MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO DE WEB ENTORNO CLIENTE
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

El lenguaje de programación de clientes

03

ÍNDICE

OBJETIVOS

Conocer los tipos de datos.

Comprender el concepto de variable.

Saber identificar el ámbito de una variable.

Identificar comentarios y etiquetas.

Sabe realizar conversiones entre tipo de datos.

/ 1. Introducción y contextualización práctica

Para comenzar con el desarrollo web en cliente es necesario conocer las tecnologías HTML y JavaScript. Como ya hemos visto, HTML permite definir la estructura de la web. Por su parte, JavaScript permite introducir interacción entre el cliente y nuestro contenido web. Además, HTML se encuadra dentro de los lenguajes de marcado y no se considera un lenguaje de programación. Por el contrario, JavaScript comenzó como un lenguaje de scripting, pues permitía definir pequeños scripts en una página web.

Hoy en día JavaScript se considera un lenguaje de programación completo. De modo que, lo que comenzó como lenguaje de script, se ha convertido en un lenguaje orientado a objetos, con todas las estructuras de un lenguaje de programación avanzado.

Fig. 1. JavaScript como lenguaje de cliente

En este tema presentaremos algunas características de JavaScript como los tipos de datos, el concepto de variable, identificadores, etc.
Escucha el siguiente audio que describe el caso práctico que iremos resolviendo a lo largo de la unidad:

Audio Intro. JavaScript Sistema de tipos
https://bit.ly/3cPpOZg

/ 2. Fundamentos del lenguaje

JavaScript es un lenguaje interpretado por el navegador y que, desde la web 2.0, permite crear páginas dinámicas en las que el usuario puede interactuar con facilidad. Gracias a JavaScript se puede liberar al servidor de realizar ciertas tareas, pues estas se realizan en el cliente. Por ejemplo, si comparamos un formulario en el cliente, el servidor no tiene que hacerlo, lo cual libera a la red y al servidor.

2.1. Relación con HTML

ara poder introducir código JavaScript en un documento web disponemos de las siguientes alternativas:

- **Embeber el código en el documento HTML**. Con esta alternativa seríamos capaces de añadir código JavaScript utilizando la etiqueta <script>.
- **Mediante archivos externos.** Con esta alternativa nos apoyamos en archivos externos que sólo contendrían código JavaScript.
- **A través de elementos HTML**. Normalmente esta alternativa se utiliza para introducir referencias a funcionalidades de JavaScript.

En cualquier desarrollo de software se persigue la facilidad de mantenimiento. Por ello, conviene utilizar archivos externos. En algunos contextos, se puede bloquear la ejecución de JavaScript por razones de seguridad. Esto es importante para que no incluyamos funciones críticas con JavaScript.

Fig. 2.Código JavaScript

2.2. Aspectos básicos del lenguaje

La sintaxis de JavaScript se parece a otros lenguajes de programación como Java. JavaScript se fundamenta en el árbol del documento de la página web. Veremos a continuación los aspectos básicos más interesantes.

2.2.1. Declaración de variables

En un lenguaje de programación, una variable se puede utilizar para almacenar valores. Durante el tiempo de vida de la variable, este valor se puede ver alterado. Es decir, puede ser modificado.

Una variable tiene asignado un nombre que, en un espacio determinado, debe ser único.
Este espacio en el que vive la variable recibe el nombre de ámbito de visibilidad.

El nombre de una variable es, normalmente, «identificador». Si hacemos un símil con las matemáticas, X y Z son variables, pero en un lenguaje de programación se utilizan nombres de indicadores más descriptivos por ejemplo nombre, edad, etc.

2.2.2. Reglas para la definición de identificadores

Algunos lenguajes de programación como JavaScript definen unas reglas sencillas para construir los identificadores de las variables. Algunas de estas normas son las siguientes:

- El identificador puede contener letras, dígitos y guiones bajos.
- El identificador debe comenzar con una letra.

- No se puede utilizar identificadores de palabras reservadas para las variables.
- Los nombres de las variables distinguen entre mayúsculas y minúsculas.

