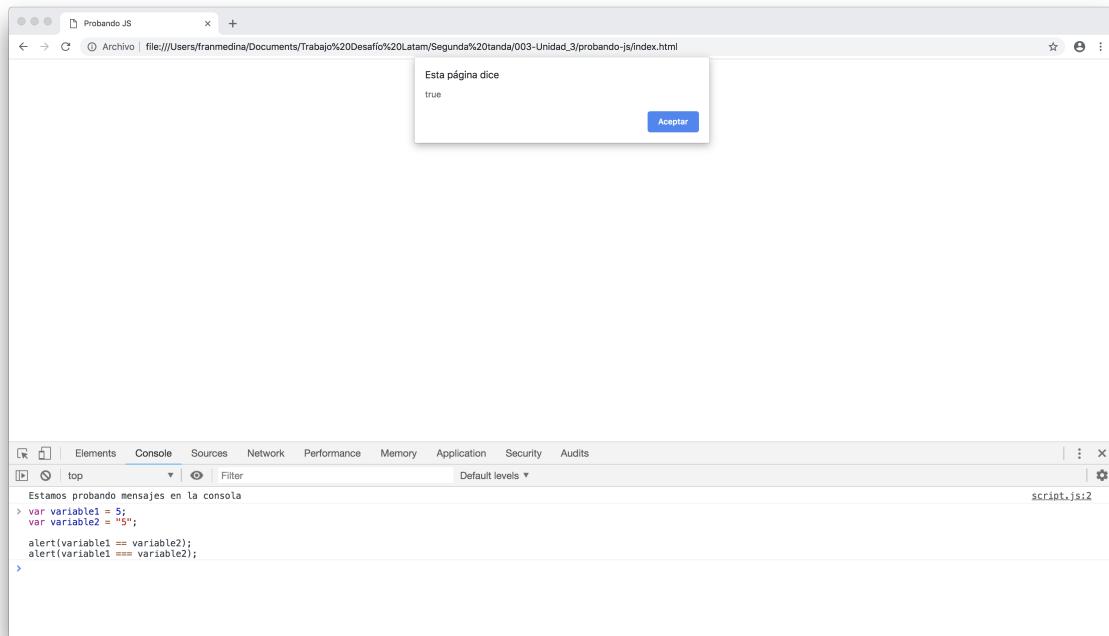


Imagen 18. Ejemplo de inspector de elemetos 1.

