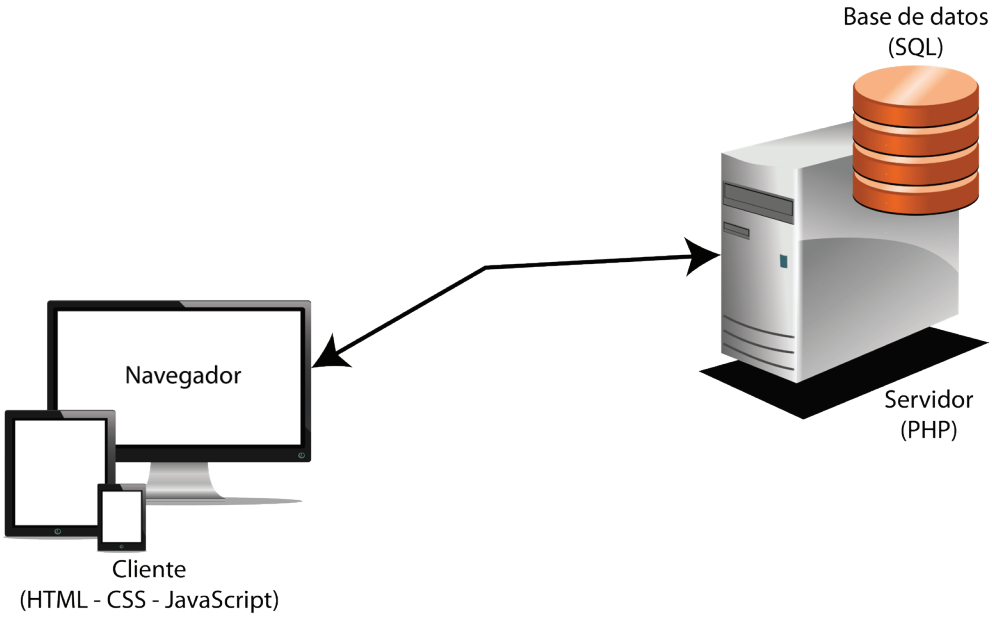
INTRODUCCIÓN

Cuando hablamos de web, es obligatorio dividirlo en dos partes bien diferenciadas: Cliente y servidor. El cliente es quien solicita el servicio y el servidor se encarga de satisfacer estas demandas.

Y ya si nos centramos en sitios web y su programación, el área de cliente es la que se desarrolla en el navegador, es decir, contendría la estructura de la página web desarrollada mediante un lenguaje llamado **HTML (HyperText Markup Language)**, a la que debemos darle un diseño mediante hojas de estilo o **CSS (Cascading Style Sheets)**. Con estas dos herramientas se crearía una página estática, propia de otros tiempos. Si queremos añadirle interactividad y dinamismo necesitaremos el lenguaje de programación por excelencia que es **Javascript**.

En el área de servidor contaremos con los servicios que vayan a ser requeridos por los clientes con sus peticiones. Habitualmente, como mínimo, se tendrá una base de datos y un programa con el que el usuario pueda acceder y tratar la información guardada. Para crear y manipular los datos utilizaremos **SQL (Structured Query Language)** y el lenguaje de programación con el que se establecerá la conexión con el usuario será **PHP (Hypertext Preprocessor).**

Si conseguimos unir todas estas características y potenciarlas seremos capaces de desarrollar una web que responda a todas las expectativas de los usuarios.



Cliente/Servidor

Un claro ejemplo de esta interacción entre cliente y servidor lo podemos encontrar en cualquier página de comercio electrónico. Entramos a ella mediante un navegador, elegimos el artículo que queremos comprar buscando en una base de datos que se encuentra en servidor. Este servidor mediante un programa nos responderá, por ejemplo, si está disponible o no, y la posibilidad de comprarlo disminuyendo así de su stock.