Vídeo 1. Ámbito de visibilidad en JavaScript
https://bit.ly/3dSORM9

2.2.3. El operador de asignación

En JavaScript, el operador de asignación funciona exactamente igual que en otros lenguajes de programación. Una asignación dispone de dos partes: a la izquierda tenemos una variable y a la derecha tenemos el valor asignado. No hay que confundir una asignación con la igualdad matemática ya que ambos casos se utiliza el operador «igual» (=).

```
var x;
var y = 6;
x = 5
var z = x + y;
```
Código 1. Asignaciones

En el ejemplo anterior, se pueden observar tres asignaciones y tres declaraciones de variable. En la primera línea se declara la variable X. En la segunda línea se declara y asigna a la variable Y el valor 6. A continuación, en la línea 3, se asigna a X el valor 5. Por último, a la variable Z se le asigna el resultado de sumar X e Y. Como se puede observar, la asignación funciona diferente a la igualdad matemática.

2.2.4. Tipos de datos

En la mayoría de los lenguajes de programación, existe el concepto de tipo de dato. De esta forma, se puede saber qué rango de valores puede almacenar una variable. En JavaScript se definen diferentes tipos de datos: números, cadenas de caracteres, objetos, etc.
Cuando programamos, conocer el tipo de dato nos aporta seguridad cuando el código se ejecuta.

```
var temp = 16 + “Gato”;
```
Código 2. Distintos tipos de datos

En el ejemplo anterior, se declara la variable temp, a la que se asigna el resultado de sumar 16 y «Gato». En esta expresión, podemos dudar del resultado, ya que 16 es un valor entero y «gato» es una cadena de caracteres. La forma en la que JavaScript interpreta esto es generando la siguiente cadena «16Gato».

Audio 1. JavaScript: Asignación
https://bit.ly/2XQAkLx

/ 3. Caso práctico 1: "Variable address"

Planteamiento. Somos responsables de un equipo de desarrollo web. Un miembro de nuestro equipo nos dice que no entiende qué está pasando en el siguiente código:

```
var num = 10
var street = "sol"
var city = "Madrid";
var address = street + "," + num + "," + city + zip_code;
```

Código 3. Caso práctico de asignaciones

Nudo. ¿Qué crees que puede estar pasando? ¿Qué valor tendrá la variable address al final?

Fig. 3. Buscando la mejor solución para nuestro problema

Desenlace. Como hemos visto en el apartado del operador de asignación, en el código anterior se muestra una serie de asignaciones a distintas variables. Se observa la declaración de cuatro variables: num, street, city y address. Las tres primeras tienen un valor asignado en la declaración de éstas, que son valores fijos. La variable address depende de las anteriores. Sin embargo, esta variable depende de otra variable denominada zip_code. Esta última no está declarada, es decir, no existe. Por ello, durante la ejecución del código JavaScript el resultado obtenido no será el esperado.

/ 4. Sistema de tipado

JavaScript es un lenguaje de programación que incluye un sistema de tipado débil. Por ello, no es necesario declarar el tipo de la variable antes de su utilización, ya que el motor de JavaScript puede averiguar el tipo de una variable cuando se esté ejecutando. Debido a este funcionamiento, es posible tener la misma variable con diferentes tipos.

En el sistema de tipos existen los llamados tipos «primitivos», que son aquellos que ya se proporcionan con el lenguaje. En el caso de JavaScript, los tipos primitivos son los siguientes:

- Boolean: es un tipo que puede almacenar valores lógicos, es decir, true o false.
- Null: es un tipo que contiene el valor nulo.
- Undefined: es un tipo que tiene aquellas variables que no hayan sido definidas previamente.
- Number: este tipo puede almacenar cualquier número. En otros lenguajes se diferencian entre enteros como flotantes y doble precisión. En JavaScript todo se almacena en el mismo.
- String: permite almacenar cadenas de caracteres. Una característica de JavaScript es que las cadenas de caracteres son inmutables, es decir, una vez que tienen valor no puede modificarse. Sin embargo, es posible crear otra cadena que modifique la actual.

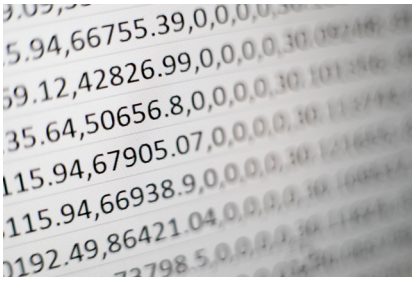

Fig. 4. En JavaScript el tipo numérico incluye a todos los números

Como JavaScript declara variables sin indicar el tipo, puede que en un momento determinado no sepamos cómo operar con dicha variable. En ese contexto, JavaScript nos proporciona el operador typeof. Utilizando este operador podemos saber el tipo de la variable.