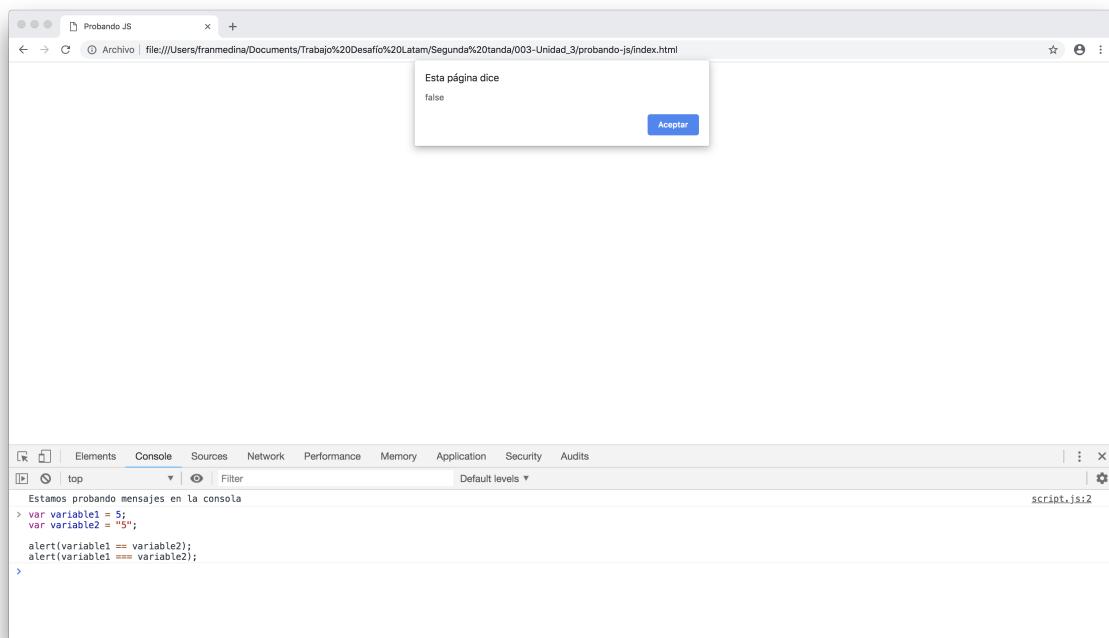


Imagen 19. Ejemplo de inspector de elemetos 2.

Ejercitando con prompt

Ahora que hemos aprendido un poco de los conceptos básicos de JavaScript llegó la hora de ejercitarse un poco.

Para ello conoceremos una función llamada `prompt()`.

`prompt()` función sirve para que el usuario ingrese un dato. Tiene 2 parámetros: el primero es el mensaje que se muestra en la ventana y el segundo es el valor inicial del área de texto.

Lo que escriba el usuario puede ser almacenado en una variable.

Hagamos un ejemplo:

```
var nombre = prompt("Hola, coloque su nombre por favor", "Nombre por defecto");

alert("Hola, nos alegra que estés por aquí " + nombre);
```

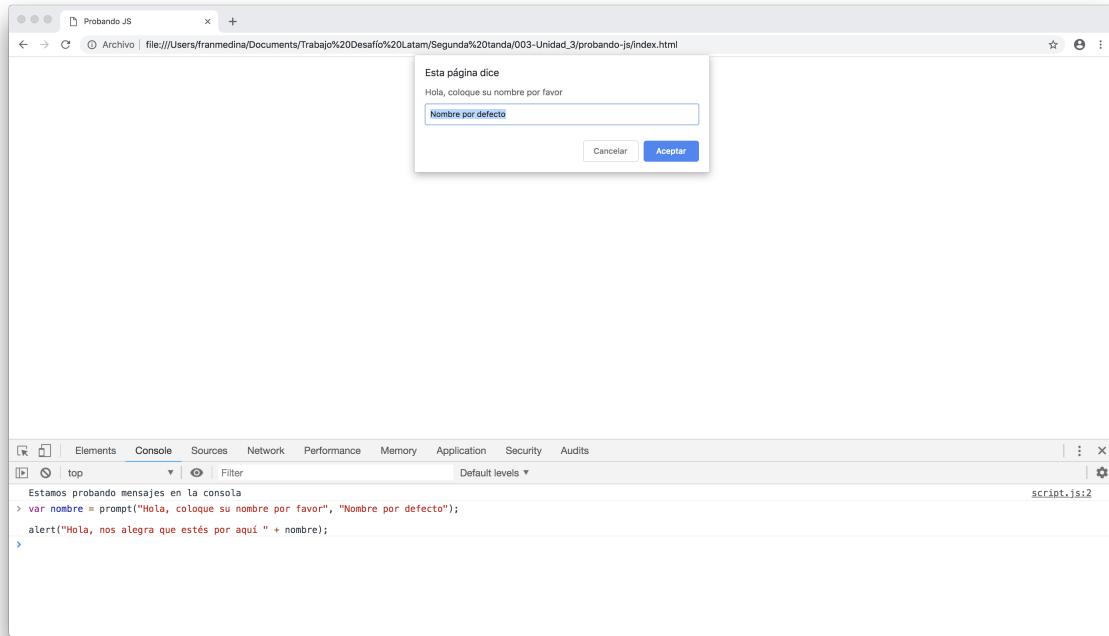


Imagen 20. Ejemplo con `prompt()`, parte 1.

