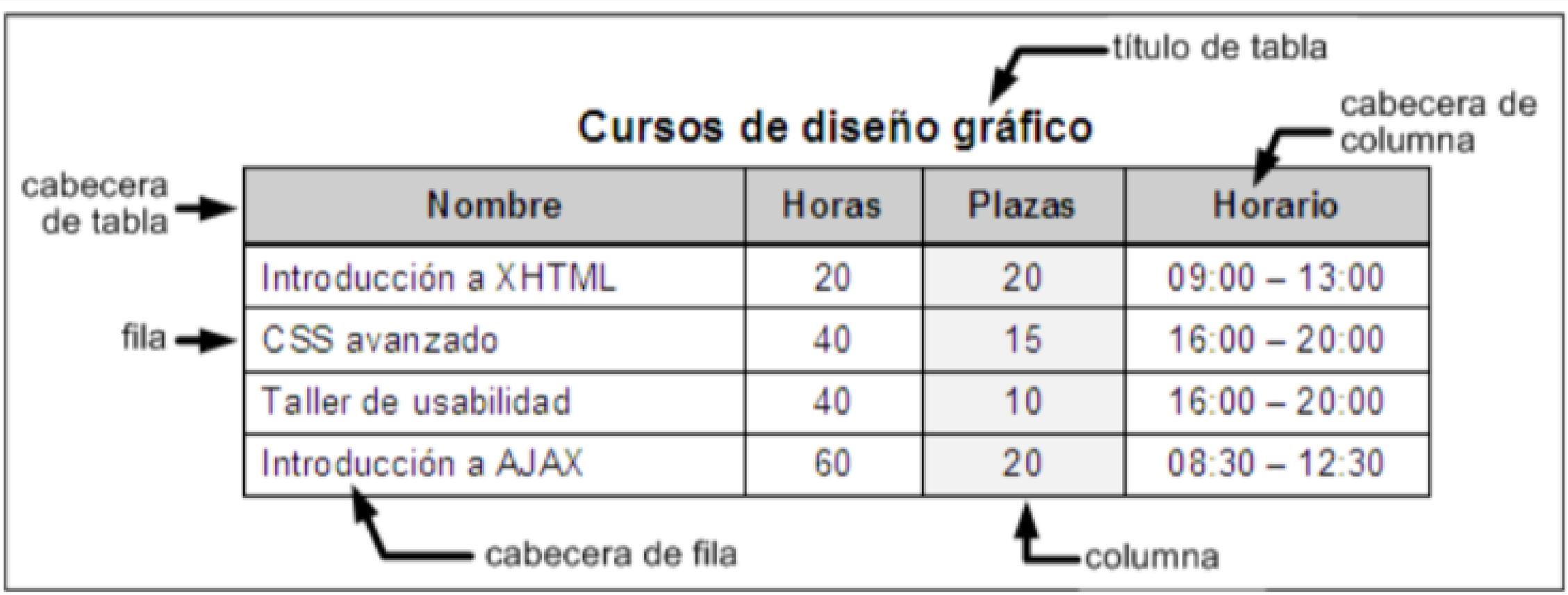
CLASE #2

BOOTCAMP BICTIA

TABLAS



Nombre	Horas	Plazas	Horario
Introducción a XHTML	20	20	09:00 – 13:00
CSS avanzado	40	15	16:00 – 20:00
Taller de usabilidad	40	10	16:00 – 20:00
Introducción a AJAX	60	20	08:30 – 12:30

EJERCICIO

Confeccionar una tabla que muestre en la ***primer columna*** los nombre de distintos empleados de una compañía y en la ***segunda*** el sueldo bruto (la compañía tiene 4 empleados)

COLSPAN

Indica el número de columnas que ocupará la celda.

Por defecto ocupa una sola columna.

ROWSPAN

Indica el número de filas que ocupará la celda.

Por defecto ocupa una sola fila.

FORMULARIOS

FORMS

- Un **form o formulario,** permite a los usuarios de un sitio web que ingresen datos.
- Estos datos son enviados al servidor y procesados por el **lenguaje de programación** (*PHP, .NET, Java, etc*).
- Es el medio ideal para registrar: comentarios de los visitantes, realizar un pedido, sacar un turno, pedir información, etc.
- Es la forma que tiene el **usuario** de comunicarse con el sitio web.

FORMULARIOS

ETIQUETAS

- Para crear un formulario, debemos utilizar el elemento

  <form> </form>

- Dentro de la etiqueta principal del formulario, van otras etiquetas o elementos que hacen referencia a: botones, campos de texto, casillas de verificación, opciones desplegables, etc

FORMULARIOS

```
1   <html>
2   <head>
3     <meta charset="utf-8">
4     <title>Formularios</title>
5   </head>
6   <body>
7
8     <!-- ejemplo de formulario GET -->
9     <form action="http://www.mi-sitio-web.com/script.php" method="GET">
10
11    </form>
12
13    <!-- ejemplo de formulario POST -->
14    <form action="script.php" method="POST">
15
16    </form>
17
18  </body>
19  </html>
20
```

FORMULARIOS

ATRIBUTOS

- ***action*** = define la ubicación (una URL) donde se enviará la información recolectada por el formulario.

- ***method*** = define con qué método HTTP se enviará la información (este puede ser "**get**" o "**post**").

ESTILOS

- Las siglas CSS (**Cascading Style Sheets)** significan «Hojas de estilo en cascada».
- Mediante **CSS** podemos aplicar **estilos** (*colores, formas, márgenes, etc...*) a uno o varios documentos (generalmente documentos HTML, páginas webs) de forma masiva.
- Se le denomina **estilos en cascada** porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia que trataremos más adelante).

ESTILOS

- El **CSS** permite *separar* **presentación** y **contenido**, intentando que los documentos **HTML** incluyan sólo información y datos (contenido), y todos los aspectos relacionados con el estilo (diseño, colores, formas) se encuentren en un documento **CSS** independiente (presentación).
- Ventajas:
  a. Si necesitamos modificar lo hacemos en un sólo lugar.
  b. Evitar duplicación de estilos en diferentes lugares.
  c. Se pueden crear versiones diferentes de estilos para distintos dispositivos.

CSS - TIPOS

CSS EXTERNO

En la cabecera del **HTML**, el bloque ***<head></head>***, incluimos una etiqueta que "llama" al archivo CSS en cuestión:

```
<link rel="stylesheet" type="text/css" href="index.css" />
```

De esta forma, los navegadores sabrán que deben aplicar los estilos de este archivo (***index.css***) al documento HTML actual.

NOTA: Se aconseja escribir esta línea lo antes posible, obligando así al navegador a aplicar los estilos cuanto antes y eliminar la falsa sensación de página no cargada por completo.

CSS - TIPOS

CSS INTERNO

Otra de las formas que existen para incluir estilos en un documento **HTML** es la de añadirlos directamente en la cabecera HTML del documento:

```
<!DOCTYPE html>
<html>
<head>
    <title>Título de la página</title>
    <style type="text/css">
        div {
            background:#FFFFFF;
        }
    </style>
</head>
...
```

NOTA: Este sistema puede servir en algunos casos, pero hay que tener en cuenta que utilizándolo, arruinamos la ventaja de tener los estilos en un documento independiente, por lo que siempre es preferible guardarlo en un archivo externo CSS.

CSS - TIPOS

CSS EMBEBIDO

Otra forma de aplicar estilos en un documento **HTML** es hacerlo directamente en las propias etiquetas, a través del atributo ***style***:

```
<p>¡Hola <span style="color:#FF0000">amigo lector</span>!</p>
```

NOTA: Al igual que en el método anterior, se recomienda no utilizarse salvo casos muy específicos, ya que se pierde la independencia de la presentación y contenido.
Sin embargo, es una opción que puede venir bien en algunos casos.

CSS - ESTRUCTURA

La estructura **CSS**, se basa en ***reglas***, que tienen el siguiente formato:

- **Selector**: es el elemento **HTML** que vamos a seleccionar del documento para aplicarle un estilo concreto. Ejemplo, el elemento ***p***., o un ***ID***, o una ***clase***
- **Propiedad**: es una de las diferentes características que brinda el lenguaje CSS. Ejemplo: *color, ancho, posición, fuente, etc.*
- **Valor**: cada propiedad **CSS** tiene una serie de valores concretos, con los que tendrá uno u otro comportamiento.

