



Desarrollo web en entorno cliente

Juan Carlos Moreno Pérez



Contenidos digitales
www.sintesis.com



D esarrollo web en entorno cliente

2.^a Edición

El módulo de Desarrollo Web en Entorno Cliente está relacionado con el certificado de profesionalidad IFCD0210 (cualificación profesional de referencia IFC154_3: Desarrollo de aplicaciones con tecnologías web):

- Unidad de competencia UC0491_3: Desarrollar elementos software en el entorno cliente.

Consulte nuestra página web: www.sintesis.com
En ella encontrará el catálogo completo y comentado



Queda prohibida, salvo excepción prevista en la ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de la propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sigs. Código Penal). El Centro Español de Derechos Reprográficos (www.cedro.org) vela por el respeto de los citados derechos.

Desarrollo web en entorno cliente

Juan Carlos Moreno Pérez

2.^a Edición



ASESOR EDITORIAL:

Juan Carlos Moreno Pérez

© Juan Carlos Moreno Pérez

© EDITORIAL SÍNTESIS, S. A.
Vallehermoso, 34. 28015 Madrid
Teléfono 91 593 20 98
<http://www.sintesis.com>

ISBN: 978-84-9171-490-3
Depósito Legal: M-12.331-2022

Impreso en España - Printed in Spain

Reservados todos los derechos. Está prohibido, bajo las sanciones penales y el resarcimiento civil previstos en las leyes, reproducir, registrar o transmitir esta publicación, íntegra o parcialmente, por cualquier sistema de recuperación y por cualquier medio, sea mecánico, electrónico, magnético, electroóptico, por fotocopia o por cualquier otro, sin la autorización previa por escrito de Editorial Síntesis, S. A.

Índice

PRESENTACIÓN	11
1. SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN	13
Objetivos	13
Mapa conceptual	14
Glosario	14
1.1. Introducción	15
1.2. Front-end y back-end	15
1.3. Lenguajes de programación en entorno cliente	16
1.3.1. ReactJS	17
1.3.2. AngularJS	17
1.3.3. Vue.js	17
1.4. Características de los lenguajes de script	18
1.5. Integración de JavaScript dentro de HTML	19
1.5.1. Ejemplo de JavaScript en ficheros js separados	20
1.5.2. Ejemplo de JavaScript con código dentro del HTML	21
1.6. Herramientas de programación en JavaScript	22
1.6.1. Herramientas online	22
1.6.2. Utilización de IDE y sistemas de control de versiones	22
1.7. Posibilidades que ofrece JavaScript	24
1.7.1. Modificación del contenido de una página web	24

1.7.2. Cambiar atributos de objetos HTML	24
1.7.3. Cambiar el estilo CSS	26
1.8. Comunicación de JavaScript con el exterior	26
1.8.1. Escribir en la consola del navegador utilizando console.log()	27
1.8.2. Escribir en cualquier elemento HTML utilizando el atributo innerHTML	27
1.8.3. Generar directamente HTML utilizando el método document.write()	27
1.8.4. Generar un mensaje de alerta utilizando el método window.alert()	28
Resumen	28
Ejercicios propuestos	29
Actividades de autoevaluación	30
2. SINTAXIS DEL LENGUAJE JAVASCRIPT	33
Objetivos	33
Mapa conceptual	34
Glosario	34
2.1. Sintaxis del lenguaje. Operadores y palabras reservadas	35
2.1.1. Las 10 reglas básicas de la sintaxis del lenguaje	35
2.1.2. Las palabras reservadas de JavaScript	36
2.1.3. Uso de variables en JavaScript	37
2.1.4. Operadores en JavaScript	38
2.2. Tipos de datos. Asignaciones y expresiones	40
2.2.1. El valor null	40
2.2.2. Conversiones entre tipos de datos en JavaScript	40
2.3. Introducción a las funciones	41
2.4. Introducción a los objetos en JavaScript	42
2.5. Variables y ámbitos de utilización	43
2.5.1. Las diferencias entre var y let	44
2.6. Eventos	44
2.6.1. Tipos de eventos en JavaScript	45
2.6.2. Los listener	46
2.7. Objeto string o cadena de caracteres	47
2.8. Números	47
2.8.1. Convertir un número en un string	47
2.8.2. Ajuste de decimales. Número de decimales de un número	48
2.8.3. Precisión de un número	48
2.8.4. Convertir cualquier variable en un número	48
2.9. Fechas	48
2.9.1. Creación de un objeto de tipo fecha	49
2.9.2. Formatos de tipo fecha	49
2.9.3. Métodos del tipo fecha	49
2.10. Arrays	50
2.10.1. Operaciones con arrays	51
2.10.2. Arrays multidimensionales	51
2.11. Sentencias condicionales	52
2.12. Bucles	54
2.12.1. Recorrido de un array	55
2.13. Práctica guiada	56
Resumen	59
Ejercicios propuestos	60
Actividades de autoevaluación	61

