JAVASCRIPT

DISEÑO WEB

Carlos Rojas Sánchez

Licenciatura en Informática

Universidad del Mar

Contenido

Introducción

¿Qué es Javascript?

Javascript es un lenguaje de programación. Es un mecanismo con el que podemos decirle a nuestro navegador que tareas debe realizar. [1]

La consola Javascript

La consola Javascript es una zona del navegador ubicada en las DevTools donde podemos escribir pequeños fragmentos de código y observar los resultados, así como revisar mensajes de información, error u otros detalles similares.

La consola Javascript

El método `console.log()`
Es una función integrada de JavaScript utilizada para imprimir mensajes en la consola del navegador o de un entorno de ejecución como Node.js. Es una herramienta común para depuración y para observar valores durante la ejecución de un programa.

Javascript en línea

Usar la etiqueta <script> para contener las órdenes o líneas de Javascript que le indican al navegador que tiene que hacer.

```
<html>
  <head>
    <title>Título de la página</title>
    <script>
      console.log("¡Hola!");
    </script>
  </head>
  <body>
    <p>Ejemplo de texto.</p>
  </body>
</html>
```

Javascript en externo

Se utiliza la etiqueta $<$ *script* $>$, sólo que en este caso, haremos referencia al archivo Javascript con un atributo src (source),

```
<html>
  <head>
    <title>Título de la página</title>
    <script src="js/index.js"></script>
  </head>
  <body>
    <p>Ejemplo de texto.</p>
  </body>
</html>
```

Ubicación de la etiqueta script

Ubicación	¿Cómo descarga el archivo Javascript?	Estado de la página
En <head>	**ANTES** de empezar a dibujar la página.	Página aún no dibujada.
En <body>	**DURANTE** el dibujado de la página.	Dibujada hasta donde está la etiqueta <script>.
Antes de </body>	**DESPUÉS** de dibujar la página.	Dibujada al 100%.

Javascript

Ed.	Fecha	ECMAScript	Cambios significativos
6	JUN/2015	ES2015	Clases, módulos, hashmaps, sets, for of, proxies...
7	JUN/2016	ES2016	Array includes(), Exponenciación **
8	JUN/2017	ES2017	Async/await
9	JUN/2018	ES2018	Rest/Spread operator, Promise.finally()...
10	JUN/2019	ES2019	flat, flatMap, trimStart(), optional error catch...
11	JUN/2020	ES2020	Dynamic imports, BigInt, Promise.allSettled
12	JUN/2021	ES2021	Promise.any, globalThis, replaceAll(), flat(), flatMap()...
13	JUN/2022	ES2022	at(), top-level await modules, private fields...
14	JUN/2023	ES2023	findLast(), hashbang, toReversed(), toSorted()...
15	JUN/2024	ES2024	groupBy, withResolvers, waitAsync, isWellFormed...

Tipos de datos en Javascript

Tipo de dato	Descripción
Tipos de datos primitivos	
NUMBER Number	Valor numérico (enteros, decimales...)
BIGINT BigInt	Valor numérico muy grande
STRING String	Valor de texto (cadenas de texto, carácteres...)
BOOLEAN Boolean	Valor booleano (valores verdadero o falso)
SYMBOL Symbol	Símbolo (valor único)
UNDEFINED undefined	Valor sin definir (variable sin inicializar)
Tipos de datos no primitivos	
OBJECT Object	Objeto (estructura más compleja)
FUNCTION Function	Función (función guardada en una variable)

Funciones

	Ejemplo	Descripción
1	`function nombre(p1, p2...) { }`	Mediante **declaración** (la más usada por principiantes)
2	`let nombre = function(p1, p2...) { }`	Mediante **expresión** (la más habitual en programadores con experiencia)
3	`new Function(p1, p2..., code);`	Mediante un constructor de **objeto** (no recomendada)

Funciones por declaración

```
function saludar() {
  return "Hola";
}

saludar();      // 'Hola'
```

Funciones por expresión

```
const saludo = function saludar() {
  return "Hola";
};

saludo(); // 'Hola'
```

Funciones como objeto

```
const saludar = new Function("return 'Hola';");

saludar(); // 'Hola'
```

Funciones autoejecutables

```
(function () {
  console.log("Hola!!");
})();
```

```
(function (name) {
  console.log(`¡Hola, ${name}!`);
})("Manz");
```

Funciones Callback

Un callback (llamada hacia atrás) es pasar una función por parámetro a otra función, de modo que esta última función puede ejecutar la función pasada por parámetro de forma genérica desde su propio código, y nosotros podemos definirlas desde fuera de dicha función.

Funciones Callback

```
const action = function () {
  console.log("Acción ejecutada.");
};

const mainFunction = function (callback) {
  callback();
};

mainFunction(action);
```

Funciones de orden superior

Las funciones de orden superior o HOF (High Order Functions) son funciones que reciben por parámetro otra función y/o devuelven una función mediante el return.

Funciones de orden superior

```
const action = function () {
  console.log("Acción ejecutada.");
};

const error = function () {
  console.error("Ha ocurrido un error");
};

const doTask = function (callback, callbackError) {
  const isError = Math.random() < 0.5;

  if (!isError) callback();
  else callbackError();
};

doTask(action, error);
```

Funciones Arrow

Las Arrow functions, funciones flecha o «fat arrow» son una forma corta y compacta de escribir las funciones tradicionales de Javascript. A grandes rasgos, se trata de eliminar la palabra function y añadir el texto $=>$ antes de abrir las llaves.

Funciones Arrow

```
const func = function () {
  return "Función tradicional.";
};

const func = () ⇒ {
  return "Función flecha.";
};
```

```
// 0 parámetros: Devuelve "Función flecha"
const func = () ⇒ "Función flecha.";

// 1 parámetro:  Devuelve el valor de e + 1
const func = (e) ⇒ e + 1;

// 2 parámetros: Devuelve el valor de a + b
const func = (a, b) ⇒ a + b;
```

Función Generadora

Una función generadora es una función que se puede pausar y reanudar, y por lo tanto, nos puede devolver múltiples valores. En lugar de llamar a return para devolver un valor, utilizamos yield.

```
function* generator() {
  // code
  yield "One";
  // code
  yield "Two";
  // code
  yield "Three";
}

const values = generator();
console.log(values.next());
console.log(values.next());
console.log(values.next());
console.log(values.next());
```

Operadores

- Aritméticos
- Asignación
- Unarios
- Comparación
  - Igualdad
  - Identidad
- Lógicos

Eventos mediante HTML

Un evento Javascript es una característica especial que ha sucedido en nuestra página y a la cuál le asociamos una funcionalidad, de modo que se ejecute cada vez que suceda dicho evento.

Eventos mediante HTML

```
<script>
  function doTask() {
    alert("Hello!");
  }
</script>

<button onClick="doTask()">Saludar</button>
```

Referencias i

J. Román.

Lenguaje javascript.

`https://lenguajejs.com/`, 2008.

[Online; accessed 5-October-2024].