CSS - ESTRUCTURA

Código **HTML** →

```html
<!DOCTYPE html>
<html>
<head>
    <title>Título de página</title>
    <link rel="stylesheet" type="text/css" href="index.css" />
</head>
<body>
    <div id="first">
        <p>Párrafo</p>
    </div>
    <div id="second">
        <span>Capa</span>
    </div>
</body>
</html>
```

Código **CSS** del archivo ***index.css***

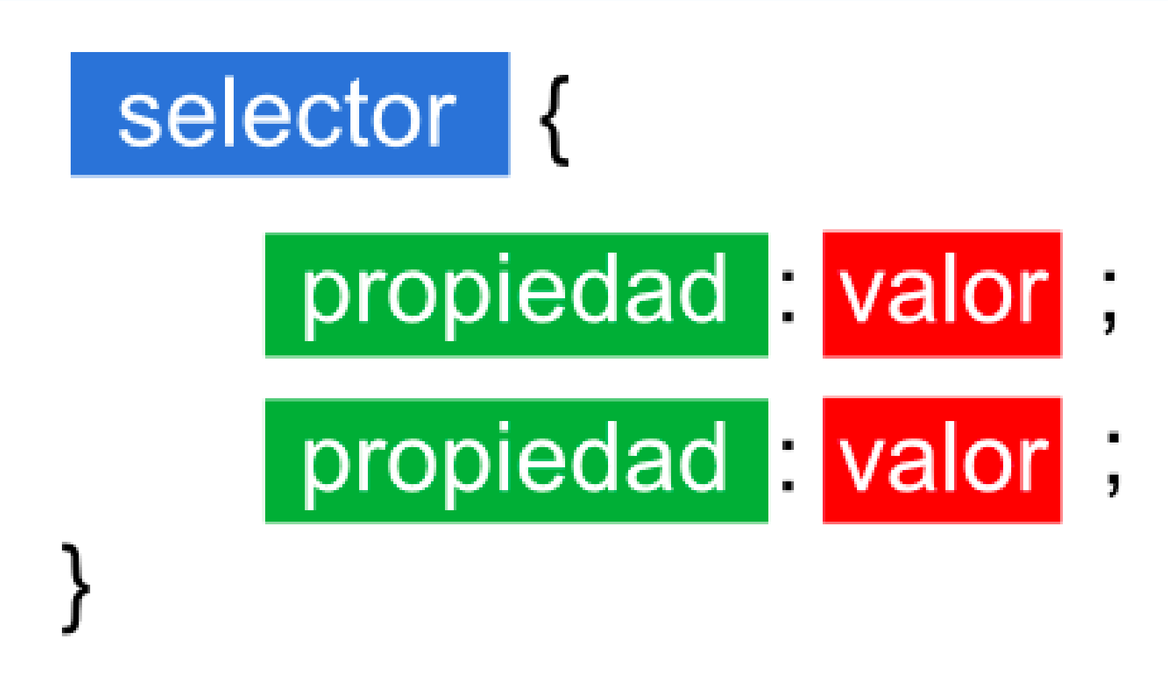
↓

```css
p {
    color:red;      /* Color de texto rojo */
}
```

NOTA: De esta forma, a todas las etiquetas ***<p>*** se le aplicará el estilo especificado: **el color rojo.**

CSS - ESTRUCTURA

Para aplicar muchas *reglas*, se utiliza el siguiente formato:



CSS - SELECTORES

Los **selectores** se utilizan para ***encontrar*** o ***seleccionar***, el/los elementos a los que queremos aplicarle estilos

Selector	Ejemplo	Descripción del ejemplo
.class	.intro	Selecciona todos los elementos con clase class="intro"
#id	#nombre	Selecciona el elemento con id="firstname"
*	*	Selecciona todos los elementos de una página
elemento	p	Selecciona todos los elementos <p>
elemento,elemento,..	div, p	Selecciona todos los elementos y todos los elementos <p>

CSS - FLEXBOX

FLEXBOX

Flexbox es un sistema de disposición de elementos flexibles en donde los elementos se adaptan y colocan automáticamente y es más fácil personalizar los diseños y establecer los parámetros para la visualización de los mismos en responsive.

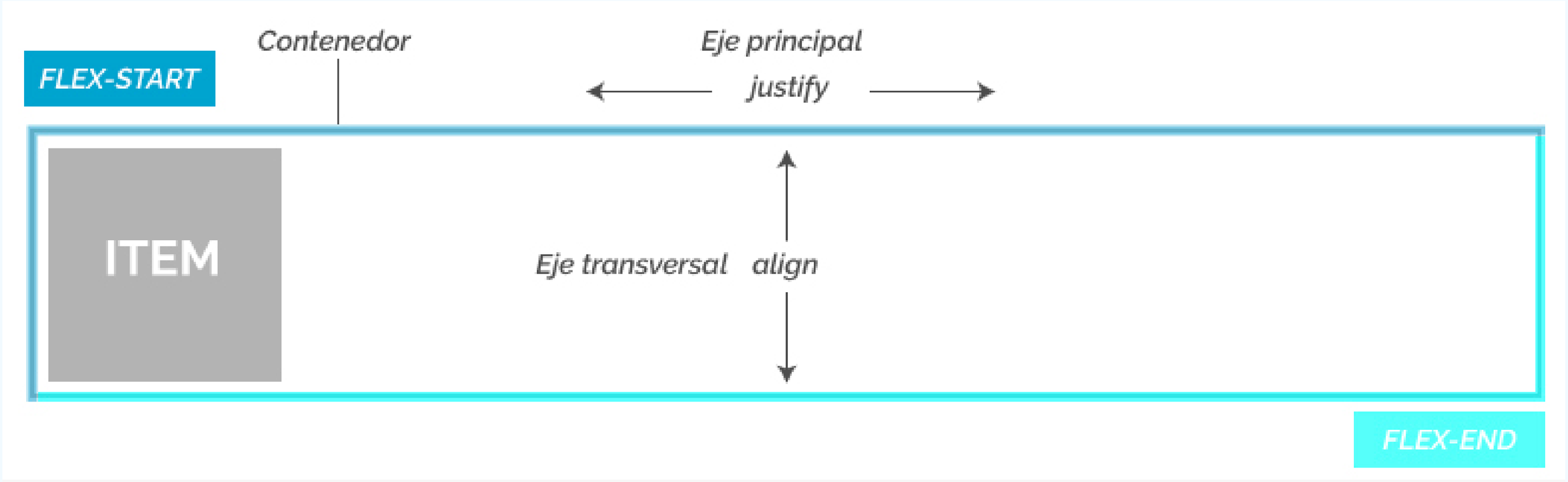
Lo primero que debes saber son los elementos a tomar en cuenta para aplicar este método:

CSS - FLEXBOX

FLEXBOX

- **Contenedor**: El elemento padre que es el contenedor que tendrá en su interior los items flexibles y adaptables.
- **Ítem**: Son los hijos flexibles que se encontraran dentro del contenedor.
- **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, es en horizontal (fila).
- **Eje transversal:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical, y viceversa.