4.1. Conversión entre tipo de datos

En JavaScript la conversión de tipos es automática la mayor parte del tiempo y se realiza correctamente. Por ejemplo, cuando mostramos un valor por pantalla, el lenguaje es capaz de convertir la variable a cadena caracteres. En otras ocasiones, el operador que estemos utilizando fijará el tipo de dato resuelto.

Cuando JavaScript no pueda realizar la conversión de tipos, es necesario acudir a la conversión explícita. Al usar este tipo de conversión, el programador es el responsable del resultado obtenido.

```
var age = Number("Valor textual");
alert(age); // NaN, conversion
```
Código 4. Error de conversión

En el ejemplo anterior no es posible convertir un valor textual a un número. Por este motivo, se produce un error en la conversión de tipos. La conversión de tipos de valores lógicos se considera más sencilla.

```
alert( Boolean(1) ); // true
alert( Boolean(0) ); // false
alert( Boolean("hola") ); // true
alert( Boolean("") ); // false
```
Código 5. Conversión de números, cadena y boolean.

En este otro ejemplo, se muestra la conversión de números y cadena de caracteres a valores lógicos. En este caso, no se produce nunca un error, pues la conversión siempre es posible. Esto no significa que el valor sea el esperado.

Por ejemplo, la conversión de 1 devuelve «verdadero» y la de 0 devuelve «falso». Además, la conversión de «hola» devuelve verdadero, y la conversión de la cadena vacía devuelve «falso». En el caso de los valores numéricos puede tener sentido. Sin embargo, la conversión de cadenas puede inducir a error.

Fig. 5. Los tipos de datos pueden convertirse a otros tipos

/ 5. Comentarios

En todos los lenguajes de programación se recomienda comentar el código fuente, ya que si marchamos de vacaciones o cambiamos de proyecto, puede que a la vuelta no sepamos por qué realizamos una cierta tarea en nuestro código. Si somos capaces de comentar correctamente el código, podremos volver sabiendo qué estábamos haciendo y qué razonamiento habíamos seguido. Además, en otras ocasiones puede que nos incorporemos a un proyecto en el que nunca hemos trabajado y, si disponemos de los comentarios o de la documentación, nos supondrá menos esfuerzo adaptarnos.

Para poder documentar/comentar el código, se ofrecen unas herramientas que se denominan «comentarios». JavaScript proporciona dos tipos de comentarios:

- Comentario de línea. Este comentario se refiere a una única línea.
- Comentario multilínea. Este tipo de comentario es capaz de generar bloques.

```
var z = 2 ; // Declara z y le asigna 2.
var y = z + 2; // Declara y, y le asigna z + 2
```
Código 6. Comentarios de línea

Como se puede observar, en el ejemplo anterior existen comentarios de línea. En este caso, a la derecha de cada línea se indica qué se ha realizado.

```
/* 1. Declara z y le asigna 2.
2. Declara y, y le asigna z + 2
/*
var z = 2 ;
var y = z + 2;
```

Código 7. Comentario en bloque

Es importante realizar el mantenimiento de los comentarios pues, si el código cambia, los comentarios tendrán que cambiar. En muchas ocasiones, esto no se realiza, lo cual genera grandes problemas a la hora de incorporarse a un nuevo proyecto. Por este motivo, anotar los cambios que se realicen a través de los comentarios es una buena práctica.

/ 6. Otras tareas que puede realizar JavaScript

En la actualidad JavaScript se considera un lenguaje de programación seguro, pues no tiene acceso a la memoria ni a la CPU. Sin embargo, esto depende del entorno en el que se ejecute y en el que lo usemos. Aunque inicialmente JavaScript fue creado para entornos web, hoy en día se puede usar para otras funciones, por ejemplo, en Node.js.

Para el desarrollo en cliente nos centraremos en lo que JavaScript puede realizar «dentro» del navegador:

- Es posible añadir nuevos elementos HTML, cambiar el contenido y modificar su apariencia visual (estilos).
- Nos proporciona mecanismos para reaccionar a eventos como por ejemplo clics, movimiento ratón y pulsación de teclas.
- A la hora de comunicar con el servidor, JavaScript nos proporciona mecanismos de comunicación asíncronos o síncronos.
- Como el protocolo HTTP no puede guardar el estado de una comunicación, JavaScript ofrece mecanismos para guardar datos como cookies y almacenamiento local.