3. UTILIZACIÓN DE LOS OBJETOS PREDEFINIDOS DEL LENGUAJE	63
Objetivos	63
Mapa conceptual	64
Glosario	64
3.1. Introducción	65
3.2. Manejo del tiempo en JavaScript	65
3.2.1. Las funciones setTimeout y setInterval	67
3.3. Cookies	69
3.3.1. ¿Para qué se utilizan las cookies?	69
3.3.2. ¿Cómo crear una cookie?	70
3.3.3. ¿Cómo leer las cookies?	70
3.4. Almacenamiento local	71
3.4.1. El objeto localStorage	71
3.5. Objetos en JavaScript	72
3.5.1. Recorrer la información de un objeto	73
3.5.2. Constructores de JavaScript	74
3.6. Funciones en JavaScript	75
3.6.1. La recursividad en JavaScript	76
3.6.2. Los parámetros de las funciones	76
3.6.3. Funciones y métodos en objetos de tipo array	77
3.6.4. Funciones y métodos en objetos y variables de tipo string	82
3.6.5. Funciones globales del lenguaje JavaScript relativas a números	84
3.6.6. Propiedades globales de JavaScript	85
3.6.7. Algunas funciones globales de JavaScript	86
3.7. Prácticas guiadas	87
3.7.1. Uso de estructuras JavaScript: creación de un generador automático de historias	87
3.7.2. Utilización avanzada de arrays: generación de un sudoku aleatorio	91
Resumen	97
Ejercicios propuestos	97
Actividades de autoevaluación	101
4. EL DOM Y EL BOM	103
Objetivos	103
Mapa conceptual	104
Glosario	104
4.1. Introducción	104
4.2. Formularios	105
4.2.1. La validación de campos	105
4.3. Expresiones regulares	106
4.4. DOM	108
4.4.1. El acceso al DOM. El método document.querySelector()	113
4.5. Eventos del DOM	115
4.6. Manejadores de eventos	117
4.7. Nodos del DOM	118
4.7.1. ¿Cómo se accede a los nodos (elementos HTML)?	118
4.7.2. ¿Cómo se crea un nuevo nodo?	119
4.7.3. ¿Cómo se elimina un nodo?	120
4.7.4. Práctica guiada: creación y eliminación de elementos	121

4.8. Propiedades del DOM	123
4.8.1. El acceso a los nodos	123
4.8.2. Capturar ciertos eventos	123
4.9. Modificando el DOM	124
4.9.1. Cambiar el tamaño del texto	125
4.9.2. Mostrar y ocultar elementos de una página web	126
4.9.3. Deshabilitar objetos	128
4.9.4. Comprobar que se introduce información en un campo de texto	129
4.10. BOM	130
4.10.1. El objeto window	131
4.10.2. El objeto location	132
4.10.3. El objeto history	132
4.10.4. El objeto navigator	132
Resumen	133
Ejercicios propuestos	135
Actividades de autoevaluación	135
5. MECANISMOS DE COMUNICACIÓN ASÍNCRONA	139
Objetivos	139
Mapa conceptual	140
Glosario	140
5.1. Introducción	140
5.2. Los web workers	141
5.3. JSON	142
5.3.1. Sintaxis JSON	143
5.3.2. Práctica guiada: instalación de un servidor JSON	144
5.4. Comunicación asíncrona. AJAX	146
5.4.1. Práctica guiada: sistema de localización de ciudades	148
5.4.2. Ejemplo con AJAX y JSON	151
5.4.3. Ejemplo con AJAX y XML	153
5.5. Comunicación asíncrona con el servidor. AJAX y jQuery	154
5.5.1. Carga simple de datos load()	155
5.5.2. Llamada asíncrona utilizando POST	156
5.6. Práctica guiada: encuestas interactivas con AJAX	158
Resumen	162
Ejercicios propuestos	163
Actividades de autoevaluación	165
6. REACTJS: UN FRAMEWORK DE JAVASCRIPT	167
Objetivos	167
Mapa conceptual	168
Glosario	168
6.1. Introducción a React	169
6.1.1. Creación de la primera aplicación React	170
6.1.2. Ejecución de la aplicación React	171
6.2. Análisis del contenido de la primera aplicación	172
6.2.1. El fichero index.html	172
6.2.2. El fichero index.js	172