A lo largo de este libro desarrollaremos un proyecto de creación de una página web de e-commerce en el que incluiremos todas las opciones para entender la complejidad y la interrelación entre ellos.

1

PROGRAMACIÓN CLIENTE

1.1 PROGRAMACIÓN CLIENTE

Tal como hemos comentado en la introducción a este libro, el modelo cliente es la programación que se ejecuta en el navegador. HTML y CSS proporcionan lo que es la estructura y el diseño al sitio web, pero añadiendo el lenguaje de programación Javascript conseguimos el dinamismo y la flexibilidad que hará nuestra web mucho más atractiva.

Al cargarse al mismo tiempo que el HTML en el navegador y residir en el cliente, hace que JavaScript sea rápido a la hora de obtener una respuesta el usuario y que pueda seguir funcionando si se produce una desconexión temporal a Internet.

Este curso se centra en la programación del sitio web mediante Javascript, pero en el siguiente apartado daremos una pincelada de HTML y CSS, como base principal.

1.2 PROGRAMACIÓN DE PÁGINAS WEB

Antes de comenzar a estudiar el lenguaje de programación Javascript, haremos una breve reseña de utilización del lenguaje HTML y de las hojas de estilo

CSS, ya que cuando se habla de programación web se suele hacer referencia a todo el conjunto aunque HTML y CSS no sean lenguajes de programación en sí.

Lo primero que tenemos que hacer a la hora de crear una página web es definir su estructura. Para ello utilizaremos HTML. La estructura básica suele tener el formato que se muestra en la ilustración, aunque no es de estricto complimiento.

