Fundamentos de JavaScript

JavaScript es el lenguaje de programación que debes usar para añadir características interactivas a tu sitio web, (por ejemplo, juegos, eventos que ocurren cuando los botones son presionados o los datos son introducidos en los formularios, efectos de estilo dinámicos, animación, y mucho más). Este artículo te ayudará a comenzar con este lenguaje extraordinario y te dará una idea de qué es posible hacer con él.

¿Qué es JavaScript realmente?

JavaScript es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Fue inventado por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y la Corporación Mozilla.

Puedes hacer casi cualquier cosa con JavaScript. Puedes empezar con pequeñas cosas como carruseles, galerías de imágenes, diseños fluctuantes, y respuestas a las pulsaciones de botones. Con más experiencia, serás capaz de crear juegos, animaciones 2D y gráficos 3D, aplicaciones integradas basadas en bases de datos ¡y mucho más!

JavaScript por sí solo es bastante compacto aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo. Esto incluye:

- Interfaces de Programación de Aplicaciones del Navegador (APIs) — APIs construidas dentro de los navegadores que ofrecen funcionalidades como crear dinámicamente contenido HTML y establecer estilos CSS, hasta capturar y manipular un vídeo desde la cámara web del usuario, o generar gráficos 3D y muestras de sonido.
- APIs de terceros, que permiten a los desarrolladores incorporar funcionalidades en sus sitios de otros proveedores de contenidos como Twitter o Facebook.
- Marcos de trabajo y librerías de terceros que puedes aplicar a tu HTML para que puedas construir y publicar rápidamente sitios y aplicaciones.

Ya que se supone que este artículo es solo una introducción ligera a JavaScript, la intención no es confundirte en esta etapa hablando en detalle sobre cuál es la diferencia entre el núcleo del lenguaje JavaScript y las diferentes herramientas listadas arriba. Puedes aprender todo eso en detalle más tarde, en el Área de Aprendizaje en MDN, y en el resto de MDN.

Debajo se presentan algunos aspectos del núcleo del lenguaje y también jugarás con unas pocas características de la API del navegador. ¡Diviértete!

Ejemplo «*¡Hola Mundo!*»

La sección de arriba suena realmente emocionante, y debería serlo. JavaScript es una de las tecnologías web más emocionantes, y cuando comiences a ser bueno en su uso, tus sitios web entrarán en una nueva dimensión de energía y creatividad.

Sin embargo, sentirse cómodo con JavaScript es un poco más difícil que sentirse cómodo con HTML y CSS. Deberás comenzar poco a poco y continuar trabajando en pasos pequeños y consistentes. Para comenzar,

mostraremos cómo añadir JavaScript básico a tu página, creando un «*¡Hola Mundo!*» de ejemplo (el estándar en los ejemplos básicos de programación).

> Advertencia: si no has venido siguiendo el resto de nuestro curso, descarga este código de ejemplo y úsalo como punto de partida.

1. Primero, ve a tu sitio de pruebas y crea una carpeta llamada `scripts`. Luego, dentro de la nueva carpeta de scripts, crea un nuevo archivo llamado `main.js` y guárdalo.
2. A continuación, abre tu archivo `index.html` e introduce el siguiente código en una nueva línea, justo antes de la etiqueta de cierre `</body>`:

```
<script src="scripts/main.js"></script>
```

3. Esto hace básicamente el mismo trabajo que el elemento `<link>` para CSS: aplica el código JavaScript a la página, para que pueda actuar sobre el HTML (y CSS, o cualquier cosa en la página).
4. Ahora añade el siguiente código al archivo `main.js`:

```
const miTitulo = document.querySelector('h1');
miTitulo.textContent = '¡Hola mundo!';
```

5. Finalmente, asegúrate de que has guardado los archivos HTML y JavaScript, y abre `index.html` en el navegador. Deberías ver algo así:

> Nota: la razón por la que has puesto el elemento `<script>` (en-US) casi al final del documento HTML es porque el navegador carga el HTML en el orden en que aparece en el archivo.
>
> Si se cargara primero JavaScript y se supone que debe afectar al HTML que tiene debajo, podría no funcionar, ya que ha sido cargado antes que el HTML sobre el que se supone debe trabajar. Por lo tanto, colocar el JavaScript cerca del final de la página es normalmente la mejor estrategia. Para aprender más sobre enfoques alternativos, mira Estrategias de carga de scripts.

¿Qué ha ocurrido?

El texto del título ha sido cambiado por *¡Hola mundo!* usando JavaScript. Hiciste esto primero usando la función `querySelector()` para obtener una referencia al título y almacenarla en una variable llamada `miTitulo`. Esto es muy similar a lo que hiciste con CSS usando selectores —quieres hacer algo con un elemento, así que tienes que seleccionarlo primero—.