6.2.3. El fichero src/App.js	173
6.3. Introducción a JSX	174
6.4. Componentes React	175
6.4.1. Class components y function components	176
6.4.2. Componentes contenedores y presentaciones	178
6.4.3. Creando el componente principal. Conversor euro-dólares	179
6.5. Los componentes. El state	180
6.5.1. El state y los métodos	180
6.5.2. Los componentes hijo y la comunicación padre-hijo: las props	182
6.6. El DOM virtual de React	184
6.6.1. Renderización y cambios	184
6.7. Los callbacks	185
6.8. Prácticas guiadas	187
6.8.1. Los reyes godos	187
6.8.2. Conversor de dólares a euros y viceversa	191
Resumen	192
Ejercicios propuestos	193
Actividades de autoevaluación	201

RECURSOS DIGITALES



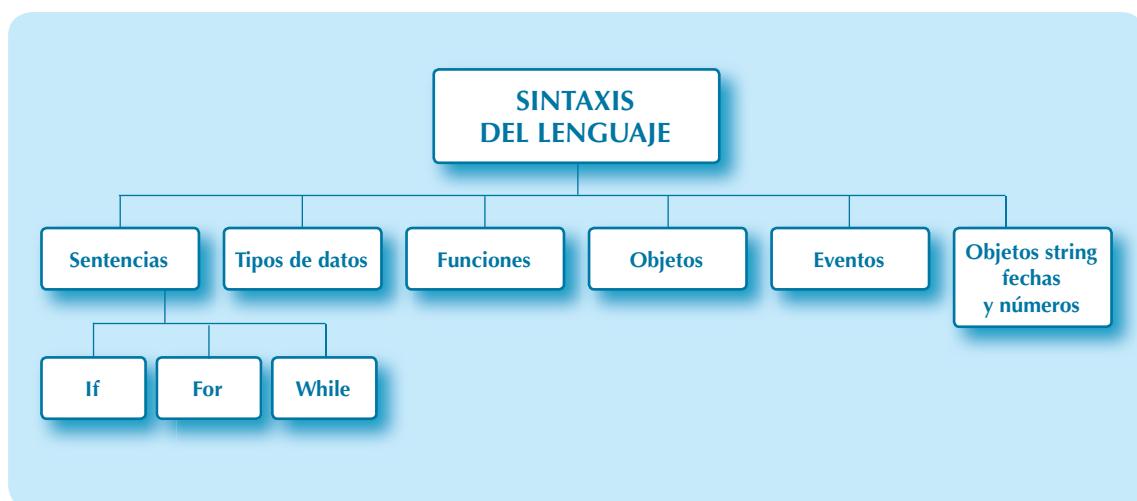
Archivo libro_cliente.rar, que incluye los recursos necesarios para trabajar los ejemplos que aparecen en el libro

Sintaxis del lenguaje JavaScript

Objetivos

- ✓ Comprender la sintaxis básica de JavaScript para poder realizar pequeños scripts funcionales.
- ✓ Entender las reglas básicas del lenguaje JavaScript.
- ✓ Conocer la utilización de las sentencias básicas de JavaScript.
- ✓ Descubrir elementos propios de JavaScript como los eventos.

Mapa conceptual



Glosario

Case-sensitive. Expresión informática que se aplica a los textos en los que tiene relevancia escribir un carácter en mayúsculas o minúsculas y que significa “sensible a mayúsculas y minúsculas”.

Consola de JavaScript. Herramienta del programador para comunicar su programa JavaScript con el exterior. Se invoca pulsando generalmente la tecla F12.

ES6 o ES2015 (ECMAScript). Especificación de lenguaje JavaScript publicada por ECMA International. Los navegadores utilizan una implementación de ECMAScript y acceso al DOM para manipular las páginas web.