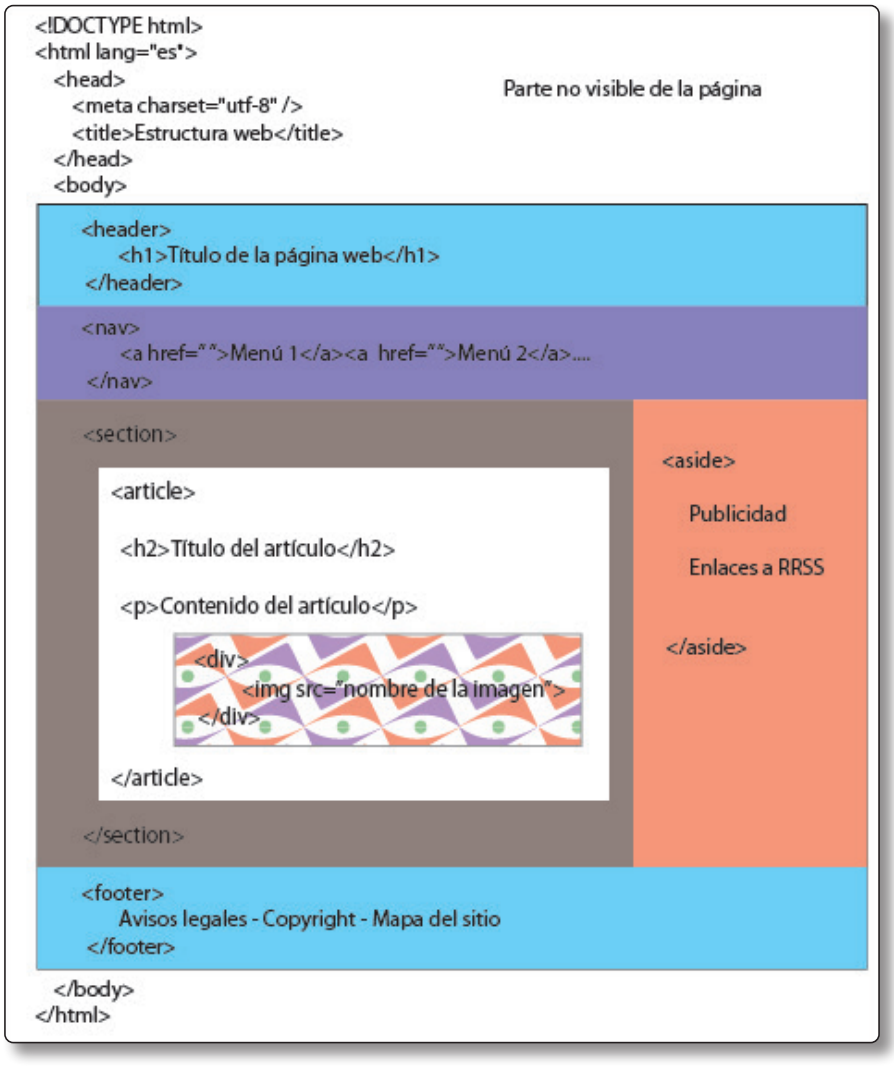


Figura 1.1. Estructura de una página web

Una vez definida la estructura del sitio web, hay que dotarle de estilo a la misma para que sea atractiva para el usuario. Esto se puede realizar de varias formas, de forma directa o embebida, aunque es recomendable incorporar uno o varios archivos de extensión .css y enlazarlos con el HTML en la zona <head>.

```
<head>
<meta charset="utf-8">
<title>Título de la página</title>
<link rel="stylesheet" href="estilos.css">
</head>
```

En este archivo, se va dando formato a cada sección de HTML, indicando si tiene bordes o no, color de fondo, bordes redondeados, color de la letra, sombras y muchas opciones más.

Ejemplo:

Con lo que siguiendo con nuestro ejemplo anterior, contaríamos con dos archivos:

index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Título de la página </title>
<link rel="stylesheet" href="estilos.css">
</head>
<body>
<header>
    <h1>Título principal</h1>
</header>
<nav>
   <ul>
```

```
        <li><a href="">Menu 1</a></li>
        <li><a href="">Menu 2</a></li>
        <li><a href="">Menu 3</a></li>
        <li><a href="">Menu 4</a></li>
    </ul>
</nav>
<section>
 <article>
          <h2>Título del artículo</h2>
          <p>Contenido del artículo ... </p>
            <div>
                <img src="imagen.jpg">
            </div>
        </article>
    </section>
    <aside>
        <p>Publicidad </p>
        <p>Enlaces a RRSS</p>
    </aside>
    <footer>
        Avisos legales - Copyright - Mapa del sitio
    </footer>
</body>
</html>
```

estilos.css

```
header{
    background-color: cornflowerblue;
    height: 50px;
    padding: 5px;
}

ul {
  list-style-type: none;
```

```
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: blueviolet;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111;
}

aside{
    float: right;
    width: 29.9%;
    background-color: burlywood;
    text-align: center;
    min-height: 520px;
    margin=1%;
}
section{
    float: left;
    width: 69.9%;
    background-color: darkkhaki;
    min-height: 520px;
}
```

```
article{
    background-color: aliceblue;
    margin: 2%;
}

article p{
    padding: 1%;
}

img{
    margin: 2%;
    border: 2px solid black;
    border-radius: 5px;
    box-shadow: 10px 10px 10px rgba(0, 0, 0, .5);
    width: 90%;
}