FLEXBOX


Contenedor
Eje principal
justify
FLEX-START
ITEM
Eje transversal
align
FLEX-END

CSS - FLEXBOX

EJEMPLO

```
<div class="container">
      <div class="flex-child" style="background-color: #b2ebf2;">1</div>
      <div class="flex-child" style="background-color: #80deea;">2</div>
      <div class="flex-child" style="background-color: #4dd0e1;">3</div>
      <div class="flex-child" style="background-color: #26c6da;">4</div>
      <div class="flex-child" style="background-color: #00bcd4;">5</div>
      <div class="flex-child" style="background-color: #00acc1;">6</div>
      <div class="flex-child" style="background-color: #0097a7;">7</div>
      <div class="flex-child" style="background-color: #00838f;">8</div>
</div>
```

CSS - FLEXBOX

EJEMPLO

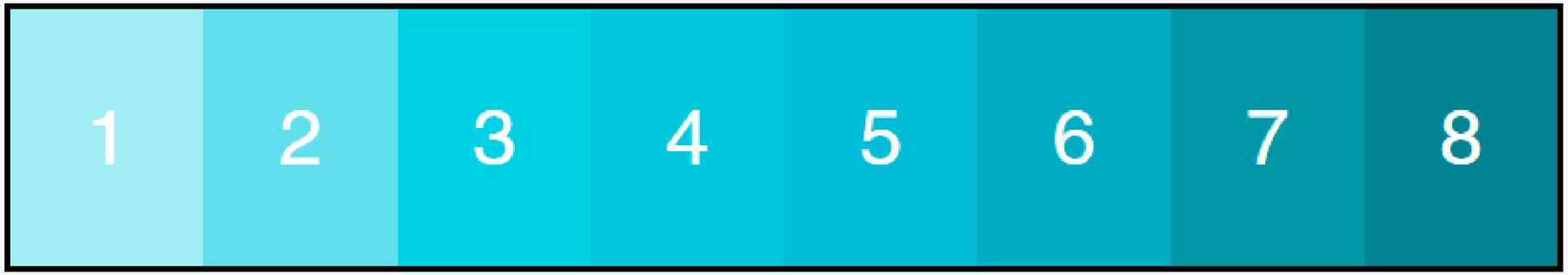
```
.container{
    display: flex;
    max-width: 800px;
    margin: 0 auto;
    border: 2px solid #000000;
}

.flex-child {
    font-size: 2.5rem;
    text-align: center;
    line-height: 132px;
    color: #FFFFFF;
    width: 100%;
}
```

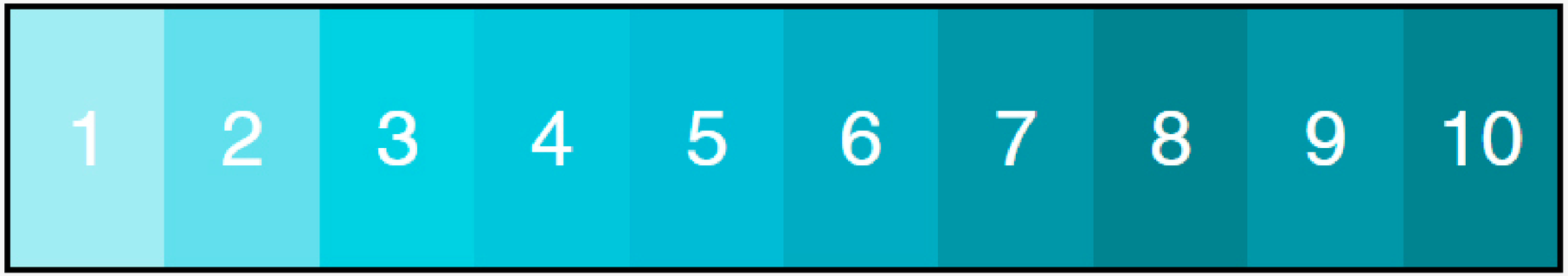
CSS - FLEXBOX

EJEMPLO

Resultado:



Si añadimos 2 elementos mas se vería de esta manera:



Es decir los elementos se ajustan automáticamente al ancho del contenedor.

CSS - FLEXBOX

DIRECCIÓN DE LOS EJES

Para cambiar la disposición de los elementos y comportamiento de los items a lo largo del eje principal del contenedor utilizamos la propiedad **FLEX-DIRECTION**:

- **Flex-direction: row**
- **Flex-direction: row-reverse**
- **Flex-direction: column**
- **Flex-direction: column-reverse**

CSS - FLEXBOX

DIRECCIÓN DE LOS EJES

Flex-direction: row;

Esta es la disposición de los elementos por defecto.

Flex-direction: row-reverse;

Invierte el orden de los elementos.

flex-direction: column;

Dispone los elementos uno debajo del otro.

flex-direction: column-reverse

Dispone los elementos uno debajo del otro invirtiendo el orden.

JAVASCRIPT

INTRO

- **JavaScript** es un lenguaje de programación que se utiliza principalmente para crear *páginas web dinámicas*.
- Una **página web dinámica** es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.
- **JavaScript** es un ***lenguaje de programación interpretado***, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de instalar nada.

JAVASCRIPT - CÓMO INCLUIR EN HTML

EN EL MISMO DOCUMENTO HTML

- El código **JavaScript** se encierra entre etiquetas ***<script>*** y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código **JavaScript** dentro de la cabecera del documento (dentro de la etiqueta ***<head>***):

JAVASCRIPT - CÓMO INCLUIR EN HTML

EN EL MISMO DOCUMENTO HTML

Para que la **página HTML** resultante sea válida, es necesario añadir el atributo **type** a la etiqueta **<script>**, con el valor **text/javascript**.

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

JAVASCRIPT - CÓMO INCLUIR EN HTML

EN UN ARCHIVO EXTERNO

- Las instrucciones **JavaScript** se pueden incluir en un archivo externo de tipo **JavaScript** que vamos a enlazar mediante la etiqueta ***<script>***.

Además del atributo ***type***, se debe definir el atributo ***src***, que es el que indica la **URL** correspondiente al archivo JavaScript que se quiere enlazar.

Cada etiqueta ***<script>*** solamente puede enlazar **un único archivo**, pero en una misma página se pueden incluir tantas etiquetas **<script>** como sean necesarias.

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

JAVASCRIPT - CÓMO INCLUIR EN HTML

INCLUIR JAVASCRIPT EN LOS ELEMENTOS

El mayor inconveniente de este método es que ensucia innecesariamente el código **HTML** de la página y complica el mantenimiento del código **JavaScript**.

- Este método es el menos utilizado, ya que consiste en incluir trozos de **JavaScript** dentro del código **HTML** de la página:

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de código JavaScript en el propio documento</title>
</head>

<body>
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

JAVASCRIPT - SINTAXIS

La **sintaxis** de un lenguaje de programación son el conjunto de ***reglas*** que deben seguirse al escribir un **código fuente**, para considerarse como correctos para ese **lenguaje de programación**.

- No se tienen en cuenta los **espacios en blanco** y las **nuevas líneas**
- Se distinguen las **mayúsculas** y **minúsculas**
- No se define el tipo de las **variables**
- Se pueden incluir **comentarios**
- Las sentencias o líneas de código, finalizan con **punto y coma** (;)

```
// a continuación se muestra un mensaje
alert("mensaje de prueba");
```

```
/* Los comentarios de varias líneas son muy útiles
cuando se necesita incluir bastante información
en los comentarios */
alert("mensaje de prueba");
```

JAVASCRIPT - MI PRIMER SCRIPT

```
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>El primer script</title>

<script type="text/javascript">
  alert("Hola Mundo!");
</script>
</head>

<body>
<p>Esta página contiene el primer script</p>
</body>
</html>
```

La instrucción ***alert()*** es una de las utilidades que incluye **JavaScript** y permite mostrar un mensaje en la pantalla del usuario.

Si se visualiza la página web de este primer script en cualquier navegador, automáticamente se mostrará una ventana con el mensaje "**Hola Mundo!**".

JAVASCRIPT - EJERCICIO

Modificar el script anterior para que:

- Todo el código **JavaScript** se encuentre en un *archivo externo* llamado ***codigo.js*** y el script siga funcionando de la misma manera.
- Después del primer mensaje, se debe mostrar otro mensaje que diga "**Soy el primer script**"
- Añadir algunos comentarios que expliquen el funcionamiento del código

JAVASCRIPT - VARIABLES

```
var numero_1 = 3;
var numero_2 = 1;
var resultado = numero_1 + numero_2;
```

```
var numero_1;
var numero_2;

numero_1 = 3;
numero_2 = 1;

var resultado = numero_1 + numero_2;
```

CORRECTO

```
var $numero1;
var _$letra;
var $$$otroNumero;
var $_a__$4;
```

Los nombres de las variables sólo pueden estar formadas por **letras**, **números** y los símbolos **$** (*dólar*) y **_** (*guión bajo*).

El primer carácter no puede ser un número.