GMT (Greenwich Mean Time). Estándar de tiempo que se refería al tiempo solar medio en el Real Observatorio de Greenwich y que dejó de ser utilizado por la comunidad científica en 1972, cuando se reemplazó por el UTC.

Listener. Código responsable de controlar los eventos. Están a la escucha (de ahí su nombre) y, cuando ocurre un evento, se ejecuta el código que el programador haya implementado.

Operador ternario. Operador que toma tres argumentos. La ventaja que ofrece es que puede reducir varias líneas de código en una sola.

UTC (Coordinated Universal Time). Principal estándar de tiempo, que casi siempre es sinónimo de GMT, y se obtiene a través del tiempo atómico internacional.

2.1. Sintaxis del lenguaje. Operadores y palabras reservadas

La sintaxis de JavaScript es muy parecida a Java y C++. Cualquier programador que sepa programar en Java, PHP u otro lenguaje con sintaxis similar será capaz de comprender la sintaxis de JavaScript de una manera rápida. No obstante, al ser JavaScript un lenguaje de programación del lado del cliente, es importante conocer sus aspectos básicos como pueden ser los eventos, el control de los elementos HTML, etc.

RECUERDA

- ✓ Cualquier programa JavaScript puede comunicarse con la consola del navegador mediante la sentencia:

```
console.log("información para la consola");
```

En la consola, se pueden escribir literales y también valores de variables.

Todos los navegadores tienen consola y es muy utilizada por los programadores con fines de depuración.

2.1.1. Las 10 reglas básicas de la sintaxis del lenguaje

A continuación, se presentan las diez reglas básicas del lenguaje JavaScript:

- *Regla 1.* Las instrucciones en JavaScript terminan en un punto y coma. Ejemplo:

```
var s = "hola";
```

- *Regla 2.* Uso de decimales en JavaScript. Los números en JavaScript que tengan decimales utilizarán el punto como separador de las unidades con la parte decimal. Ejemplos de números:

```
var x = 4;
var pi = 3.14;
```

- *Regla 3.* Los literales se pueden escribir entre comillas dobles o simples. Ejemplo:

```
var s1 = "hola";
var s2 = 'hola';
```

- *Regla 4.* Cuando sea necesario declarar una variable, se utilizará la palabra reservada *var*.
- *Regla 5.* El operador de asignación, al igual que en la mayoría de lenguajes, es el símbolo igual (=).
- *Regla 6.* Se pueden utilizar los siguientes operadores aritméticos: (+ - * /). Ejemplo:

```
var x = (5*4)/2+1;
```

- *Regla 7.* En las expresiones, también se pueden utilizar variables. Ejemplo:

```
var t = 4;
var x = (5*t)/2+1;
var y;
y = x * 2;
```

- *Regla 8.* Comentarios en JavaScript. Existen dos opciones para comentar el código:

- // cuando se desea comentar el resto de la línea a partir de estas dos barras invertidas.
- /* y */. todo lo contenido entre ambas etiquetas quedará comentado.

- *Regla 9.* Los identificadores en JavaScript comienzan por una letra o la barra baja (_) o el símbolo del dólar (\$).
- *Regla 10.* JavaScript es sensible a las mayúsculas y minúsculas (case-sensitive). Ejemplo:

```
var nombre = "Julio";
var Nombre = "Ramón";
```



TOMA NOTA

- *Nombre* y *nombre* son dos variables diferentes.
- Ten cuidado al escribir la palabra reservada *var*. Si escribes *Var* o *VAR*, tu código no funcionará.

2.1.2. Las palabras reservadas de JavaScript

Algunas de estas palabras reservadas se han visto ya y otras se trabajarán a lo largo del libro. En el cuadro 2.1, se muestran las más utilizadas.

CUADRO 2.1

Palabras reservadas más comunes de JavaScript

Palabra	Descripción
<code>var</code>	Utilizada para declarar una variable.
<code>if ... else</code>	Estructura condicional.
<code>for</code>	Estructura de repetición. Se ejecutará mientras la condición sea verdadera.
<code>do ... while</code>	Estructura de repetición. Se ejecutará mientras la condición sea verdadera.
<code>switch</code>	Serie de sentencias que van a ser ejecutadas dependiendo de diferentes circunstancias.

[.../...]

CUADRO 2.1 (CONT.)