h1, h2{
    text-align: center;
}

footer{
  /*  float: left;
    width: 100%;*/
    background-color: darkgray;
    text-align: center;
   /* min-height: 0px;*/
    padding: 5px;
}
```

que nos mostraría la siguiente página básica:

PROYECTO

Comenzaremos creando un sitio web para la venta de un producto. Se recomienda que no se realice de muchos, sino que se tomen dos o tres productos de ejemplo.

Por ejemplo, puede ser una página de venta de zapatos, ropa o electrodomésticos.

Para este sitio, vamos a crear tres páginas sencillas:

- Índice
- Quienes somos – Contacto
- Productos

Recuerda que la página debe ser sencilla, responsive y se deben agregar enlaces a redes sociales.

1.3 INTRODUCCIÓN A JAVASCRIPT

Podemos decir que Javascript es un lenguaje basado en scripts, es decir, pequeños programas sencillos para conseguir un resultado que le va a proveer a la página las opciones dinámicas que necesita para hacerla más interactiva con el usuario brindándole animaciones, acciones que se activan al pulsar botones o mensajes emergentes de aviso.

JavaScript es un lenguaje interpretado, es decir, no necesita ser compilado y es interpretado por el navegador, con la desventaja que, en algunas ocasiones, podremos obtener resultados diferentes si cambiamos de navegador.

Además es un lenguaje orientado a objetos, que hacen que el código sea más claro e intuitivo. Como su nombre indica, se basa en scripts o guiones que se insertan en el lenguaje HTML. Esto se puede hacer de tres formas, igual que ocurre con css:

- Insertándolo dentro del contenido html. Si se va a introducir de esta forma, es conveniente saber que el navegador lee HTML de forma lineal.

```
…
<h1> Titulo principal </h1>
<script type="text/javascript">
alert("Bienvenido a nuestra página web");
</script>
…
```

- Definiendolo en el head del html

```
…
<head>
<title>Título de la página</title>
<script type="text/javascript">
   alert("Bienvenido a nuestra página web");
</script>
```

```
</head>
…
```

- Creando un fichero externo .js

HTML

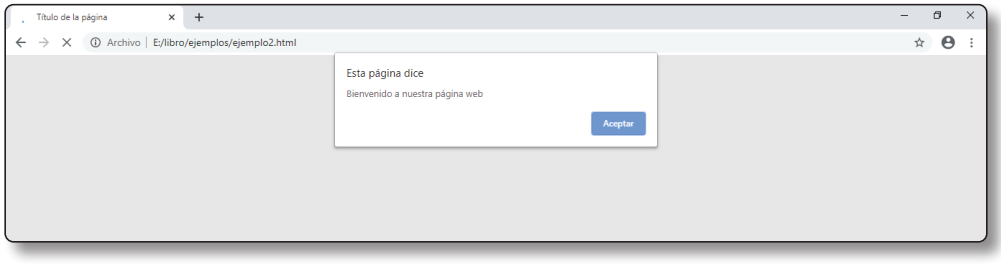
```
...
<head>
<meta charset="utf-8">
<title>Título de la página</title>
<link rel="stylesheet" href="estilos.css">
<script type="text/javascript" src="script.js"></script>
</head>
```

Archivo script.js

```
alert("Bienvenido a nuestra página web");
```

Al igual que en css, se recomienda siempre la utilización de uno o varios archivos externos para la claridad y el mantenimiento del código. Por ejemplo, si necesitamos una calculadora, crearemos un archivo con el nombre calculadora.js donde colocaremos todas las funciones referentes a la misma.

El navegador aceptará el código, sea cual sea la opción anterior elegida, se nos mostrará el siguiente mensaje antes de mostrarnos la página web



Consideraciones previas

Hay algunas consideraciones que tenemos que tener en cuenta antes de comenzar a trabajar con JavaScript:

- JavaScript distingue entre mayúsculas y minúsculas.
- Aunque no es obligatorio, es recomendable finalizar las sentencias en ;
- Como en HTML no se tienen en cuenta los espacios en blanco o las nuevas líneas.
- Los comentarios se colocan mediante // si es un comentario de una línea y entre /* y */ si es un comentario multilínea
- Para evitar errores de código, antes de empezar a escribir el código se coloca las etiquetas de comentario multilínea de HTML <!-- //-->, así si el navegador no entiende la etiqueta <script>, la ignora siguiendo en código HTML, con lo que todo el script quedará envuelto en un comentario.