INCORRECTO

```
var 1numero;        // Empieza por un número
var numero;1_123;   // Contiene un carácter ";"
```

JAVASCRIPT - TIPO DE VARIABLES

Numéricas

- Se utilizan para almacenar valores numéricos *enteros* (llamados ***integer*** en inglés) o *decimales* (llamados ***float*** en inglés).
- En este caso, el valor se asigna indicando directamente el *número entero o decimal*.
- Los números decimales utilizan el carácter . (punto) en vez de , (coma) para separar la parte entera y la parte decimal

```
var iva = 16;          // variable tipo entero
var total = 234.65;    // variable tipo decimal
```

JAVASCRIPT - TIPO DE VARIABLES

Cadenas de texto (*string*)

- Se utilizan para almacenar ***caracteres***, ***palabras*** y/o ***frases de texto***.
- Para asignar el valor a la variable, se encierra el valor entre ***comillas dobles*** o ***simples***, para delimitar su comienzo y su final.

```
var mensaje = "Bienvenido a nuestro sitio web";
var nombreProducto = 'Producto ABC';
var letraSeleccionada = 'c';
```

```
/* El contenido de texto1 tiene comillas simples, por lo que
    se encierra con comillas dobles */
var texto1 = "Una frase con 'comillas simples' dentro";

/* El contenido de texto2 tiene comillas dobles, por lo que
    se encierra con comillas simples */
var texto2 = 'Una frase con "comillas dobles" dentro';
```

JAVASCRIPT - EJERCICIO

Modificar el script anterior para que:

- El mensaje que se muestra al usuario se almacene en una *variable* llamada ***mensaje*** y el funcionamiento del script sea el mismo.

JAVASCRIPT - TIPO DE VARIABLES

Arrays

- En ocasiones, a los **arrays** se les llama ***vectores***, ***matrices*** e incluso ***arreglos***.
- Un ***array*** es una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.

```
var dia1 = "Lunes";
var dia2 = "Martes";
...
var dia7 = "Domingo";
```

JAVASCRIPT - TIPO DE VARIABLES

Arrays

- En este tipo de casos, se pueden agrupar todas las variables relacionadas en una colección de variables o **array**. El ejemplo anterior se puede rehacer de la siguiente forma:

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];
```

- Ahora, una única variable llamada ***dias*** almacena todos los valores relacionados entre sí.
- Para definir un array, se utilizan los caracteres **[** y **]** para delimitar su comienzo y su final, y se utiliza el carácter **,** *(coma)* para separar sus elementos

JAVASCRIPT - TIPO DE VARIABLES

Arrays

- Para acceder a un elemento del array, se debe indicar su posición dentro del **array**.
- La única complicación, que es responsable de muchos errores cuando se empieza a programar, es que las posiciones de los elementos empiezan a contarse en el **0** y no en el **1**

```
var diaSeleccionado = dias[0];      // diaSeleccionado = "Lunes"
var otroDia = dias[5];              // otroDia = "Sábado"
```

JAVASCRIPT - TIPO DE VARIABLES

Booleanos

- Las variables de tipo **boolean** o **booleano** también se conocen con el nombre de variables de tipo lógico.
- Una variable de tipo boolean almacena un tipo especial de valor que solamente puede tomar dos valores: ***true*** (*verdadero*) o ***false*** (*falso*).

```
var clienteRegistrado = false;
var ivaIncluido = true;
```

JAVASCRIPT

TRABAJO INTEGRADOR

Crear una página web que permita gestionar nuestros gastos e ingresos.

- ABM de gastos (alta, baja y modificación)
- Listados de gastos
- ABM de ingresos (alta, baja y modificación)
- Listado de ingresos
- Registro de usuarios

JAVASCRIPT - IF

ESTRUCTURA DEL "IF"

- La estructura más utilizada en JavaScript y en la mayoría de lenguajes de programación es la estructura if. Se emplea para tomar decisiones en función de una condición. Su definición formal es:

```
if(condicion) {
  ...
}
```

- Si la condición se cumple *(es decir, si su valor es **true**)* se ejecutan todas las instrucciones que se encuentran dentro de **{...}**. Si la condición no se cumple *(es decir, si su valor es **false**)* no se ejecuta ninguna instrucción contenida en **{...}** y el programa continúa ejecutando el resto de instrucciones del script.

JAVASCRIPT - IF

EJEMPLO

```
var mostrarMensaje = true;

if(mostrarMensaje) {
  alert("Hola Mundo");
}
```

En este ejemplo, el mensaje sí que se muestra al usuario ya que la variable mostrarMensaje tiene un valor de **true** y por tanto, el programa entra dentro del bloque de instrucciones del ***if***.

El mismo ejemplo podría escribirse de esta manera:

```
var mostrarMensaje = true;

if(mostrarMensaje == true) {
  alert("Hola Mundo");
}
```

JAVASCRIPT - IF

```
var mostrarMensaje = true;


// Se comparan los dos valores
if(mostrarMensaje == false) {
  ...
}


// Error - Se asigna el valor "false" a la variable
if(mostrarMensaje = false) {
  ...
}
```

JAVASCRIPT - EJERCICIO

Completar las condiciones de los ***if*** del siguiente script para que los mensajes de los ***alert()*** se muestren siempre de forma correcta:

```
var numero1 = 5;
var numero2 = 8;

if(...) {
  alert("numero1 no es mayor que numero2");
}
if(...) {
  alert("numero2 es positivo");
}
if(...) {
  alert("numero1 es negativo o distinto de cero");
}
if(...) {
  alert("Incrementar en 1 unidad el valor de numero1 no lo hace mayor o igual que numero2");
}
```

JAVASCRIPT - IF...ELSE

ESTRUCTURA DEL "IF...ELSE"

- Usualmente, las decisiones que se deben realizar no son del tipo *"si se cumple la condición, hazlo; si no se cumple, no hagas nada"*.
- Normalmente suelen ser del tipo *"si se cumple esta condición, hazlo; si no se cumple, haz esto otro"*.
- Existe una variante de la estructura **if** llamada ***if...else***.

```
if(condicion) {
  ...
}
else {
  ...
}
```

JAVASCRIPT - IF...ELSE

EJEMPLO

```
var edad = 18;

if(edad >= 18) {
  alert("Eres mayor de edad");
}
else {
  alert("Todavía eres menor de edad");
}
```

Si el valor de ***edad*** es mayor o igual que ***18***, la condición del ***if()*** se cumple y por tanto, se ejecutan sus instrucciones y se muestra el mensaje "**Eres mayor de edad**".

Sin embargo, cuando el valor de la variable ***edad*** no es igual o mayor que ***18***, la condición del ***if()*** no se cumple, por lo que automáticamente se ejecutan todas las instrucciones del bloque ***else { }***.

En este caso, se mostraría el mensaje "**Todavía eres menor de edad**".

JAVASCRIPT - IF...ELSE

EJEMPLO

```
var nombre = "";


if(nombre == "") {
  alert("Aún no nos has dicho tu nombre");
}
else {
  alert("Hemos guardado tu nombre");
}
```

```
if(edad < 12) {
  alert("Todavía eres muy pequeño");
}
else if(edad < 19) {
  alert("Eres un adolescente");
}
else if(edad < 35) {
  alert("Aun sigues siendo joven");
}
else {
  alert("Piensa en cuidarte un poco más");
}
```

JAVASCRIPT - FOR

ESTRUCTURA DEL "FOR"

- La estructura ***for*** nos permite realizar "repeticiones" *(también llamadas bucles)* de una forma muy sencilla.