<code>break</code>	Termina un switch o un bucle.
<code>continue</code>	Sale del bucle y se coloca al comienzo de este.
<code>function</code>	Declara una función.
<code>return</code>	Sale de una función.
<code>try ... catch</code>	Utilizadas para el manejo de excepciones.

2.1.3. Uso de variables en JavaScript

Las variables son la base de la programación. Una variable es una posición de memoria donde se pueden alojar datos. El nombre de la variable permitirá a JavaScript poder ubicar y localizar dicho espacio cuando interprete los scripts. En este apartado, se mostrará el uso de las variables en JavaScript.

FUNDAMENTAL

Constantes en JavaScript

En JavaScript, a partir de ES6 (ES2015), se puede utilizar la palabra reservada `const`. Por lo tanto, en vez de utilizar:

```
var pi = 3.141592;
```

se aconseja emplear:

```
const PI = 3.141592;
```

dado que `pi` es una constante (valor invariable). Por convención, se suele utilizar mayúsculas cuando se definen constantes.

En JavaScript, se pueden ejecutar las siguientes órdenes:

```
var x;
x = 2 * x + 1;
var pi = 3.141592;
var paginaweb = "Myfpschool";
var pregunta = '¿cuantos años tienes?', respuesta="veinte";
paginaweb="Myfpschool.com";
paginaweb="Myfpschool" + "." + "com";
```

RECUERDA

- ✓ El resultado a la derecha de una asignación se almacena en la variable del lado izquierdo de esta.

2.1.4. Operadores en JavaScript

A continuación, se muestran distintos tipos de operadores utilizados en JavaScript.

A) Operadores aritméticos

Todos los lenguajes de programación tienen operadores y, generalmente, suelen coincidir (salvo los operadores incremento y decremento que aparecieron con C++, siendo uno de sus rasgos más significativos).

CUADRO 2.2

Operadores aritméticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
++	Incremento
--	Decremento

B) Operadores de asignación

Otros operadores básicos son los operadores de asignación. En el cuadro 2.3, se muestran los operadores de asignación de JavaScript.

CUADRO 2.3

Operadores de asignación

Operador	Ejemplo de uso
=	x = y;
+=	x += y; // igual que (x = x + y)
-=	x -= y; // igual que (x = x - y)
*=	x *= y; // igual que (x = x * y)
/=	x /= y; // igual que (x = x / y)
%=	x %= y; // igual que (x = x % y)

C) Operadores de manejo de strings

En JavaScript, se utilizan los operadores + y += para concatenar strings. Véase un ejemplo de uso:

```
var = "hola" + " " + "mundo";
```

O lo que sería igual:

```
var = "hola";
var += " ";
var += "mundo";
```

D) Operadores lógicos y de comparación

Tanto los operadores lógicos como los de comparación son profusamente utilizados por los programadores. En el cuadro 2.4, se muestran los operadores tanto lógicos como de comparación para los programadores JavaScript.

CUADRO 2.4
Operadores lógicos y de comparación

Operador	Descripción
==	Igual que
===	Igual valor y tipo
!=	Distinto
!==	Distinto valor o distinto tipo
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
?	Operador ternario

E) Operadores de tipo

Los operadores de tipo permiten conocer si un objeto es una instancia de un tipo concreto o bien conocer el tipo de una variable. A continuación, se muestran los operadores de tipo disponibles en JavaScript:

- *typeof*. Devuelve el tipo de una variable.
- *instanceof*. Devuelve true si un objeto es una instancia de un tipo de objeto.

2.2. Tipos de datos. Asignaciones y expresiones

JavaScript es un lenguaje bastante simplificado en lo que a tipos de datos se refiere y, por ello, solamente tiene cinco tipos de datos:

1. String.
2. Number.
3. Boolean.
4. Array.
5. Object.

Véase un ejemplo de utilización de cada uno de estos tipos:

```
var edad = 25; // Number
var nombre = "Dimas"; // String
var asignatura = ["lengua", "mate", "cono"]; // Array
var persona = {nombre:"Dimas", apellido:"Moreno"}; // Objeto
```

“Dimas” o “lengua” son literales y, por eso, aparecen entrecomillados.

Para JavaScript, las siguientes dos líneas de código son iguales:

```
var dato = "Ronaldo " + 10;
var dato = "Ronaldo " + "10";
```

JavaScript interpretará el número como un string.

2.2.1. El valor null

Null para los lenguajes de programación es *nada* o algo que no existe. Generalmente, cuando se asigna null a una variable, es porque es o será un objeto.