```
<script>
<!--
   Código de Javascript
//-->
</script>
```

- Utilizamos la etiqueta <noscript> cuando el navegador no puede ejecutar JavaScript

```
<noscript> Para ver correctamente esta página web,
necesita un navegador que ejecute Javascript
</noscript>
```

1.4 FUNDAMENTOS DE PROGRAMACIÓN

Tenemos dos formas de realizar un programa, utilizando la programación estructurada o bien utilizando la programación orientada a objetos. Javascript es un lenguaje que puede utilizar cualquiera de las dos metodologías.

La programación estructurada, tal como su nombre indica, es una estructura que se va ejecutando paso a paso de forma secuencial.

La programación orientada a objetos, se basa precisamente en esto, en crear objetos que se tratan como un todo e interactuar entre ellos.

1.4.1 Variables

Una variable es un espacio que se guarda en memoria y va cambiando de contenido según las necesidades del programa. En el caso de Javascript no es obligatorio declararlas, aunque si es aconsejable.

Además, JavaScript es un ***lenguaje no tipado***, es decir, una vez declarada la variable se puede almacenar en ella cualquier tipo de datos, un número, una letra, un string..es decir asigna el tipo por inferencia o deducción.

Una variable se declara mediante la instrucción **var**

```
var numero;
```

y también se puede iniciar o inicializar la variable con un determinado valor

```
var numero=1;
```

si no se asigna valor a la variable, esta tendrá como contenido **undefined.**

Se pueden declarar múltiples variables en una solo línea escribiendo var y los nombres separados por coma.

```
var numero1, numero2
```

Aunque se le puede dar el nombre que queramos, con unas determinadas especificaciones:

- Sólo puede contener caracteres alfanuméricos (a-z, A-Z, 0-9), y los símbolos: _, $.
- No puede comenzar por número.
- No puede ser una de las palabras reservadas utilizadas por el lenguaje (var, if, for, etc.).

Es aconsejable que se nombre dependiendo del uso que le vayamos a dar, para luego tener una referencia a la hora de utilizarla, por ejemplo:

```
var nombreCliente = "Juan Pérez";
```

1.4.1.1 AMBITO DE LAS VARIABLES

El ámbito de las variables es el lugar donde estarán disponibles. Por lo general, está disponible en el lugar donde se ha declarado, por ejemplo, las variables declaradas dentro de una página web estarán accesibles dentro de ella.

- **Variables globales:** las variables globales son las que están declaradas en el ámbito más amplio, es decir las declararemos en la parte superior del script. Son accesibles desde cualquier lugar, incluidas funciones, manejadores de eventos, etc.
- **Variables locales:** las variables locales son aquellas que se declaran en lugares más acotados como una función y sólo se tendrán acceso a ellas dentro del lugar donde se han declarado.

Aunque es posible, no es aconsejable declarar variables locales y globales con el mismo nombre, ya que puede crear confusión sobre qué variable se está utilizando en un momento determinado.

1.4.1.2 TIPOS DE DATOS

JavaScript interpretará que tipo de dato se está utilizando en el momento en que se le asigna el valor y realizará la conversión automática cuando cambiemos de tipo de dato. Los tipos básicos de datos son:

- **Entero/decimal:** permite cualquier número que sea necesario. Para indicar que es negativo basta con poner el signo menos adelante y el . se utiliza como separador decimal, pudiéndose omitir el 0 si el número no tiene parte entera, es decir, se puede utilizar .34 como número válido. También admite los número en notación científica, con E o e de manera indistinta.

- **Carácter/cadena de caracteres o strings:** a diferencia de otros lenguajes javascript utiliza el valor string tanto para un carácter como para una cadena de texto. Para definirlo se pueden utilizar tanto comillas simples como comillas dobles, aunque es preferible utilizar las comillas dobles para HTML y dobles para Javascript para que el navegador determine cuales son las de apertura y cierre.

- **Booleanos**: puede contener valores verdadero o falso únicamente y se suele utilizar para determinar si se cumple o no una condición.

Ejemplo:

```
var numero1=1;
//declaramos un número entero llamado numero1
var numero2=2.5;
//declaramos numero2 de tipo decimal o real
var numero3=numero1+numero2;
//numero3 valdrá número1+numero2, es decir 3.5
var texto = "Hola";
```

```
//declaramos un string (cadena) de valor Hola
var mezcla = texto + numero1; //mezcla dará por resultado Hola1
var cobrado=verdadero;
//cobrado es de tipo booleano (verdadero o falso)
```

También hay que tener en cuenta que si se quiere añadir caracteres especiales a un texto se debe agregar el carácter \ dentro de los apostrofes del string:

Secuencia	Resultado	Secuencia	Resultado	Secuencia	Resultado
\\	\	\"	"	\"	"
\n	Salto de línea	\t	tabulador	\b	Retorno de carro

Ejemplo:

```
var cadena="Esta cadena lleva el carácter \\ seguido de tres
saltos de línea \n \n\n y prosigue comilla simple \" y doble
\"\n"
```

Javascript también permite utilizar variables sin que se hayan declarado previamente. Cuando encuentre una variable no declarada, la crea y permite su utilización, lo que facilita la programación.

1.4.1.3 ARRAYS

Javascript como cualquier lenguaje de programación, también trabaja con arrays, que son variables que, mediante un índice, permiten organizar datos bajo un mismo nombre. Pero a diferencia de otros lenguajes, en Javascript cada uno de los valores del array puede ser de distinto tipo.

Como cualquier variable, el array se declara mediante la opción var, aunque para que javascript reconozca que efectivamente es un array hay que indicárselo:

```
var array = [];
```

y a continuación añadir los valores:

```
array[0]=valor0;
```

o bien agregando los valores a la hora de la declaración:

```
var array = [valor0, valor1, valor2, … ]
```

Así mismo, para acceder a cada elemento del array se utiliza el índice de la posición, es decir, el valor de array[0] será valor0. Si se intenta acceder a un índice de un array inexistente, el resultado será **undefined**. En Javascript el número de elementos del array se ajusta dinámicamente.

1.4.2 Operadores

Dependiendo del tipo de datos que estemos utilizando, se necesitan diferentes operadores:

- ***Matemáticos*** (+, - , *, / , %) realizan las operaciones matemáticas entre números, % devuelve el resto de la división, por lo que es un número entero.

```
var valor1 = 1;
var valor2 = 2;
var suma=valor1 + valor2;               //suma valdrá 3
var resta = valor2-valor1;              //resta valdrá 1
var multiplicacion =  valor1 * valor2;
//multiplicacion valdrá 2
var division = valor2/valor1;
//división valdrá 2
var resto =valor2%valor1;               //resto será 0
```

- ***Asignación*** (=, +=, -=, *=, /=, %=) para darle valor a una variable. En el caso de utilizarlo junto a un operador matemático, primero hará la operación y a continuación le asignará el valor a la variable.

```
var valor1 = 1;
valor1 += 2;            //valor1 pasará a valer 3
```

- ***Concatenación*** (+) se utiliza en cadenas de caracteres para unir textos.

```
var valor1 = "Nombre";
var valor2 = "Apellidos";
Nombre = valor1 + " " + valor2;
//Nombre será "Nombre Apellidos"
```

- ***Comparación*** (== , !=, <, >, <=; >=) se utilizan para comparar dos variables. Hay que tener cuidado en confundir el operador de asignación (=) con el igual (==). Devuelven un valor booleano, siendo true si la condición se cumple y false en caso contrario, siendo muy útil para opciones condicionales

```
var valor1 = 1;
var valor2 = 2;
if (valor1 >= valor2)   //devuelve false ya que 1<2
if (valor1 == valor2)   //devuelve false
if (valor1=valor2)
//devuelve true ya que el valor1 pasará a valer 2
```

- ***Lógicos*** (&& (y), || (o), ! (no)) se basan en las tablas de verdad, y es verdadero cuando se cumplen todas las condiciones, o es verdadero cuando se cumple una de las condiciones enumeradas, y no devuelve el valor contrario, es decir, verdadero si la condición es falsa.