Después de eso, estableciste el valor de la propiedad `textContent` de la variable `miTitulo` (que representa el contenido del título) como *¡Hola mundo!*

Nota: Las dos características que has utilizado en este ejercicio forman parte de la API del Modelo de Objeto de Documento (DOM), que tiene la capacidad de manipular documentos.

Curso intensivo de fundamentos del lenguaje

Ahora se explicarán algunas de las funciones básicas del lenguaje JavaScript para que puedas comprender mejor cómo funciona todo. Mejor aún, estas características son comunes para todos los lenguajes de programación. Si puedes entender esos fundamentos, deberías ser capaz de comenzar a programar en casi cualquier cosa.

Advertencia: en este artículo, trata de introducir las líneas de código de ejemplo en la consola de tu navegador para ver lo que sucede. Para más detalles sobre consolas JavaScript, mira Descubre las herramientas de desarrollo de los navegadores.

Variables

Las Variables son contenedores en los que puedes almacenar valores. Primero debes declarar la variable con la palabra clave `var` (menos recomendado) o `let`, seguida del nombre que le quieras dar. Se recomienda más el uso de `let` que de `var` (más adelante se profundiza un poco sobre esto):

```
let nombreDeLaVariable;
```

Nota: todas las líneas en JS deben acabar en punto y coma (;) para indicar que es ahí donde termina la declaración. Si no los incluyes puedes obtener resultados inesperados. Sin embargo, algunas personas creen que es una buena práctica tener punto y coma al final de cada declaración. Hay otras reglas para cuando se debe y no se debe usar punto y coma. Para más detalles, vea Guía del punto y coma en JavaScript (en inglés).

Nota: puedes llamar a una variable con casi cualquier nombre, pero hay algunas restricciones (ver este artículo sobre las reglas existentes). Si no estás seguro, puedes comprobar el nombre de la variable para ver si es válido.

Nota: JavaScript distingue entre mayúsculas y minúsculas. `miVariable` es una variable distinta a `mivariable`. Si estás teniendo problemas en tu código, revisa las mayúsculas y minúsculas.

Nota: para más detalles sobre la diferencia entre `var` y `let`, vea Diferencia entre var y let.

Tras declarar una variable, puedes asignarle un valor:

```
nombreDeLaVariable = 'Bob';
```

Puedes hacer las dos cosas en la misma línea si lo necesitas:

```
let nombreDeLaVariable = 'Bob';
```

Puedes obtener el valor de la variable llamándola por su nombre:

```
nombreDeLaVariable;
```

Después de haberle dado un valor a la variable, puedes volver a cambiarlo:

```
let nombreDeLaVariable = 'Bob';
nombreDeLaVariable = 'Steve';
```

Advierte que las variables tienen distintos tipos de datos:

Variable	Explicación	Ejemplo
String	Esto es una secuencia de texto conocida como cadena. Para indicar que la variable es una cadena, debes escribirlo entre comillas.	`let miVariable = 'Bob';`
Number	Esto es un número. Los números no tienen comillas.	`let miVariable = 10;`
Boolean	Tienen valor verdadero/falso. `true` / `false` son palabras especiales en JS, y no necesitan comillas.	`let miVariable = true;`
Array	Una estructura que te permite almacenar varios valores en una sola referencia.	`let miVariable = [1,'Bob','Steve',10];` Llama a cada miembro del array así: `miVariable[0]`, `miVariable[1]`, etc.
Object	Básicamente cualquier cosa. Todo en JavaScript es un objeto y puede ser almacenado en una variable. Mantén esto en mente mientras aprendes.	`let miVariable = document.querySelector('h1');` Todos los ejemplos anteriores también.

Entonces, ¿para qué necesitamos las variables? Las variables son necesarias para hacer cualquier cosa interesante en programación. Si los valores no pudieran cambiar, entonces no podrías hacer nada dinámico, como personalizar un mensaje de bienvenida de un usuario que visita tu página, cambiar la imagen que se muestra en una galería de imágenes, etc.

Comentarios

Puedes escribir comentarios entre el código JavaScript, igual que puedes en CSS. El navegador ignora el texto marcado como comentario. En JavaScript, los comentarios de una sola línea se escriben así:

```
// Esto es un comentario
```

Pero también puedes escribir comentarios en más de una línea, igual que en CSS:

```
/*
Esto es un comentario
de varias líneas.
*/
```

Operadores

Un operador es básicamente un símbolo matemático que puede actuar sobre dos valores (o variables) y producir un resultado. En la tabla de abajo aparecen los operadores más simples, con algunos ejemplos para probarlos en la consola del navegador.