Véase un ejemplo de uso del null:

```
var persona = null;
persona = {nombre:"Dimas", apellido:"Moreno"};
```

2.2.2. Conversiones entre tipos de datos en JavaScript

JavaScript es un lenguaje interpretado, por lo tanto, la conversión entre un tipo de dato y otro es transparente al programador. JavaScript asignará el tipo de datos más acertado a la variable que el programador esté utilizando.

No obstante, existen funciones de conversión como String() y Number() para convertir a cadenas de caracteres y números, respectivamente.

También existen las funciones:

- *parseInt()*. Que convierte una cadena a un número entero.
- *parseFloat()*. Que convierte una cadena a un número decimal.

Véase un ejemplo de uso de estas funciones:

```
alert("Entero: " + parseInt("9.99")); //mostrará Entero: 9
alert("Float: " + parseFloat("9.99")); //mostrará Float: 9.99
```

Es posible también convertir variables de tipo fecha a string con la función `toDateString()`. A continuación, se muestra un ejemplo de dicha conversión:

```
var fecha = new Date();
var cadena = fecha.toDateString(); // Ahora cadena contendrá la fecha
actual "Sun, 13 Jan 2019".
```

Si lo que se desea es convertir la hora en formato UTC, se puede utilizar alternativamente la función `toUTCString()`. Se muestra un ejemplo de dicha conversión:

```
var fecha = new Date();
var cadena = fecha.toUTCString(); // Ahora cadena contendrá la fecha actual en formato UTC "Sun, 13 Jan 2019 07:45:49 GMT".
```

2.3. Introducción a las funciones

Las funciones son uno de los elementos de la base de la programación estructurada. Reutilizar el código es algo básico en cualquier lenguaje de programación porque la regla que se debe seguir es “escribe una vez y ú lízala tantas veces como puedas”. La variación del resultado de las funciones dependerá de sus argumentos.

Un ejemplo de estructura básica de una función es el siguiente:

```
function suma(a, b) {
    return a + b;
}
```

Como se puede ver, se tiene la palabra reservada *function* para que JavaScript interprete que se encuentra frente a una función seguida de su nombre (en el ejemplo anterior, *suma*). Esta función aceptará dos parámetros *a* y *b* y devolverá (return) la suma de ambos.

TOMA NOTA



- Una función puede tener cero, uno o varios parámetros.
- El código de la función estará dentro de las llaves {} y }.
- La función devolverá el resultado con la sentencia `return`.

Véase un ejemplo de utilización de la función anterior:

```
var c = suma(3,3); // c valdrá 6
c = suma(c,3); // ahora c valdrá 9
var text = "El valor de c será ahora: "+c;
```

Actividad propuesta 2.1



Vista la función suma, realiza la función resta, multiplicación y división.

2.4. Introducción a los objetos en JavaScript

Desde hace mucho tiempo, la programación estructurada era el único paradigma efectivo y eficiente en programación, pero las cosas cambiaron y nació la programación orientada a objetos, que rompía con lo establecido. No era una evolución, era una nueva filosofía que no tenía nada que ver con lo anterior.

Hoy en día, es prácticamente imposible programar sin utilizar programación orientada a objetos. Esta forma de programar es más cercana a cómo se expresarían las cosas en la vida real que en otros tipos de programación clásicos.

Analistas y programadores piensan y describen las realidades y el entorno de una manera distinta para plasmar dichos conceptos en programas en términos de objetos, atributos y métodos.

Véase cómo se definiría un objeto en JavaScript:

```
var perro = {  
    raza:"Podenco",  
    peso:12,  
    altura:58,  
    color:"negro"  
};
```

Como se puede observar, un objeto, al igual que en la vida real, puede tener muchas características o atributos, y cada atributo se verá reflejado en una variable dentro del objeto. Un objeto se diferenciará de otros por el valor de sus atributos.

También se pueden encontrar definiciones de objetos en una sola línea (aunque la forma anterior parece más legible):

```
var perro = {raza:"Podenco", peso:12, altura:58,color:"negro"};
```

Para acceder a los atributos de un objeto, se puede hacer de la siguiente manera:

```
perro.raza;
```

O bien:

```
perro["raza"];
```



Actividad propuesta 2.2

Crea una página web con un script que contenga un objeto de la clase persona, la cual tendrá los siguientes atributos:

- Nombre: Dimas.
- Edad: 28.
- Altura: 185.