```
var valor1 = true;
var valor2 = false;
if (valor1 && valor2)
//devuelve false ya que no cumple las dos
```

```
if (valor1 || valor2)
//devuelve true ya que una de las dos es verdadera
if (!valor1)
//devuelve false por ser el valor contrario
```

- ***Incremento/Decremento:*** el incremento/decremento permite agregar/restar una unidad a una determinada variable. Dependiendo de la posición en donde se coloque el operador, se incrementará/decrementará la variable antes o después de realizar otra operación (n++, --n)

```
var valor1 = 3;
valor1++;                    //valor1 pasa a valer 4
```

Si intentamos realizar una operación matemática con variables no numéricas, el resultado será **NaN** (Not a Number), lo que evitará que el programa deje de ejecutarse.

```
var valor1, valor2;
valor1 = 3;
resultado = valor1+valor2;
//resultado será NaN ya que valor2 no tiene valor
```

1.4.3 Interacción con el usuario

Existen tres formas básicas de interactuar con el usuario:

- ***alert***: envía un mensaje de información. Incluye un botón Aceptar.

```
alert("Mensaje informativo");
```

- ***confirm***: se utiliza para confirmar una tarea. Incluye dos botones: Aceptar y Cancelar, por lo que se obtiene un valor booleano (verdadero o falso) para realizar o no la acción determinada.

```
valor = confirm("Mensaje con acción a confirmar");
```

- ***prompt:*** solicita información al usuario.

```
valor = prompt("Mensaje con la información a solicitar",
[valor por defecto](opcional));
```

1.4.4 Conversión de tipos

Si queremos saber qué tipo de datos es una variable, utilizaremos la opción typeof, por ejemplo

```
var numero = 1;
//declara una variable número y le asigna el valor 1
alert(typeof numero);
//muestra un mensaje con la opción "number"
```

Hemos visto que podemos concatenar un string con un número y JavaScript automáticamente lo convierte a string.

Ejemplo:

```
var mezcla = "Hola " + 21;  //mezcla tiene el valor "Hola 21"
```

El problema surge cuando solicitamos un valor, y a continuación queremos realizar una suma de valores, al tomarlo como cadena de caracteres lo que hará es una concatenación en lugar de una suma.

Ejemplo:

```
var numero1 = prompt("Introduzca el primero número", 0);
//pide el primer número
var numero2 = prompt("Introduzca el segundo número", 0);
//pide el segundo número
var numero3 = numero1 + numero2;
//En lugar de sumar, concatena los valores
alert("El resultado es " + numero3);
//Si numero1=0 y numero2=0, numero3 = 00
```

Este error sólo ocurre cuando utilizamos el operador +, ya que se puede utilizar tanto para sumar número como para concatenar cadenas. En este caso, debemos convertir los valores de cadenas a número, para ello utilizamos dos instrucciones:

- ***parseInt***: convierte una cadena en número entero.
- ***parseFloat***: convierte una cadena en número decimal.

Entonces, siguiendo con el ejemplo anterior:

```
var numero1 = prompt("Introduzca el primero número", 0);
//pide el primer número
var numero2 = prompt("Introduzca el segundo número", 0);
//pide el segundo número
var numero3 = parseInt(numero1) + parseInt(numero2);
//Con conversión de valores
alert("El resultado es " + numero3);
//Si numero1=0 y numero2=0, numero3 = 0
```

1.4.5 Sentencia condicional

1.4.5.1 SENTENCIA CONDICIONAL IF-ELSE

Una sentencia condicional permite elegir realizar una acción dependiendo de las circunstancias que ocurran. Se suelen utilizar en la vida cotidiana, como por ejemplo, si llueve => llevar paraguas.

En JavaScript, las sentencias condicionales se utilizan mediante la sentencia if