Operador	Explicación	Símbolo(s)	Ejemplo
Suma/concatena	Se usa para sumar dos números, o juntar dos cadenas en una.	`+`	`6 + 9;` `"Hola " + "mundo!";`

Resta, multiplicación, división	Estos hacen lo que esperarías que hicieran en las matemáticas básicas.	`-`, `*`, `/`	`9 - 3;`
			`8 * 2; // La multiplicación en JS es un asterisco`
			`9 / 3;`

Operador de asignación	Los has visto anteriormente: asigna un valor a una variable.	`=`	`let miVariable = 'Bob';`
identidad/igualdad	Comprueba si dos valores son iguales entre sí, y devuelve un valor de `true` / `false` (booleano).	`===`	`let miVariable = 3;` `miVariable === 4;`

Operador	Explicación	Símbolo(s)	Ejemplo

			La expresión básica es `true`, pero la comparación devuelve `false` porque lo hemos negado:
			
			`let miVariable = 3;`
			`!miVariable === 3;`

Negación, distinto (no igual)	En ocasiones utilizado con el operador de identidad, la negación es en JS el equivalente al operador lógico NOT — cambia `true` por `false` y viceversa.	`!`, `!==`	Aquí estamos comprobando "`miVariable` NO es igual a 3". Esto devuelve `false`, porque `miVariable` ES igual a 3.
			
			`let miVariable = 3;`
			`miVariable !== 3;`

Hay muchos operadores por explorar, pero con esto será suficiente por ahora. Mira Expresiones y operadores para ver la lista completa.

> Nota: mezclar tipos de datos puede dar lugar a resultados extraños cuando se hacen cálculos, así que asegúrate de que relacionas tus variables correctamente y de que recibes los resultados que esperabas. Por ejemplo, teclea: `"3" + "25"` en tu consola. ¿Por qué no obtienes lo que esperabas? Porque las comillas convierten los números en "strings" (el término inglés para denominar cadenas de caracteres) y de este modo has acabado con los "strings" concatenados entre sí, y no con los números sumados. Si tecleas: `35 + 25`, obtendrás el resultado correcto.

Condicionales

Las condicionales son estructuras de código que permiten comprobar si una expresión devuelve *true* o no, y después ejecuta un código diferente dependiendo del resultado. La forma de condicional más común es la llamada `if... else`. Entonces, por ejemplo:

```
let helado = 'chocolate';
if (helado === 'chocolate') {
  alert('¡Sí, amo el helado de chocolate!');
} else {
  alert('Awwww, pero mi favorito es el de chocolate...');
}
```

La expresión dentro de `if (... )` es el criterio — este usa al operador de identidad (descrito arriba) para comparar la variable `helado` con la cadena `chocolate` para ver si las dos son iguales. Si esta comparación devuelve `true` , el primer bloque de código se ejecuta. Si no, ese código se omite y se ejecuta el segundo bloque de código después de la declaración `else` .

Funciones

Las funciones son una manera de encapsular una funcionalidad que quieres reutilizar, de manera que puedes llamar esa función con un solo nombre, y no tendrás que escribir el código entero cada vez que la utilices. Ya has visto algunas funciones más arriba, por ejemplo:

1. `let nombreDeLaVariable = document.querySelector('h1');`

2. `alert('¡Hola!');`

Estas funciones `document.querySelector` y `alert` están integradas en el navegador para poder utilizarlas en cualquier momento.

Si ves algo que parece un nombre de variable, pero tiene paréntesis — `()` — al final, probablemente es una función. Las funciones con frecuencia toman argumentos —pedazos de datos que necesitan para hacer su trabajo—. Estos se colocan dentro de los paréntesis, y se separan con comas si hay más de uno.

Por ejemplo, la función `alert()` hace aparecer una ventana emergente dentro de la ventana del navegador, pero necesitas asignarle una cadena como argumento para decirle qué mensaje se debe escribir en la ventana emergente.

Las buenas noticias son que podemos definir nuestras propias funciones —en el siguiente ejemplo escribimos una función simple que toma dos números como argumentos y los multiplica entre sí—:

```
function multiplica(num1,num2) {
  let resultado = num1 * num2;
  return resultado;
}
```

Trata de ejecutar la función anterior en la consola. Después trata de usar la nueva función algunas veces, p.ej:

```
multiplica(4, 7);
multiplica(20, 20);
multiplica(0.5, 3);
```

> Nota: la sentencia `return` le dice al navegador que devuelva la variable `resultado` fuera de la función, para que esté disponible para su uso. Esto es necesario porque las variables definidas dentro de funciones, solo están disponibles dentro de esas funciones. Esto se conoce como «ámbito (*scope* en inglés) de la variable». Lee más sobre ámbito o alcance de la variable.