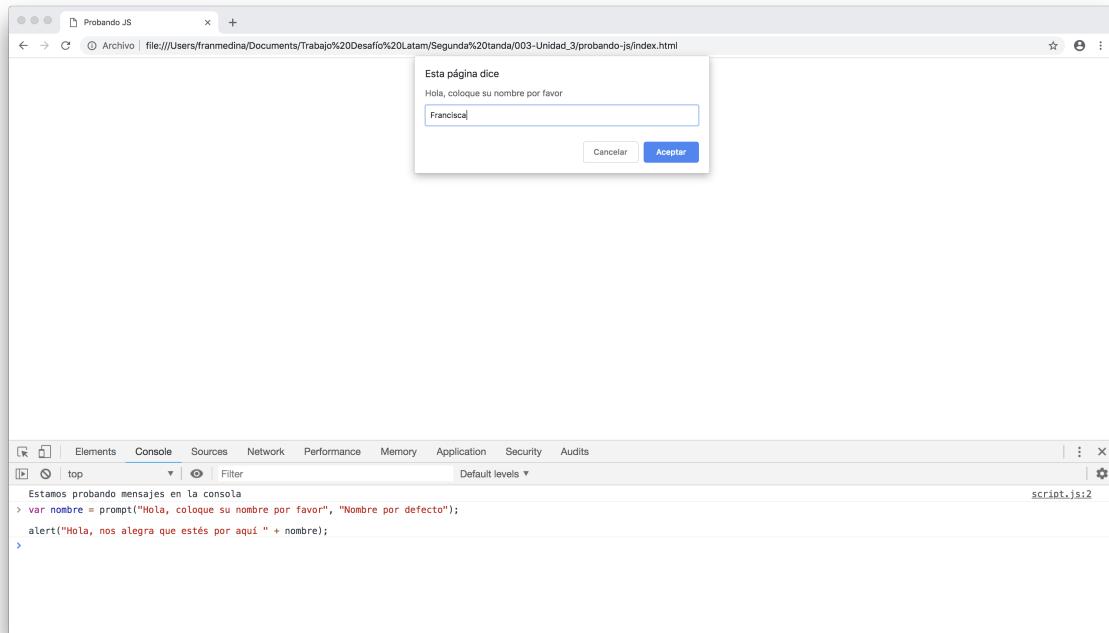


Imagen 21. Ejemplo con prompt(), parte 2.

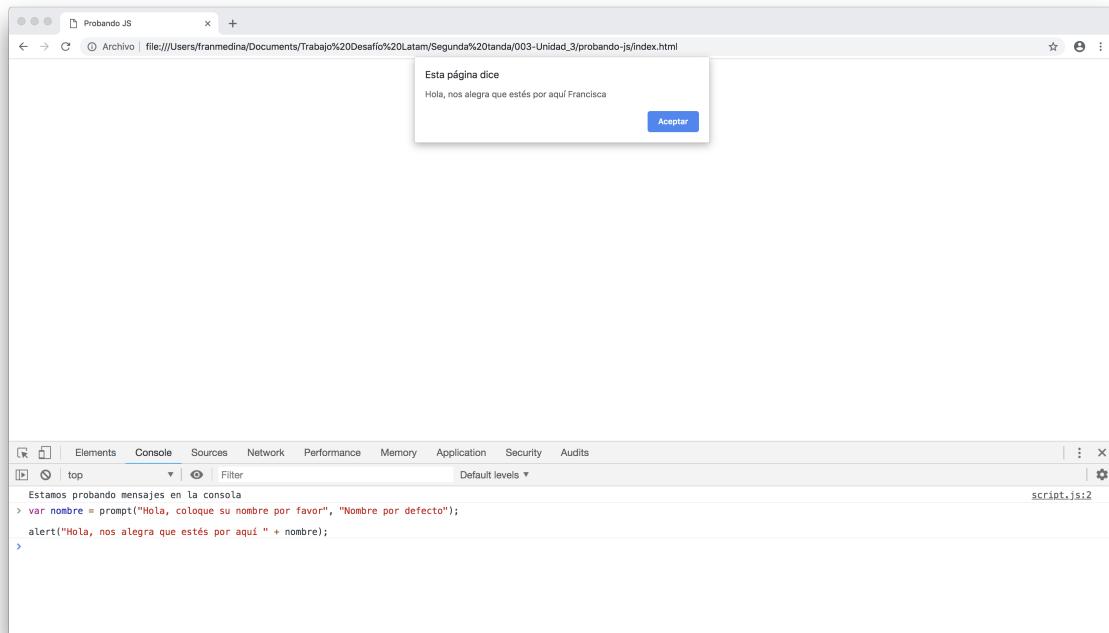


Imagen 22. Ejemplo con prompt(), parte 3.