```
for(inicializacion; condicion; actualizacion) {
  ...
}
```

La idea del funcionamiento de un bucle for es la siguiente: *"mientras la **condición** indicada se siga cumpliendo, repite la ejecución de las instrucciones definidas dentro del **for**. Además, después de cada repetición, **actualiza** el valor de las variables que se utilizan en la condición"*.

- La "***inicialización***" es la zona en la que se establece los valores iniciales de las variables que controlan la repetición.
- La "***condición***" es el único elemento que decide si continua o se detiene la repetición.
- La "***actualización***" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

JAVASCRIPT - FOR

```
var mensaje = "Hola, estoy dentro de un bucle";

for(var i = 0; i < 5; i++) {
  alert(mensaje);
}
```

- La parte de la **inicialización** del bucle consiste en: `var i = 0;`

  En primer lugar se crea la variable **i** y se le asigna el valor de **0**.

  Esta zona de **inicialización** solamente se tiene en consideración justo antes de comenzar a ejecutar el bucle. Las siguientes repeticiones no tienen en cuenta esta parte de inicialización.

- La zona de **condición** del bucle es: `i < 5`

  Los bucles se siguen ejecutando mientras se cumplan las condiciones y se dejan de ejecutar justo después de comprobar que la condición no se cumple. En este caso, mientras la variable **i** valga menos de **5** el bucle se ejecuta indefinidamente.

- La zona de **actualización**, en la que se modifica el valor de las variables que controlan el bucle `i++`

  En este caso, el valor de la variable **i** se incrementa en una unidad después de cada repetición.

  La zona de actualización se ejecuta después de la ejecución de las instrucciones que incluye el for.

JAVASCRIPT - FOR

```
var mensaje = "Hola, estoy dentro de un bucle";

for(var i = 0; i < 5; i++) {
  alert(mensaje);
}
```

Así, durante la ejecución de la *quinta* repetición el valor de ***i*** será ***4***.

Después de la *quinta* ejecución, se actualiza el valor de ***i***, que ahora valdrá ***5***. Como la **condición** es que ***i*** sea menor que ***5***, la condición ya no se cumple y las instrucciones del **for** no se ejecutan una sexta vez.

Normalmente, la variable que controla los bucles **for** se llama ***i***, ya que recuerda a la palabra *índice* y su nombre tan corto ahorra mucho tiempo y espacio.

JAVASCRIPT - FOR

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

for(var i=0; i<7; i++) {
  alert(dias[i]);
}
```

Este ejemplo, muestra en un mensaje, cada día de la semana (previamente almaecenados en un array)

JAVASCRIPT - EJERCICIO

El factorial de un número entero ***n*** es una operación matemática que consiste en multiplicar todos los factores ***n*** x ***(n-1)*** x ***(n-2)*** x **...** x ***1***.

Así, el factorial de ***5*** (escrito como ***5!***) es igual a:

5! = **5** x **4** x **3** x **2** x **1** = **120**

Utilizando la estructura **for**, crear un script que calcule el factorial de un número entero.

JAVASCRIPT - VARIOS

FUNCIONES ÚTILES PARA CADENAS DE TEXTOS

- ***length =*** calcula la longitud de una cadena de texto (el número de caracteres que la forman)

```
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length; // numeroLetras = 10
```

- ***+*** = se emplea para concatenar varias cadenas de texto

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";
var mensaje = mensaje1 + mensaje2; // mensaje = "Hola Mundo"
```

```
var variable1 = "Hola ";
var variable2 = 3;
var mensaje = variable1 + variable2; // mensaje = "Hola 3"
```

- ***toUpperCase()*** = transforma un string en mayúsculas

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase(); // mensaje2 = "HOLA"
```

- ***toLowerCase()*** = transforma un string en minúsculas

```
var mensaje1 = "HolA";
var mensaje2 = mensaje1.toLowerCase(); // mensaje2 = "hola"
```

TRABAJO INTEGRADOS

Plataforma de gestión de gastos e ingresos.

Tarea: comenzar a programar las siguientes pantallas en archivos ***HTML*** diferentes:

- Registro de **GASTOS**
  - Categoría *(select)*
  - Fecha *(input)*
  - Detalle *(textarea)*
  - Importe *(input)*
  - Botón *(submit)*
- Registro de **CATEGORÍAS**
  - Nombre *(input)*
  - Color *(input)*
  - Tipo *(select)*
  - Botón *(submit)*
- Registro de **INGRESOS**
  - Categoría *(select)*
  - Fecha *(input)*
  - Detalle *(textarea)*
  - Importe *(input)*
  - Botón *(submit)*

JAVASCRIPT - FUNCIONES

ESTRUCTURA DE UNA FUNCION

Las funciones en JavaScript se definen mediante la palabra reservada ***function***, seguida del nombre de la función.

```
function nombre_funcion() {
   ...
}
```

```
function suma_y_muestra() {
  resultado = numero1 + numero2;
  alert("El resultado es " + resultado);
}

var resultado;

var numero1 = 3;
var numero2 = 5;

suma_y_muestra();
```

JAVASCRIPT - FUNCIONES

ARGUMENTOS Y RETORNOS

```
// Definición de la función
function suma_y_muestra(primerNumero, segundoNumero) {
  var resultado = primerNumero + segundoNumero;
  alert("El resultado es " + resultado);
}

// Declaración de las variables
var numero1 = 3;
var numero2 = 5;

// Llamada a la función
suma_y_muestra(numero1, numero2);
```

```
function calculaPrecioTotal(precio) {
  var impuestos = 1.16;
  var gastosEnvio = 10;
  var precioTotal = ( precio * impuestos ) + gastosEnvio;
  return precioTotal;
}

var precioTotal = calculaPrecioTotal(23.34);
```

JAVASCRIPT - EJERCICIO

- Escribir el código de una función a la que se pasa como parámetro un número entero y devuelve como resultado una cadena de texto que indica si el número es par o impar. Mostrar por pantalla el resultado devuelto por la función. *(utilizar %)*

- Escribir una función que al ser llamada, muestre el texto que se pasa por parámetro, en mayúscula, minúscula y la cantidad de caracteres.

GIT & GITHUB

¿QUÉ ES GIT?

GIT es un ***software de control de versiones***, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Existe la posibilidad de trabajar de forma remota y una opción es GitHub

DOM - DOCUMENT OBJECT MODEL

Cuando un navegador carga un documento **HTML** crea el modelo **HTML DOM** del documento.

El modelo **HTML DOM** define una estructura de árbol de objetos que refleja la naturaleza jerárquica del lenguaje HTML.

El modelo **HTML DOM** es un estándar que define clases que representan:

- **Elementos** HTML, por ejemplo, una tabla, un párrafo, e incluso el propio documento HTML.
- **Propiedades** de los elementos HTML, por ejemplo, el borde de una tabla (*border*) o el ancho de una imagen (*width*).
- **Métodos** para acceder a los objetos que representan los elementos HTML.
- **Eventos** para capturar la interacción del usuario de la aplicación con los elementos HTML en la pantalla del navegador.