Eventos

Para crear una interacción real en tu sitio web, debes usar eventos. Estos son unas estructuras de código que captan lo que sucede en el navegador, y permite que en respuesta a las acciones que suceden se ejecute un código. El ejemplo más obvio es un clic (click event), que se activa al hacer clic sobre algo. Para demostrar esto, prueba ingresando lo siguiente en tu consola, luego da clic sobre la página actual:

```
document.querySelector('html').onclick = function() {
    alert('¡Ouch! ¡Deja de pincharme!');
}
```

Hay muchas maneras de enlazar un evento a un elemento; aquí hemos seleccionado el elemento `<html>` y le asignamos a su propiedad `onclick` una función anónima (función sin nombre) que contiene el código que se ejecutará cuando el evento suceda.

Nota que

```
document.querySelector('html').onclick = function(){};
```

es equivalente a

```
let miHTML = document.querySelector('html');
miHTML.onclick = function(){};
```

es solo un modo más corto de escribirlo.

Sobrecargar tu sitio web de ejemplo

Ahora vas a repasar un poco lo básico de JavaScript. Añadirás un par de funcionalidades a tu sitio para demostrar lo que puedes hacer.

Añadir un cambiador de imagen

En esta sección añadirás otra imagen a tu sitio usando la DOM API y agregarás un poco de código para cambiar entre imágenes al hacer clic.

1. Primero que todo, busca una imagen que te guste para tu sitio. Asegúrate que sea del mismo tamaño que la primera, o lo más cerca posible.
2. Guarda tu imagen en tu carpeta `images` .
3. Renombra esta imagen «firefox2.png» (sin las comillas).
4. Ve a tu archivo `main.js` y agrega el siguiente JavaScript (si tu JavaScript de *«Hola Mundo»* está aún allí, bórralo).

```
let miImage = document.querySelector('img');
miImage.onclick = function () {
    let miSrc = miImage.getAttribute('src');
    if (miSrc === 'images/firefox-icon.png') {
      miImage.setAttribute('src','images/firefox2.png');
```

```
    } else {
      miImage.setAttribute('src', 'images/firefox-icon.png');
    }
}
```

5. Guarda todos los archivos y carga `index.html` en tu navegador. Ahora cuando hagas clic en la imagen, ¡esta debe cambiar por otra!

Esto fue lo que sucedió: se almacena una referencia a tu elemento `<img>` en la variable `miImage`. Luego, haces que esta propiedad del manejador de evento `onclick` de la variable sea igual a una función sin nombre (una función «anónima»). Ahora, cada vez que se haga clic en la imagen:

1. El código recupera el valor del atributo `src` de la imagen.
2. El código usa una condicional para comprobar si el valor `src` es igual a la ruta de la imagen original:
   i. Si es así, el código cambia el valor de `src` a la ruta de la segunda imagen, forzando a que se cargue la otra imagen en el elemento `<img>`.
   ii. Si no es así (significa que ya fue modificada), se cambiará el valor de `src` nuevamente a la ruta de la imagen original, regresando a como era en un principio.

Añadir un mensaje de bienvenida personalizado

Ahora añadirás un poco más de código, para cambiar el título de la página o incluir un mensaje personalizado de bienvenida para cuando el usuario ingrese por primera vez. Este mensaje de bienvenida permanecerá luego de que el usuario abandone la página y estará disponible para cuando regrese. Lo guardarás usando Web Storage API. También se incluirá una opción para cambiar el usuario y por lo tanto también el mensaje de bienvenida en cualquier momento que se requiera.

1. En `index.html`, agrega el siguiente código antes del elemento `<script>` (en-US):

```
<button>Cambiar de usuario</button>
```

2. En `main.js`, agrega el siguiente código al final del archivo, exactamente como está escrito. Esto toma referencia al nuevo botón que se agregó y al título y los almacena en variables:

```
let miBoton = document.querySelector('button');
let miTitulo = document.querySelector( 'h1');
```

3. Ahora agrega la siguiente función para poner el saludo personalizado, lo que no causará nada aún, pero arreglarás esto en un momento:

```
function estableceNombreUsuario() {
    let miNombre = prompt('Por favor, ingresa tu nombre.');
    localStorage.setItem('nombre', miNombre);
    miTitulo.textContent = 'Mozilla es genial,' + miNombre;
}
```

