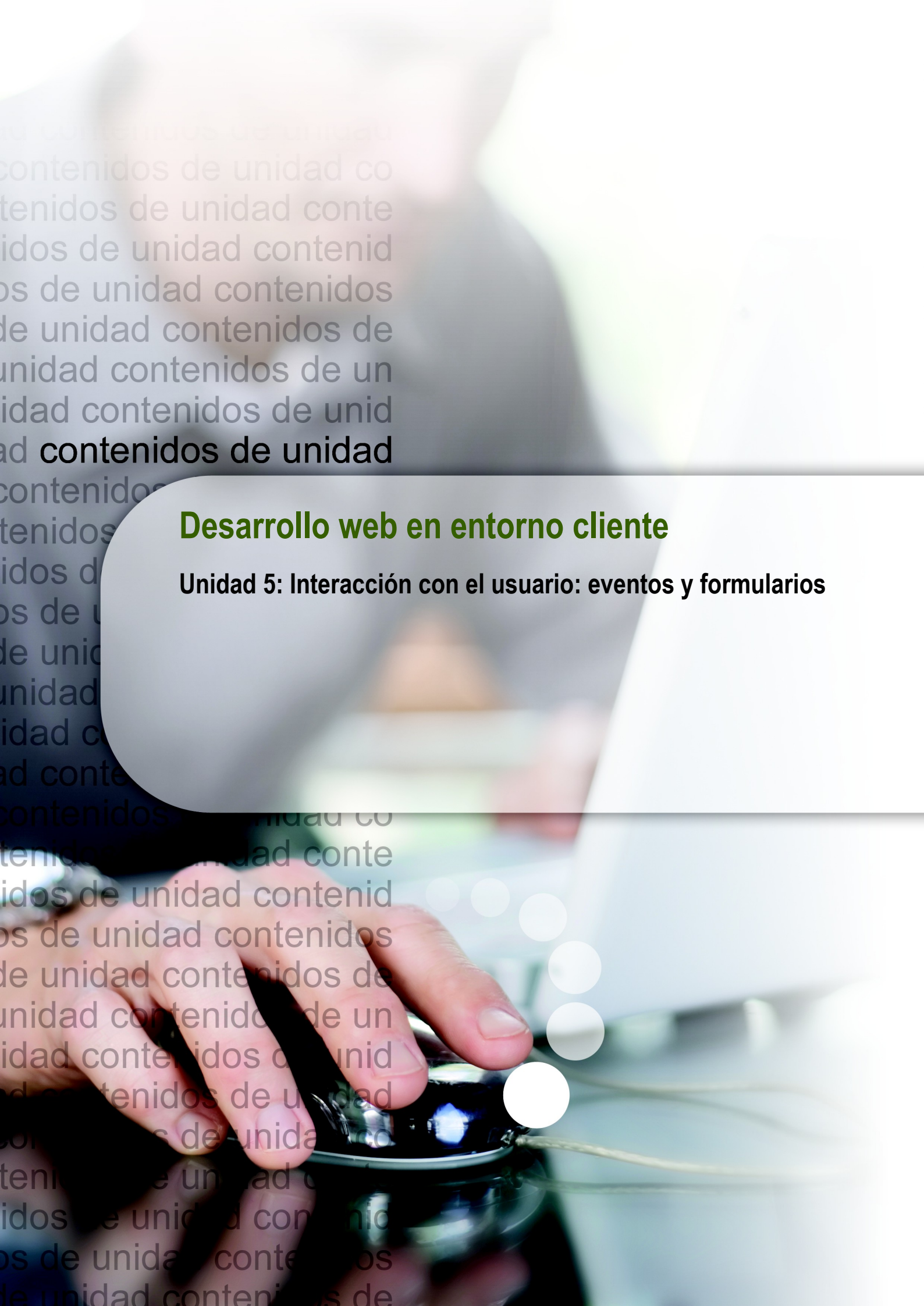
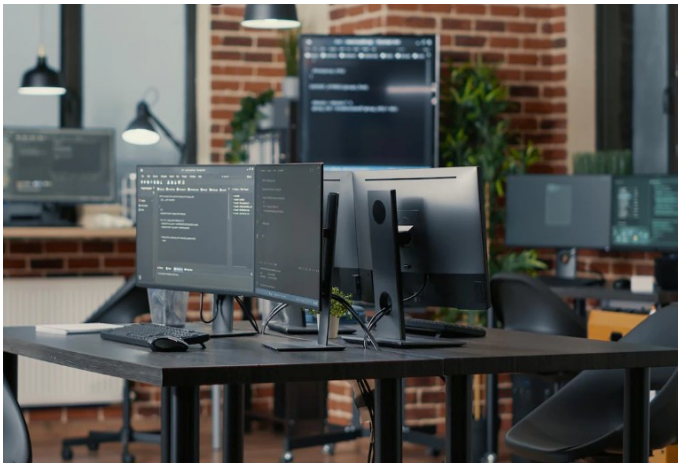