JAVASCRIPT - DOM

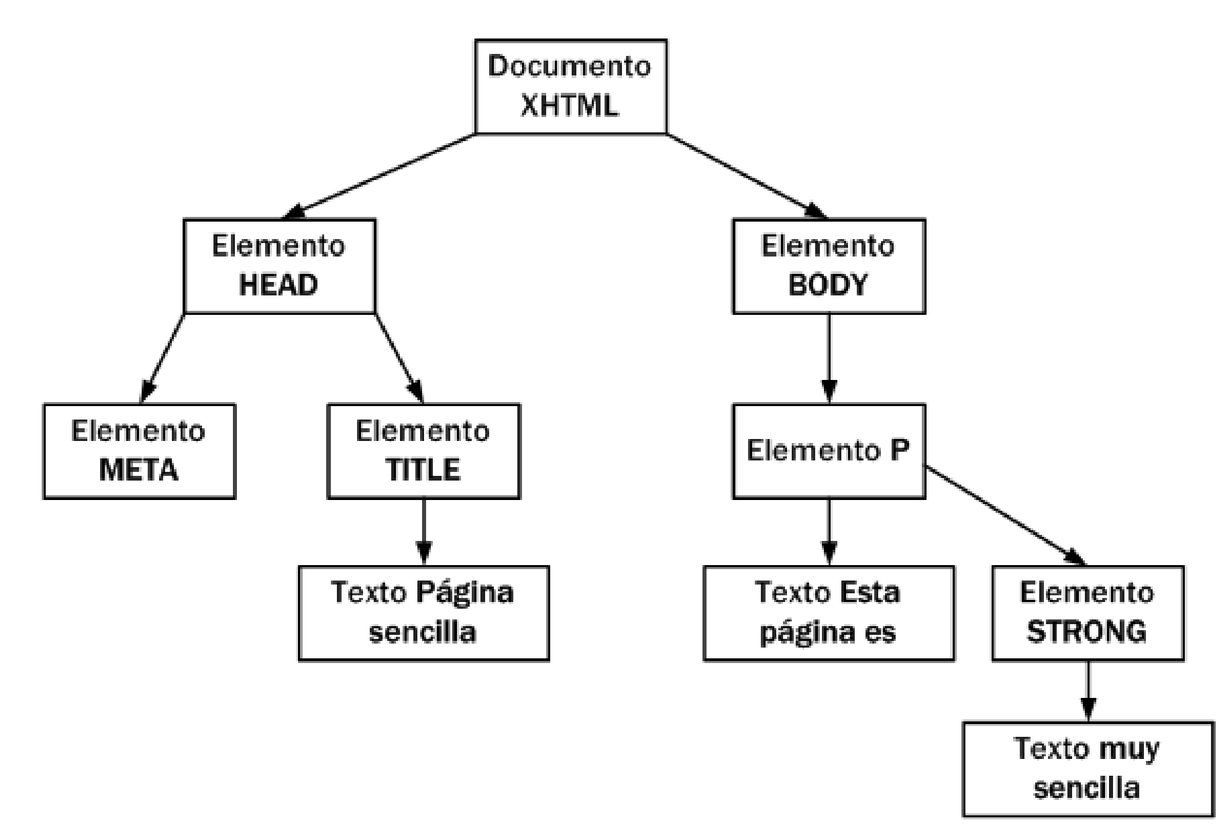
Página HTML

```
<html >
<head>

<title>Página sencilla</title>
</head>

<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

Árbol DOM



JAVASCRIPT - DOM

- En el esquema anterior, cada rectángulo representa un **nodo DOM** y las flechas indican las **relaciones** entre nodos.
- Dentro de cada **nodo**, se ha incluido su tipo (que se verá más adelante) y su contenido.
- La raíz del árbol de nodos de cualquier página XHTML siempre es la misma: un nodo de tipo especial denominado "***Documento***".
- A partir de ese nodo raíz, cada etiqueta XHTML se transforma en un nodo de tipo "***Elemento***".
- La conversión de etiquetas en nodos se realiza de forma jerárquica.
- Es importante recordar que el acceso a los nodos, su modificación y su eliminación solamente es posible cuando el árbol DOM ha sido construido completamente, es decir, después de que la página XHTML se cargue por completo.

ACCESO DIRECTO A NODOS

Se puede acceder a todos los elementos del árbol DOM, con cualquiera de estas funciones especiales.

- **getElementsByTagName(*etiqueta*)** = esta función obtiene todos los elementos de la página **XHTML** cuya etiqueta sea igual que el parámetro que se le pasa a la función.

```
var parrafos = document.getElementsByTagName("p");
```

ACCESO DIRECTO A NODOS

- **getElementsByName(*nombreEtiqueta*)** = esta función es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo name sea igual al parámetro proporcionado.

```
var parrafoEspecial = document.getElementsByName("especial");


<p name="prueba">...</p>
<p name="especial">...</p>
<p>...</p>
```

ACCESO DIRECTO A NODOS

- **getElementById(*id*)** = esta función es una de la más utilizadas, y devuelve el elemento HTML cuyo atributo ***id*** coincide con el parámetro indicado en la función.

```
var cabecera = document.getElementById("cabecera");


<div id="cabecera">
  <a href="/" id="logo">...</a>
</div>
```

ACCESO DIRECTO A ATRIBUTOS

Una vez que se ha accedido a un nodo, el siguiente paso natural consiste en acceder y/o modificar sus atributos y propiedades. Mediante **DOM**, es posible acceder de forma sencilla a todos los atributos **HTML** y todas las propiedades **CSS** de cualquier elemento de la página.

Los atributos **HTML** de los elementos de la página se transforman automáticamente en propiedades de los nodos.
Para acceder a su valor, simplemente se indica el nombre del atributo **HTML** detrás del nombre del nodo.

JAVASCRIPT - DOM

```
var enlace = document.getElementById("enlace");
alert(enlace.href); // muestra http://www...com

<a id="enlace" href="http://www...com">Enlace</a>
```

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.style.fontWeight);  // muestra "bold"

<p id="parrafo" style="font-weight: bold;">...</p>
```

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.class); // muestra "undefined"
alert(parrafo.className); // muestra "normal"

<p id="parrafo" class="normal">...</p>
```

EVENTOS

El nombre de cada evento se construye mediante el prefijo ***on***, seguido del nombre en inglés de la acción asociada al evento.

Así, el evento de pinchar un elemento con el ratón se denomina ***onclick*** y el evento asociado a la acción de mover el ratón se denomina ***onmousemove***.

La siguiente tabla resume los eventos más importantes definidos por JavaScript:

JAVASCRIPT - DOM

EVENTOS

Evento	Descripción	Elementos para los que está definido
`onblur`	Deseleccionar el elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onchange`	Deseleccionar un elemento que se ha modificado	`<input>`, `<select>`, `<textarea>`
`onclick`	Pinchar y soltar el ratón	Todos los elementos
`ondblclick`	Pinchar dos veces seguidas con el ratón	Todos los elementos
`onfocus`	Seleccionar un elemento	`<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, `<body>`
`onkeydown`	Pulsar una tecla (sin soltar)	Elementos de formulario y `<body>`
`onkeypress`	Pulsar una tecla	Elementos de formulario y `<body>`
`onkeyup`	Soltar una tecla pulsada	Elementos de formulario y `<body>`
`onload`	La página se ha cargado completamente	`<body>`
`onmousedown`	Pulsar (sin soltar) un botón del ratón	Todos los elementos

JAVASCRIPT - DOM

EVENTOS

`onmousemove`	Mover el ratón	Todos los elementos
`onmouseout`	El ratón *"sale"* del elemento (pasa por encima de otro elemento)	Todos los elementos
`onmouseover`	El ratón *"entra"* en el elemento (pasa por encima del elemento)	Todos los elementos
`onmouseup`	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
`onreset`	Inicializar el formulario (borrar todos sus datos)	`<form>`
`onresize`	Se ha modificado el tamaño de la ventana del navegador	`<body>`
`onselect`	Seleccionar un texto	`<input>`, `<textarea>`
`onsubmit`	Enviar el formulario	`<form>`
`onunload`	Se abandona la página (por ejemplo al cerrar el navegador)	`<body>`

JAVASCRIPT - DOM

EVENTOS

```
<input type="button" value="Pinchame y verás" onclick="alert('Gracias por pinchar');" />
```

```
<div onclick="alert('Has pinchado con el ratón');" onmouseover="alert('Acabas de pasar el ratón por encima');">
  Puedes pinchar sobre este elemento o simplemente pasar el ratón por encima
</div>
```

```
<body onload="alert('La página se ha cargado completamente');">
    ...
</body>
```

JQUERY

¿QUÉ ES JQUERY?

JQuery es una biblioteca de JavaScript que simplifica la forma de desarrollar aplicaciones web. Las aplicaciones que utilizan **JQuery** suelen necesitar menos tiempo y menos código que las aplicaciones hechas con **JS** puro. Por este motivo, **JQuery** es muy popular y se utiliza en montones de páginas web.

JQuery permite manipular elementos del DOM (textos, imágenes, enlaces, etc.) , cambiar el diseño CSS o realizar peticiones ***AJAX*** utilizando instrucciones simples, a través de un código muy conciso y sencillo.

JQUERY

AGREGANDO JQUERY A NUESTRO SITIO

Hay 2 formas de agregar JQuery a nuestros sitios web:

1) Descargar el archivo de ***jquery.com***, y añadirlo a nuestro HTML, como cualquier otro archivo de JavaScript.

```
1 <script src="jquery.js"></script>
```

2) Incluir la biblioteca **JQuery** en línea

```
1 <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
```

En la página oficial existen dos versiones de la biblioteca **JQuery**, la versión comprimida o minimizada (*jquery.min.js*) y la no comprimida (*jquery.js*).

EL BLOQUE $(DOCUMENT).READY()

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. jQuery permite detectar dicho estado a través de la declaración $(document).ready() de forma tal que el bloque se ejecutará sólo una vez que la página este disponible.