   La función `estableceNombreUsuario()` contiene una función `prompt()`, que crea un cuadro de diálogo como lo hace `alert()`; la diferencia es que `prompt()` pide al usuario un dato, y almacena este dato en una variable cuando el botón Aceptar del cuadro de diálogo es presionado. En este caso, pedirás

al usuario que ingrese su nombre. Luego, llamarás la API `localStorage`, que nos permite almacenar datos en el navegador y recuperarlos luego. Usarás la función `setItem()` de localStorage, que crea y almacena un dato en el elemento llamado `'nombre'`, y coloca este valor en la variable `miNombre` que contiene el nombre que el usuario ingresó. Finalmente, establecerás el `textContent` del título a una cadena, más el nombre de usuario recientemente almacenado.

4. Luego, agregarás este bloque `if ... else`. Se podría llamar a esto el código de inicialización, como se ha establecido para cuando carga la app por primera vez:

```
if (!localStorage.getItem('nombre')) {
    estableceNombreUsuario();
}
else {
    let nombreAlmacenado = localStorage.getItem('nombre');
    miTitulo.textContent = 'Mozilla es genial,' + nombreAlmacenado;
}
```

   La primera línea de este bloque usa el operador de negación (NO lógico representado por `!`) para comprobar si el elemento `'nombre'` existe. Si no existe, la función `estableceNombreUsuario()` se iniciará para crearlo. Si ya existe (como por ejemplo cuando el usuario ya ingresó al sitio), se recupera el dato del nombre usando `getItem()` y se fija mediante `textContent` del título a la cadena, más el nombre del usuario, como hiciste dentro de `estableceNombreUsuario()`.

5. Finalmente, agrega abajo el evento `onclick` que manipulará el botón, de modo que cuando sea pulsado se inicie la función `estableceNombreUsuario()`. Esto permitirá al usuario establecer un nuevo nombre cada vez que lo desee al pulsar el botón:

```
miBoton.onclick = function() {
    estableceNombreUsuario();
}
```

Ahora cuando visites tu sitio por primera vez, este te pedirá tu nombre y te dará un mensaje personalizado de bienvenida. Puedes cambiar cuantas veces quieras el nombre al presionar el botón. Y como un bonus añadido, ya que el nombre se almacena en el localStorage, este permanecerá después de que cierre el sitio, ¡manteniendo ahí el mensaje personalizado cuando abras el sitio la próxima vez!

¿Un nombre de usuario nulo?

Cuando ejecutes el ejemplo y obtengas el cuadro de diálogo que solicita que introduzcas tu nombre de usuario, intenta pulsar el botón *Cancelar*. Deberías terminar con un título que diga que *Mozilla es genial, null*. Esto sucede porque, cuando cancelas el mensaje, el valor se establece como `null`. Null (nulo) es un valor especial en JavaScript que se refiere a la ausencia de un valor.

Además, prueba a dar clic en *Aceptar* sin introducir un nombre. Deberías terminar con un título que diga que *Mozilla es genial*, por razones bastante obvias.

Para evitar estos problemas, podrías comprobar que el usuario no ha introducido un nombre en blanco. Actualiza tu función `estableceNombreUsuario()` a lo siguiente:

```
function estableceNombreUsuario() {
  let miNombre = prompt('Introduzca su nombre.');
  if(!miNombre) {
```

```
    estableceNombreUsuario();
  } else {
    localStorage.setItem('nombre', miNombre);
    miTitulo.innerHTML = 'Mozilla is genial, ' + miNombre;
  }
}
```

En el lenguaje humano, esto significa que si `miNombre` no tiene ningún valor, ejecute `estableceNombreUsuario()` de nuevo desde el principio. Si tiene un valor (si la afirmación anterior no es verdadera), entonces almacene el valor en `localStorage` y establézcalo como el texto del título.

Conclusión

Si has seguido las instrucciones en este artículo, tendrás una página que luzca como esta (también puede ver nuestra versión aquí):

Si tuviste problemas, siempre puedes comparar su trabajo con el código terminado del ejemplo en GitHub.

Aquí solo has rozado la superficie de JavaScript. Si has disfrutado aprendiendo y deseas avanzar más, visita la Guía de JavaScript.

Ve también

JavaScript

Sumérgete en JavaScript con mucho más detalle.

Aprende JavaScript

¡Este es un excelente material para los aspirantes a desarrolladores web! Aprende JavaScript en un entorno interactivo, con lecciones cortas y pruebas interactivas, guiadas por una evaluación automatizada. Las primeras 40 lecciones son gratis. El curso completo está disponible por un pequeño pago único (en inglés).

En este módulo

Last modified: 5 sept 2022, by MDN contributors