Es importante entender lo que acabamos de hacer.

Primero a través de la función `prompt();` le pedimos el nombre al usuario. Luego, a través de una alerta lo imprimimos junto a una cadena de texto. Lo que hicimos fue concatenar un string junto con el valor de una variable.

Ahora conoceremos la función `document.write();`

Lo que hace esta función es escribir en la página el texto que se ingresa como parámetro.

Hagamos la prueba cambiando la función `alert();` por un `document.write();`

```

var nombre = prompt("Hola, coloque su nombre por favor", "Nombre por defecto");
document.write("Hola, nos alegra que estés por aquí " + nombre);

```

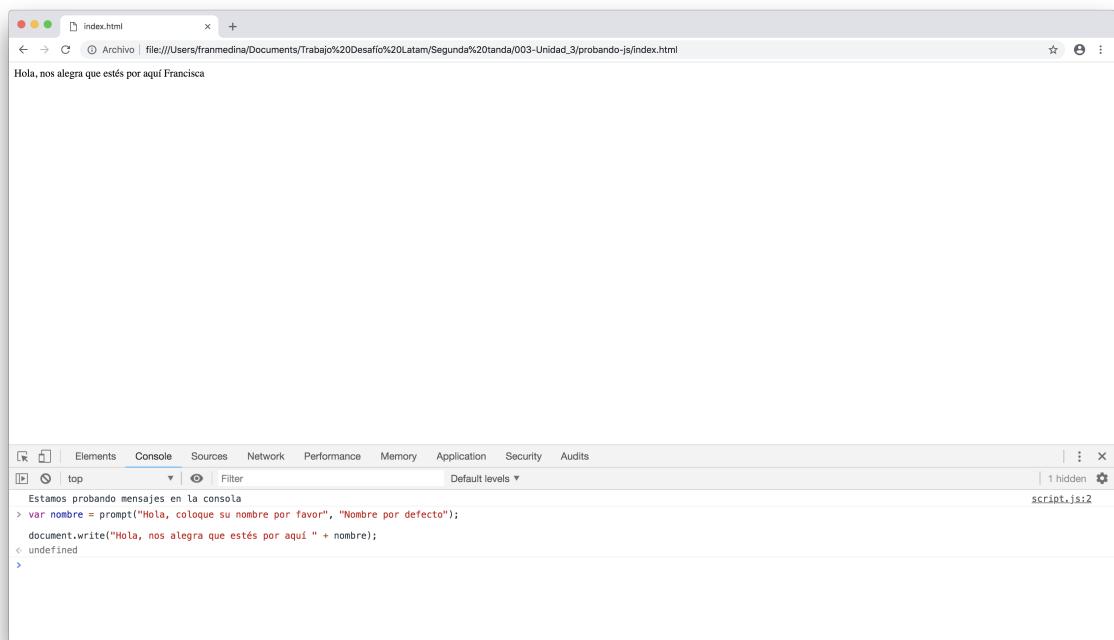


Imagen 23. Ejemplo con document.white().

Ahora, sumemos dos números que ingrese el usuario.

```

var sumando1 = prompt("Hola, coloque un número como primer sumando", "10");
var sumando2 = prompt("Hola, coloque un número como segundo sumando", "2");

var total_suma = sumando1 + sumando2;
document.write("La suma entre los dos números es " + total_suma);