```
if (condición) acción;          //if (llueve) llevar paraguas
```

También podemos considerar realizar una acción diferente si la condición no se cumple, es decir si llueve => llevar paraguas sino => llevar gafas de sol. En este caso, la opción sino se traduce en JavaScript como else

```
if (condición) acción;          //if (llueve) llevar paraguas
else  acción;                   //else llevar gafas de sol
```

La cláusula else es no obligatoria e incluso puede dejarse vacía.

Cuando se realizaran varias acciones dentro del if, deberemos encuadrarlas dentro de {} para que se consideren un conjunto, por tanto se recomienda siempre utilizar las llaves, aunque sea una única acción, para evitar errores y tener un código más claro. Volviendo al ejemplo:

```
if (condición) {
   Accion1;
   Acción2;
}
else {
   Acción3;
}
```

Además, podemos querer combinar más de una condición, con lo cual podemos anidar los if. En nuestro ejemplo: Si llueve => llevar paraguas, sino => si hace calor => ir a la playa => ir a esquiar

```
if (condición1) acción;        //if (llueve) llevar paraguas
else                           //sino
   if(condición2) acción;      //si (hace calor ) ir a la playa;
   else acción;                //sino ir a esquiar
```

La condición es una expresión booleana, y se pueden utilizar los operadores de comparación y lógicos vistos anteriormente.

Ejemplo:

Consideremos la evaluación de unos alumnos, si su nota es inferior a 5, será "Suspenso", si es inferior a 7 tendrá "bien", si es inferior a 9 tendrá "Notable", sino será "Sobresaliente". También tenemos que tener en cuenta que la nota no puede ser menor que 0 ni mayor de 10. Traduciéndola a código:

```
var nota = prompt("Escribe la nota del alumno", 0);
var calif;
if (nota<0 || nota>10)
    alert("La nota debe estar entre 0 y 10");
else
    if (nota<5)
        calif="suspenso";
    else
        if (nota<7)
            calif="bien";
        else
            if (nota<9)
                calif="notable";
            else
                calif="sobresaliente";
alert("La calificación del alumno es " + calif);
```

1.4.5.2 SENTENCIA SWITCH

Cuando tenemos varias condiciones podemos utilizar la sentencia switch en lugar de utilizar ifs encadenados, con la única diferencia que solo permite evaluar valores concretos, no intervalos. Esta instrucción realiza diferentes acciones dependiendo de la variable o expresión. La forma de utilización es:

```
Switch(variable o expresión){
case valor1: acciones; break;
case valor2: acciones; break;
…
default: acciones; break
}
```

La instrucción **break** se utiliza para salir/saltar la opción switch, terminando la evaluación y continuando con el resto del código. Esta instrucción, aunque opcional es recomendable.

La instrucción **default** representa las acciones que se ejecutaran sino se han cumplido ninguna de las opciones anteriores. También es opcional.

Ejemplo:

```
switch(new Date().getDay()){
//Date().getDay() devuelve el dia de la semana en número
case 0:
        dia="Domingo";
        break;
case 1:
        dia="Lunes";
        break;
case 2:
        dia="Martes";
        break;
case 3:
```

```
        dia="Miércoles";
        break;
case 4:
        dia="Jueves";
        break;
case 5:
        dia="Viernes";
        break;
case 6:
        dia="Sábado";
        break;
}
```

1.4.6 Bucles: while / do while / for

Un bucle es una estructura repetitiva que realiza la iteración mientras o hasta que cumple una condición o simplemente se repiten un número determinado de veces. Los bucles se pueden anidar dentro de otro al igual que las sentencias condicionales.

En JavaScript tenemos tres tipos de instrucciones repetitivas:

1.4.6.1 WHILE

While significa mientras, por tanto, el bucle se repetirá mientras se cumpla la condición.

```
while (condición){
instrucciones;
}
```

Condición es una expresión que da un resultado true o false, que hará que se ejecuten o no las instrucciones.