Fig. 6. Las funciones extendidas de JavaScript lo hacen destacar

/ 7. Limitaciones de JavaScript

La funcionalidad de JavaScript dentro del navegador está limitada por cuestiones de seguridad. La idea principal es evitar el acceso malicioso a determinados datos y evitar comportamientos indeseados. Alguna de las restricciones que se incluyen en JavaScript son las siguientes:

- Dentro del entorno de ejecución de un navegador JavaScript no puede acceder al disco duro ni ejecutar un programa. De esta forma, JavaScript no tiene acceso directo al sistema operativo.
- Es posible que JavaScript interactúe con la cámara o con el micrófono de nuestro sistema. Sin embargo, para poder realizar esta tarea, primero debe solicitar permiso de forma explícita al usuario.
- En los navegadores actuales es posible tener más de una pestaña o ventana abierta al mismo tiempo. JavaScript no puede acceder al contenido de una ventana o pestaña diferente a la que está ejecutando si no provienen del mismo dominio. Esto se conoce como Same Origin Policy.

/ 8. Caso práctico 2: “Procesamiento de una respuesta del servidor”

Planteamiento. Te asignan un proyecto para el desarrollo en JavaScript. El cliente te solicita que, durante el procesamiento de una respuesta del servidor, la página web acceda al sistema de archivos de la máquina y cambie el contenido del archivo de facturación.

Nudo. ¿Cómo podrías realizar esto? ¿Qué tendrías que responder al cliente?

Desenlace. Como hemos visto en el apartado sobre las limitaciones de JavaScript, este lenguaje, cuando se ejecuta en un entorno de navegador, no puede acceder a algunas tareas del sistema operativo. Es cierto que podría acceder a algunos dispositivos, pero cuando se intenta acceder al sistema de archivos, el navegador lo bloquea por razones de seguridad. Esto no quiere decir que JavaScript no pueda acceder al sistema archivos, pues solo ocurre cuando se ejecuta en un entorno de navegador. Si ejecutamos JavaScript en otros contextos, tendríamos acceso completo al sistema archivos y podríamos realizar lo que el usuario nos solicita.

Fig. 7. JavaScript permite implementar grandes funcionalidades

/ 9. Resumen y resolución del caso práctico de la unidad

Gracias a JavaScript se puede mejorar la interacción del usuario con la página web. A lo largo de este tema hemos presentado las principales características del lenguaje JavaScript para ejecutarlo en entornos de navegador.
Durante este tema hemos definido también el concepto de variable, tipo de datos, conversión de tipos y comentarios. Recordad que, para el manejo de variables, es importante saber cómo definirlas en cada lenguaje, es decir, conocer las reglas de creación de identificadores. Tened en cuenta también que el concepto de tipo de datos es muy importante a la hora trabajar con variables, pues define el dominio de esa variable.

Con el fin de mantener el código fuente hemos visto que es necesario introducir aclaraciones a través de comentarios.

Resolución del caso práctico de la unidad
Como hemos visto en este tema, JavaScript es un lenguaje de programación no seguro en cuanto a los tipos, pues se pueden producir errores de tipo en tiempo de ejecución. Cuando realizamos una asignación de una variable y la parte izquierda toma valor de NaN, quiere decir que hemos realizado una conversión errónea de tipos. En concreto el error NaN, se traduce como Not a Number, lo que quiere decir que estamos realizando una conversión a número de algo que no es un número. Como es lógico, el lenguaje no puede realizar esta conversión y lo comunica de esta forma. Uno de los problemas que tiene JavaScript es precisamente que avisa tarde de los errores.
Para poder solucionar esto, antes de publicar cualquier aplicación se hacen pruebas exhaustivas que puedan garantizar que no se van a producir errores cuando el usuario esté trabajando con la misma.

/ 10. Bibliografía

Duckett, J. & Duckett, J. (2014). Web design with HTML, CSS, JavaScript and jQuery. Hoboken: John Wiley & Sons.

Gómez, M. (2017). Curso de Desarrollo Web: HTML, CSS y JavaScript. Madrid: Anaya Multimedia.