# Desarrollo web en entorno cliente

## Unidad 5: Interacción con el usuario: eventos y formularios

## ÍNDICE

# INTRODUCCIÓN



Después de estudiar en profundidad el **lenguaje JavaScript** y todas sus complejidades, en esta unidad vamos a centrarnos más en la aplicación a las **páginas web**, y más en concreto, en la interacción con el usuario.

Empezaremos con los **eventos**, viendo que son, como se utilizan en JavaScript y como se utilizan y manejan. Esto nos permitirá un primer paso para **interactuar con el usuario**.

Posteriormente veremos uno de los **objetos** más utilizados en las aplicaciones web, como son los **formularios**. Qué son, cómo crearlos, cómo acceder a sus campos, validarlos, y un largo etcétera.

Veremos también una serie de conceptos, que, aunque más independientes entre sí, se utilizan también en la interacción con el usuario. Aquí hablaremos de **cookies** y **parámetros**.

Finalmente veremos el concepto de **framework** y como éste puede simplificar en gran medida el proceso de programación actualmente.

Como es habitual, todos estos conceptos necesitarán del apoyo de otros módulos, especialmente de **programación web**, pero también de programación tradicional, donde también se usan eventos.

## *OBJETIVOS / CAPACIDADES*

***En esta unidad de aprendizaje, las capacidades que más se van a trabajar son:***

- ✔ **Analizar y utilizar los recursos y oportunidades de aprendizaje relacionadas con la evolución científica, tecnológica y organizativa del sector y las tecnologías de la información y la comunicación, para mantener el espíritu de actualización y adaptarse a nuevas situaciones laborales y personales.**

## *PROYECTO DE LA UNIDAD*

Antes de empezar a trabajar el contenido, te presentamos la **actividad** que está relacionada con esta unidad de aprendizaje. Se trata de un **caso práctico** basado en una **situación real** con la que te puedes encontrar en tu puesto de trabajo. Con esta actividad se evaluará la puesta en práctica de los **criterios de evaluación** vinculados al resultado de aprendizaje que se trabaja en esta unidad. Para realizarla deberás hacer lo siguiente: lee el enunciado que te presentamos a continuación, dirígete al área general del módulo profesional, concretamente a la actividad de evaluación que se encuentra dentro de esta unidad, allí encontrarás todos los detalles sobre fecha y forma de entrega, objetivos... A lo largo de la unidad irás adquiriendo los conocimientos necesarios para ir elaborando este proyecto.

**Enunciado**:

**Construyendo el esqueleto del proyecto secreto**

Por fin podemos saber unos cuantos detalles más del nuevo proyecto secreto de la empresa. Aún no nos han querido confirmar la temática, pero sí un par de cosas:

- Será una aplicación web, que capturará eventos del ratón.
- Habrá formularios que requerirán de validación mediante expresiones regulares.

Así pues, nos piden:

- A partir de la plantilla dada (index.html), crea un archivo llamado evaluate.js con el código necesario para validar los diferentes campos de texto.
- Cada vez que se cumpla una condición haz que aparezca un icono de ✔ para que el usuario vea que ese campo es correcto.
- Mientras la condición no se cumpla el icono no se ha de ver.
- Si por algún motivo, la condición deja de cumplirse, tampoco ha de verse el icono.

- Al pasar el ratón por encima de un campo ha de aparecer un mensaje si no está correcto.

El programa tiene que estar comentado y probado en al menos 2 navegadores.