```
$(document).ready(function() {
    console.log('el documento está preparado');
});
```

JQUERY - SELECCION DE ELEMENTOS

EL BLOQUE $(DOCUMENT).READY()

El concepto más básico de **jQuery** es el de *"seleccionar algunos elementos y realizar acciones con ellos"*.

La biblioteca soporta gran parte de los selectores **CSS3** y varios más no estandarizados.

En ***api.jquery.com/category/selectors*** se puede encontrar una completa referencia sobre los selectores de la biblioteca.

A continuación veremos algunas técnicas comunes para la selección de elementos:

JQUERY - SELECCION DE ELEMENTOS

SELECCIÓN DE ELEMENTOS EN BASE A SU "ID"

```
$('#myId'); // notar que los IDs deben ser únicos por página
```

SELECCIÓN DE ELEMENTOS EN BASE AL NOMBRE DE CLASE

```
$('.myClass');


$('div.myClass'); // si se especifica el tipo de elemento,
                  // se mejora el rendimiento de la selección
```

JQUERY - SELECCION DE ELEMENTOS

SELECCIÓN DE ELEMENTOS POR SU ATRIBUTO

```
$('input[name=apellido]'); // tenga cuidado, que puede ser muy lento
```

SELECCIÓN DE ELEMENTOS EN FORMA DE SELECTOR CSS

```
$('#contenidos ul.personas li');
```

JQUERY - SELECCION DE ELEMENTOS

PSEUDO-SELECTORES

```
// selecciona el primer elemento <a> con la clase 'external'
$('a.external:first');

// selecciona todos los elementos <tr> impares de una tabla
$('tr:odd');

// selecciona todos los elementos del tipo input dentro del formulario #myForm
$('#myForm :input');

// selecciona todos los divs visibles
$('div:visible');

// selecciona todos los divs excepto los tres primeros
$('div:gt(2)');

// selecciona todos los divs actualmente animados
$('div:animated');
```

JQUERY - SELECCION DE ELEMENTOS

SELECCIÓN DE ELEMENTOS DE UN FORMULARIO

jQuery ofrece varios pseudo-selectores que ayudan a encontrar elementos dentro de los formularios, éstos son especialmente útiles ya que dependiendo de los estados de cada elemento o su tipo, puede ser difícil distinguirlos utilizando selectores CSS estándar.

JQUERY - SELECCION DE ELEMENTOS

- `:button` Selecciona elementos `<button>` y con el atributo `type='button'`
- `:checkbox` Selecciona elementos `<input>` con el atributo `type='checkbox'`
- `:checked` Selecciona elementos `<input>` del tipo checkbox seleccionados
- `:disabled` Selecciona elementos del formulario que están deshabitados
- `:enabled` Selecciona elementos del formulario que están habilitados
- `:file` Selecciona elementos `<input>` con el atributo `type='file'`
- `:image` Selecciona elementos `<input>` con el atributo `type='image'`
- `:input` Selecciona elementos `<input>`, `<textarea>` y `<select>`
- `:password` Selecciona elementos `<input>` con el atributo `type='password'`
- `:radio` Selecciona elementos `<input>` con el atributo `type='radio'`
- `:reset` Selecciona elementos `<input>` con el atributo `type='reset'`
- `:selected` Selecciona elementos `<options>` que están seleccionados
- `:submit` Selecciona elementos `<input>` con el atributo `type='submit'`
- `:text` Selecciona elementos `<input>` con el atributo `type='text'`

UTILIZANDO PSEUDO-SELECTORES EN ELEMENTOS DE FORMULARIOS

```
// obtiene todos los elementos inputs dentro
// del formulario #myForm
$('#myForm :input');
```

BASE DE DATOS

¿QUÉ ES SQL?

SQL *(Structured Query Language)* es un lenguaje estándar para acceder y manipular base de datos.

¿Qué se puede hacer con SQL?

- Ejecutar *consultas* a una base de datos
- Obtener datos de una base de datos
- Insertar, actualizar y eliminar registros en una base de datos
- Crear base de datos, tablas, vistas, etc.
- Configurar permisos

USANDO SQL EN NUESTRO SITIO WEB

Para desarrollar un sitio web que utilice datos almacenados en una base de datos, necesitamo:

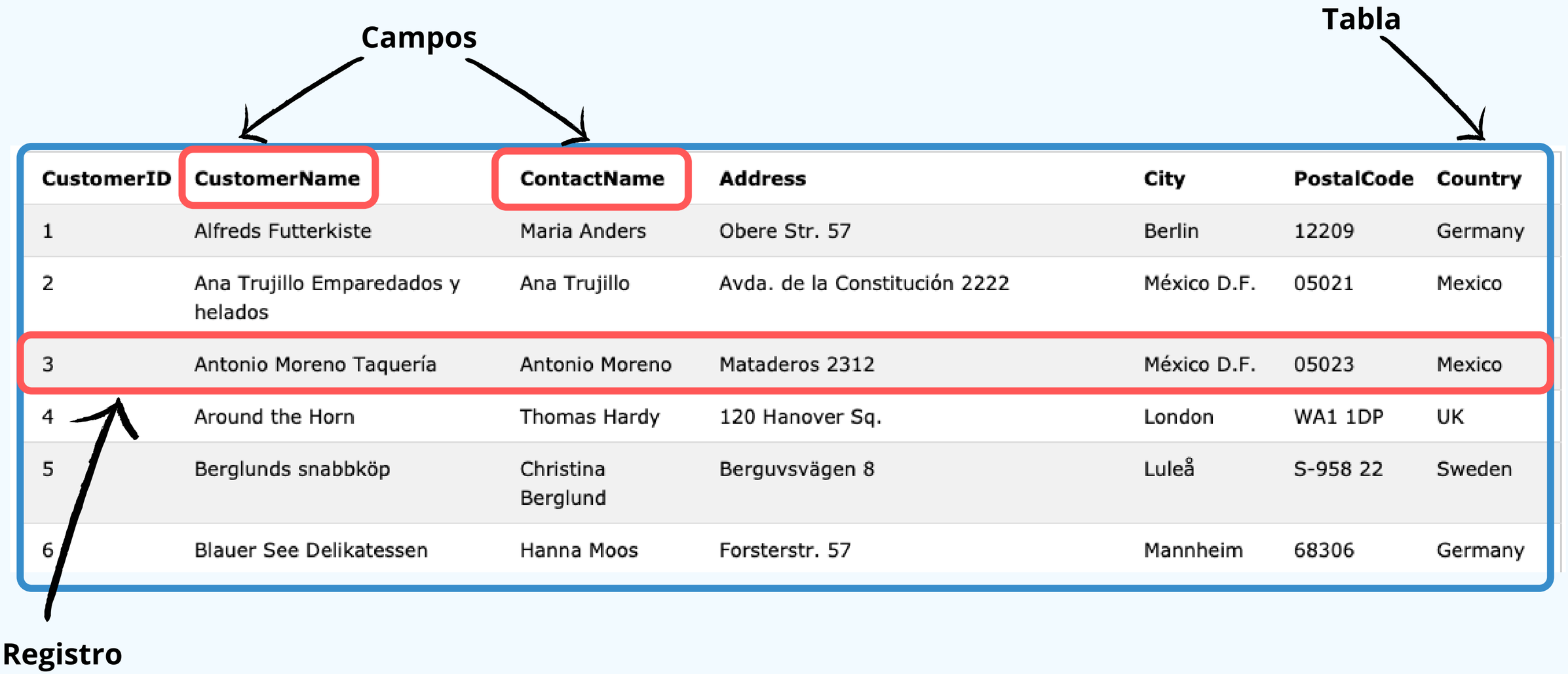
- Sistema de Gestión de Base de Datos - RDBMS *(ej: MySQL, SQL Server, Access, etc)*.
- Un lenguaje de programación del lado del servidor *(ej: PHP, .NET, etc)*.
- Usar SQL para acceder y manipular los datos que deseamos.
- HTML y CSS para mostrar los datos en nuestro sitio.

SQL

INTRO

- En un Sistema de Gestión de Base de Datos, la información se almacena en ***tablas***.
- Una ***tabla*** es una colleción de datos relacionados, que consiste en *filas* y *columnas*.
- Cada tabla se divide a su vez, en entidades más pequeñas llamadas ***campos***.
- Un ***campo*** es una columna, dentro de la ***tabla***, diseñada para guardar información específica de cada registro de la ***tabla***.
- Un ***registro***, también llamado ***fila*** es cada entrada individual que existe en la ***tabla***.

SQL - TABLAS



CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany

SQL - TABLAS

- Una base de datos, usualmente, contiene una o más ***tablas***
- Cada ***tabla*** se identifica con un ***nombre*** (ej: Clientes, Pedidos, Estudiantes, etc.)
- Las ***tablas*** contienen ***registros*** (*filas*) con datos
- Cada ***dato*** es información que el programador considera escencial para guardar, y que tiene relación a esa tabla.

SQL - COMANDOS

ALGUNOS COMANDOS DE SQL

- ***SELECT =*** obtiene datos de la base de datos (de una tabla o más tablas)
- ***UPDATE*** = actualiza datos de la base de datos
- ***DELETE*** = elimina datos de la base de datos
- ***INSERT INTO*** = inserta un nuevo dato o registro
- ***CREATE DATABASE*** = crea una nueva base de datos
- ***ALTER DATABASE*** = mofidica una base de datos
- ***CREATE TABLE*** = crea una nueva tabla
- ***ALTER TABLE*** = modifica una tabla
- ***DROP TABLE*** = elimina una tabla

SQL - SINTAXIS

SELECT

Aquí ***column1***, ***column2***, ... son los nombre de los campos de la tabla de la cual queremos obtener datos (*table_name*).