```

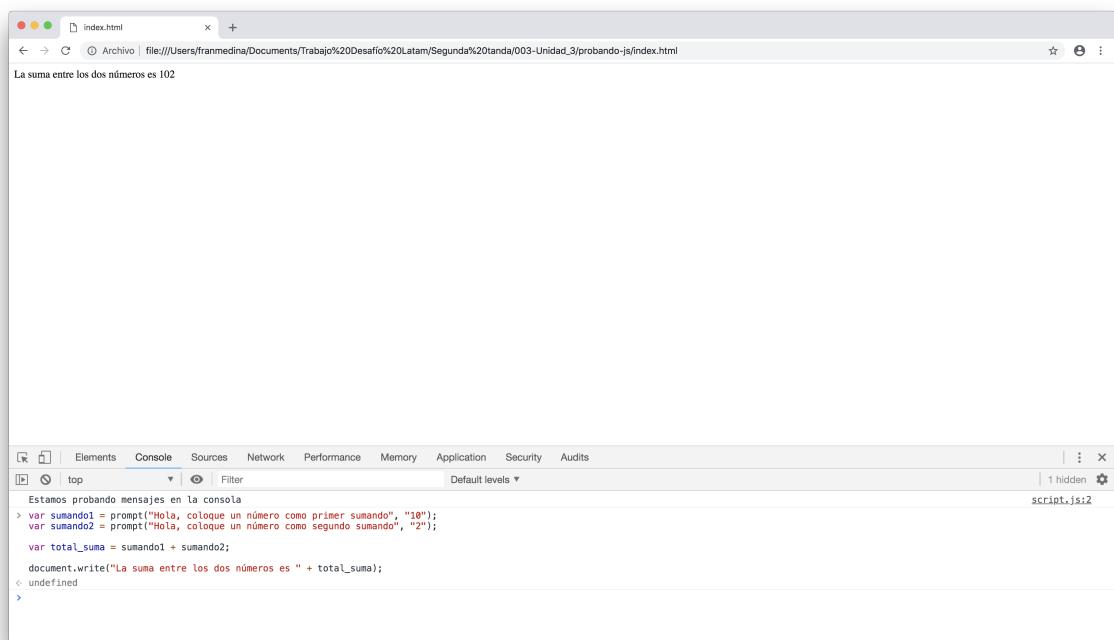


Imagen 24. Ejemplo con document.white(), suma.

¿Qué pasó?

Bueno, lo que sucede es que lo que ingresa el usuario se interpreta como string (cadena de texto). Y al verse enfrentado los dos valores de tipo string con un signo `+` se concatenan, o sea, se unen. Para solucionar este problema y nos de la suma debemos transformar ese string a número.

```
var sumando1 = prompt("Hola, coloque un número como primer sumando", "10");
var sumando2 = prompt("Hola, coloque un número como segundo sumando", "2");

sumando1 = parseInt(sumando1);
sumando2 = parseInt(sumando2);

var total_suma = sumando1 + sumando2;

document.write("La suma entre los dos números es " + total_suma);
```