*Pulsa aquí para poder descargar la plantilla.*

# 1. MODELO DE GESTIÓN DE EVENTOS. TIPOS Y CAPTURA DE EVENTOS. MODELO DE EVENTOS ESTÁNDAR

DESTACADO

*Un evento es una acción que sucede en la ejecución de una aplicación, en nuestro caso, en una página web. Pueden ser acciones como un clic del ratón, una pulsación del teclado, la carga de una imagen…*

A continuación, veremos algunos de los **aspectos más relevantes** sobre los **eventos**:

- **Eventos importantes**: en **JavaScript** hay multitud de posibles eventos. Algunos de los más importantes podrían ser:
  - ✔ AnimationEvent.
  - ✔ DragEvent.
  - ✔ FocusEvent.
  - ✔ HashChangeEvent.
  - ✔ KeyboardEvent.
  - ✔ MouseEvent.
  - ✔ PageTransitionEvent.
  - ✔ ProgressEvent.
  - ✔ TouchEvent.
  - ✔ UiEvent.
  - ✔ WheelEvent.
- **Capturar evento HTML/JavaScript**: lo que hace especiales a los eventos es que pueden ser "escuchados" o **capturados**, para posteriormente asociarles

una **acción**. Por ejemplo, si quisiéramos que nos aparezca la fecha actual al pulsar un botón:

```
<html>
    <body>
        <button onclick = "printDate();"> The time is?</button>
        <p id="demo"></p>
        <script>
            function printDate() {
                document.getElementById('demo').innerHTML =
Date();
            }
        </script>
    </body>
</html>
```

Vemos como capturamos el evento mediante el atributo "onclick" de la etiqueta **<button>**, y como usamos JavaScript para mostrar la fecha.

➔ **Capturar evento solo con JavaScript**: si quisiéramos el mismo ejemplo, pero **100% en JavaScript** podríamos hacer:

```
<button>The time is?</button>
<p id="demo"></p>
<script>
    document.getElementsByTagName("button")[0].onclick =
printDate;
    function printDate() {
        document.getElementById('demo').innerHTML = Date();
    }
</script>
```


*En el siguiente capítulo veremos de forma más detallada como podemos capturar y manejar los eventos producidos.*

## 2. MANEJADORES DE EVENTOS

Hay diferentes **opciones para capturar los eventos**, como hemos visto, algunas mezclan HTML y JavaScript y otras tan sólo usan el segundo lenguaje.

En concreto hablaremos de **tres formas diferentes de capturarlos**:

- Captura en línea.
- Captura tradicional.
- Captura por escucha.

➔ **Captura en línea**: dentro de la etiqueta `<button>` utilizamos el atributo **onclick**, para llamar a la función de JavaScript asociada. Sería el primer ejemplo del capítulo anterior.

Una cosa a tener en cuenta, es que, a diferencia de HTML, JavaScript sí que es **case-sensitive**, por lo que la llamada a la función `printDate()` tendrá que respetar las minúsculas.

➔ **Captura tradicional**: la captura tradicional (segundo ejemplo) permite separar la **vista del controlador**, por lo que el código queda mucho más limpio y modular. En este caso la parte de HTML tan solo contiene **etiquetas**, sin código asociado.

Es en la parte del **script**, donde primero buscamos el botón, y le asignamos como atributo la función `printDate` (sin paréntesis, porque sólo indicamos el nombre, no hacemos que la ejecute). El resto funciona igual que en primer ejemplo.

➔ **Captura por escucha**: al igual que en la captura tradicional, permite **separar vista y controlador**. La diferencia aquí es la agilidad a la hora de trabajar que aporta, ya que podemos, en cualquier momento de la ejecución del código,

decirle a un elemento que “escuche” un evento o deje de “escucharlo”, si nos conviene.

Además, aporta información de cuándo se produce el evento, si es en fase de **capturing** o **bubbling**.

A continuación, veremos un **ejemplo** de cómo **capturar un evento por escucha**:

*Ejemplo: capturar por escucha*

Veamos el ejemplo:

```
</html>
<html> <body>
    <button>The time is?</button>
    <p id="demo"></p>
    <script>
        document.getElementsByTagName( "button" )
[0].addEventListener("click", printDate, false);
         function printDate() {
             document.getElementById('demo').innerHTML = Date(); }
    </script>
</body>
</html>
```

La función addEventListener tiene tres parámetros:

- El nombre del evento.
- La función a ejecutar cuando se produzca.
- Un booleano indicando si haremos caso en fase de capturing o bubbling.

Para eliminar la escucha de un evento usaríamos el método removeEventListener, de la misma manera.

*Vídeo: eventos onmousedown – onmouseup*


*Visualiza el siguiente vídeo podrás ver aspectos relevantes sobre los eventos eventos onmousedown – onmouseup.*
*https://www.youtube.com/embed/eZKlbZ2hRM0*

## Actividad de aprendizaje 1: trabajar con eventos

Nuestra siguiente labor en la empresa consiste en la elaboración de una aplicación web para la consulta de datos. En la empresa son muy cautos y no nos han querido explicar aún la temática. Lo harán más adelante, pero de momento parece que tiene que ver con la interacción con los usuarios.