```
SELECT column1, column2, ...
FROM table_name;
```

Si quisieramos seleccionar todos los campos de una tabla, podemos usar el *

```
SELECT * FROM table_name;
```

SQL - SINTAXIS

WHERE

La cláusula ***WHERE*** se utiliza para "filtrar" registros o filas.

Se utiliza para extraer SÓLO aquellos registros que CUMPLAN con determinada condición.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Por ejemplo:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

SQL - SINTAXIS

OPERADORES: *AND*, *OR*, *NOT*

La cláusula ***WHERE*** se puede combinar con los operadores ***AND***, ***OR*** y ***NOT***.

Los operadores ***AND*** y ***OR*** se utilizan para filtrar registros basados en más de una condición:

- El operador ***AND*** muestra un registro si TODAS las condiciones separadas por "***AND***" son **VERDADERAS**.
- El operador ***OR*** muestra un registro si ALGUNA de las condiciones separadas por "OR" es **VERDADERA**.

El operador ***NOT*** muestra un registro si la condición es **NO VERDADERA**.

SQL - SINTAXIS

AND

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

MONGODB

¿QUÉ ES MONGODB?

MongoDB es una base de datos ***NoSQL*** orientada a documentos.

Es la base de datos *NoSQL* más utilizada en el mundo.

¿Qué es una base de datos NoSQL?

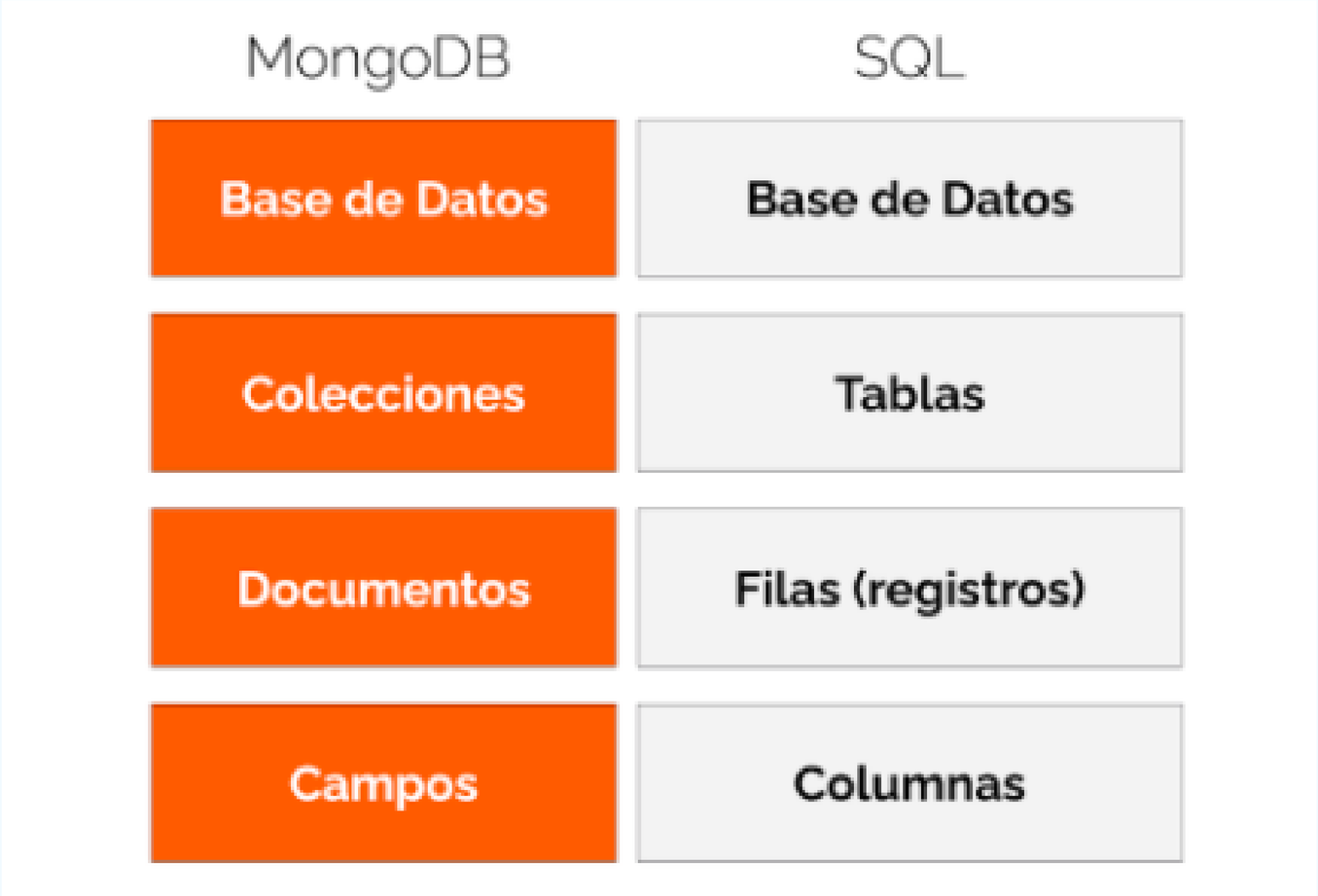
- NoSQL viene de **N**ot **O**nly **SQL** (no solo SQL)
- Surgen por una gran demanda de bases de datos que puedan trabajar con datos masivos de forma más eficiente
- Las bases de datos **NoSQL** no utilizan tablas ni registros (como si lo hacen las DB Relacionales)
- No necesitan una estructura fija

MONGODB

Ejemplos de base de datos NoSQL

MONGODB

¿Cómo funciona MongoDB?



MONGODB

Ejemplo: base de datos para una tienda online

Base de Datos	Tienda	
Colecciones	Usuarios	Pedidos
Documentos	{Nombre: 'Juan', edad: 25}	{...}
	{Nombre: 'Felipe'}	{...}

MONGODB

¿CÓMO SE GUARDAN LOS DOCUMENTOS?

Los documentos de una base de datos **MongoDB** se almacenan utilizando el formato **JSON** o **BSON**.

BSON es una versión binaria de un archivo **JSON** (son documentos compuestos por clave y valor)

JSON (BSON)

```
{
    “nombre”: “Juan”,
    “edad”: 25
    “dirección”:
          {
                “ciudad”: “Barcelona”
          },
    “aficiones”:[
          {“nombre”: “Fútbol” },
          {“nombre”: “Esquí” }
    ]
}
```

VENTAJAS

Hay varias ventajas al utilizar una base de datos MongoDB en una aplicación:

- **Es más rápido,** ya que almacena todo los datos en un mismo sitio (una colección).
- **No hace falta unir tablas** como en SQL.
- Da **más flexibilidad** para diseñar el *schema* y crear relaciones complejas.