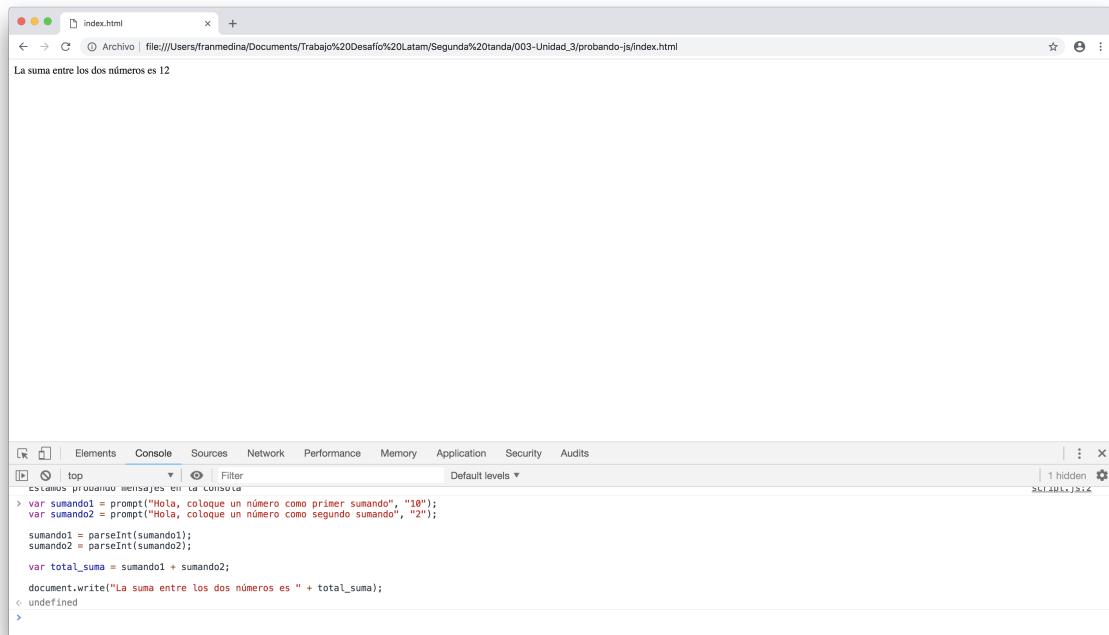


Imagen 25. Ejemplo con `document.white()`, de string a número.

Lo que hicimos fue asignar a la variable su mismo valor pero transformado a `integer` o número entero.

Una vez transformado a número entero se puede realizar la suma entre números.

Ahora, haremos lo siguiente:

```
document.write("<h1 id='color_letra'>Estoy aprendiendo JavaScript</h1>");

var dividendo = prompt("Hola, coloque un número dividendo", "10");
var divisor = prompt("Hola, coloque un número divisor", "2");
var color = prompt("Hola, escoja también un color de los colores por nombre de CSS \n Ejemplo: blue, yellow, green, red, etc...", "blue");

dividendo = parseInt(dividendo);
divisor = parseInt(divisor);

total_divi = dividendo / divisor;
total_modulo = dividendo % divisor;
```

```

document.write("La division entre los dos números es " + total_divi + " y su
módulo es " + total_modulo + ".");
document.write("Además escogiste el color " + color + " lo cuál cambia de
color de letra al contenido de la etiqueta H1");

document.getElementById("color_letra").style.color = color;

```

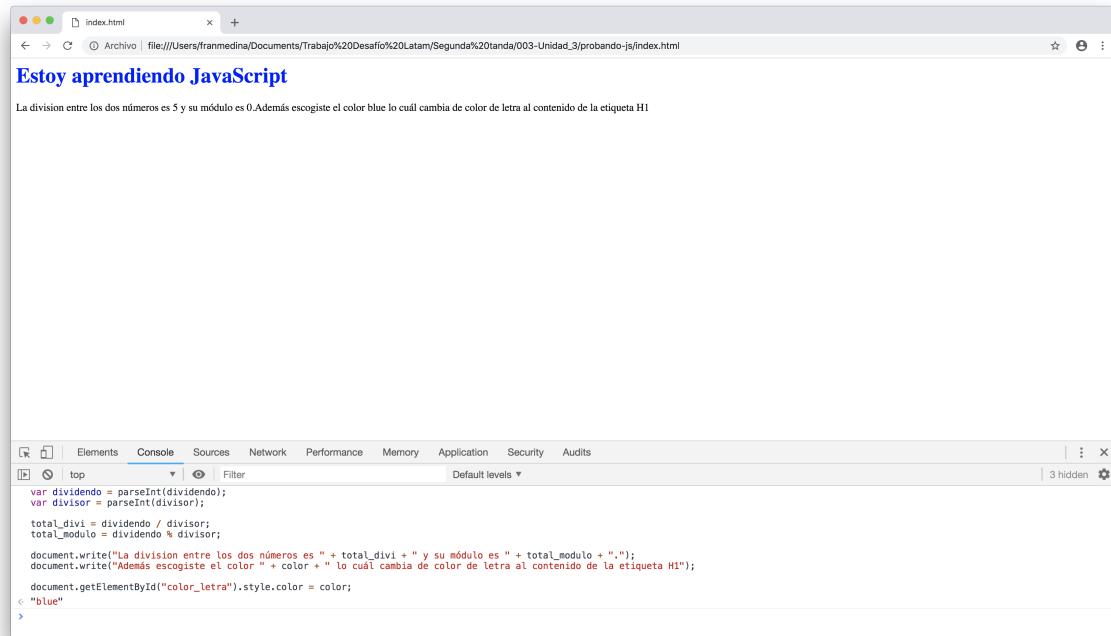


Imagen 26. Incluyendo Integer.

Con la función `document.getElementById();` encontramos el H1 correspondiente y le cambiamos el color según la variable.