Una vez creado el objeto, se deberá mostrar por consola: "El usuario Dimas tiene 28 años y mide 185 centímetros".

Todo ello utilizando los atributos del objeto.

2.5. Variables y ámbitos de utilización

A continuación, se estudiará la diferencia que existe entre variables globales y locales en JavaScript. Véase un ejemplo:

```
var a; // Esta es una variable global
a=10;

function suma() {
    var b = 5;
    return a + b; // la función devolverá 15
}
```

La diferencia entre las variables *a* y *b* es que *a* puede utilizarse en otras funciones que se puedan crear, mientras que *b* solamente existirá dentro de la función suma.

¿Qué es lo que ocurre cuando no se declara una variable como la variable *d* en el siguiente código?

```
var a; // Esta es una variable global
a=10;
function suma() {
    var b = 5;
    c = 20;
    return a + b; // la función devolverá 15
}
```

Que será global a todo el JavaScript de esa página.

RECUERDA

- ✓ Las variables, y obviamente sus valores, desaparecen cuando se cierra la página web.

2.5.1. Las diferencias entre var y let

JavaScript permite declarar las variables con var y let. La diferencia entre ambas palabras reservadas es que, cuando se declara una variable con var, la variable puede utilizarse fuera del bloque donde fue declarada. Véase un ejemplo de esto:

```
var a = 10; // x vale 10
{
    let a = 5; // a vale 5
}
// aquí a vale 10
{
    let b=3;
}
// aquí no se puede utilizar b
{
    var c=7;
}
// aquí sí se puede utilizar c y valdrá 7.
```

⌚ FUNDAMENTAL

Las palabras reservadas let y const

El estándar ES2015 añadió las palabras clave *let* y *const*. La primera se utiliza para declarar variables en un bloque determinado y la segunda, para definir constantes. Anteriormente, en JavaScript, las variables estaban definidas a nivel global o a nivel de función.

2.6. Eventos

Los eventos son cualquier suceso que le pueda ocurrir a un elemento HTML. Como se puede suponer, JavaScript puede darse cuenta de ese evento y reaccionar a este ejecutando el código que se haya programado.

Algunos eventos que pueden ocurrir en una página web son, entre otros:

- Pulsar un botón.
- Modificar un campo de texto.
- Pulsar una tecla.
- La página ha terminado de cargarse.
- Hacer clic sobre un elemento HTML.

El formato de programar un evento en JavaScript sería el siguiente:

```
<Elemento_HTML evento='código_JavaScript'>
```

También es posible cambiar las comillas simples por comillas dobles. Se pueden utilizar de forma indistinta.

Véase un ejemplo de evento en JavaScript programado para un botón:

```
<button onclick='this.innerHTML=Date()'>Pulsa para saber la hora</button>
```

2.6.1. Tipos de eventos en JavaScript

En un navegador, hay muchos tipos de eventos que pueden ser controlados por JavaScript. En los siguientes cuadros, se muestra una clasificación de los distintos tipos de eventos clasificados por campos de texto, ratón y carga:

CUADRO 2.5
Eventos en campos de texto

Evento	Ejecutado
onblur	Cuando se abandona un campo de entrada.
onchange	Cuando se cambia el contenido de un campo de entrada o cuando un usuario selecciona un valor de una lista desplegable.
onfocus	Cuando obtiene el foco un campo de entrada.
onselect	Cuando se selecciona el texto de entrada.
onsubmit	Cuando se hace clic en el botón de enviar.
onreset	Cuando se hace clic en el botón de reinicio.
onkeydown onkeypress	Cuando se presiona o mantiene presionada una tecla.
onkeyup	Cuando se deja de presionar una tecla.

CUADRO 2.6
Eventos relativos al ratón

Evento	Ejecutado
onclick	Cuando se hace clic en un botón.
ondblclick	Cuando se hace doble clic en un texto.
onmouseover onmouseout	Cuando el ratón pasa sobre un elemento.
onmousedown onmouseup	Al presionar o soltar un botón del ratón.
onmousemove	Al mover el puntero del ratón sobre de una imagen.
onmouseout	Al mover el puntero del ratón fuera de una imagen.
onmouseover	Al mover el ratón sobre una imagen.
onmouseout	Al mover el ratón fuera de una imagen.