Por eso debemos empezar a trabajar con los eventos, y nos piden:

- Generar una página HTML con 4 botones `<input type= "button"/>` o `<button>`.
- Cada botón tendrá que responder a un tipo de evento de ratón.
    - onclick.
    - ondblclick.
    - onmousedown.
    - onmouseover.
- El evento tendrá que mostrar una ventana emergente, indicando el nombre del tipo de evento reconocido.
- La captura de los eventos ha de hacerse mediante escucha.

Ponte de acuerdo con un compañero y entre ambos realizad esta actividad.

Entregad el archivo .html en el espacio habilitado para ello.

## Cuestionario

**Lee el enunciado e indica la opción correcta**:

¿Qué método de captura de eventos utiliza el atributo onclick dentro de la etiqueta <button>?

**a.** Captura en línea.

**b.** Captura tradicional.

**c.** Captura por escucha.

**Lee el enunciado e indica la opción correcta**:

¿Qué método de captura de eventos permite separar la vista del controlador y mantener un código más limpio y modular?

**a.** Captura en línea.

**b.** Captura tradicional.

**c.** Captura por escucha.

**Lee el enunciado e indica la opción correcta**:

¿Qué método de captura de eventos proporciona información sobre cuándo se produce el evento y permite agregar o quitar escuchadores en cualquier momento?.

**a.** Captura en línea.

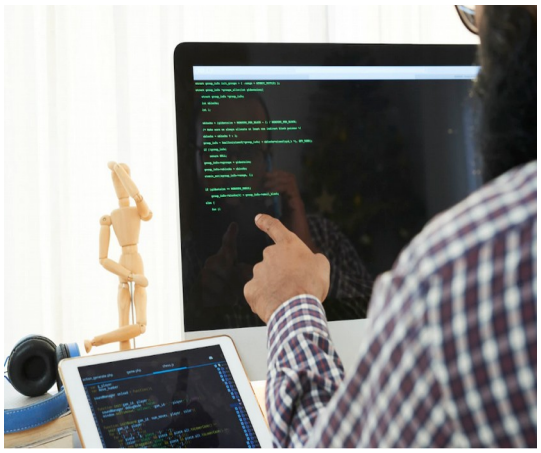
**b.** Captura tradicional.

**c.** Captura por escucha.

## 3. UTILIZACIÓN DE FORMULARIOS DESDE CÓDIGO. EL OBJETO FORMULARIO. CARGAR, MOSTRAR Y OCULTAR

Los formularios son uno de los métodos más utilizados para la interacción con el usuario en aplicaciones web. En los siguientes capítulos hablaremos acerca de ellos.


*Un formulario es una etiqueta HTML `<form>`, que permite crear una interfaz de diálogo cliente-servidor. Al ser un elemento del DOM, lo podemos manipular mediante JavaScript.*



La manipulación se basa sobre todo a acciones de **inspección** y **validación**. Es cierto que con el nuevo estándar HTML5, algunas de estas acciones ya son soportadas nativamente, pero con el uso de JavaScript tenemos un control más amplio sobre ellos.


*Para trabajar con formularios desde JavaScript disponemos del objeto formulario (form), que en realidad es una propiedad del objeto document, que se corresponde con la etiqueta `<form>` de HTML.*

Una vez construido un formulario HTML podemos acceder a él de varias formas usando JavaScript:

- Usando el **método getElementById()** y el identificador del formulario:

```
var formulario = document.getElementById("formulario1");
```

- A través del **método getElementsByTagName()**, con lo que podemos acceder a todos los formularios del documento:

```
var formularios = document.getElementsByTagName('form');
var primerFormulario = formularios[0];
```

➔ Con la **colección forms[]** del objeto document, con lo que conseguimos algo parecido a lo anterior. Lo podríamos hacer de diferentes maneras:

```
var formularios = document.forms;

var formulario = formularios[0];
var formulario = document.forms[0];
var formulario = formularios["formulario"];
```

*Ejemplo: formularios*

Una vez accedido al formulario que queremos usar, podremos hacer multitud de cosas con él, entre ellas, por ejemplo, mostrarlos u ocultarlos usando style.display:

```
document.getElementById("formulario").style.display = "none";
document.getElementById("formulario ").style.display = "block";
```

O, si usamos alguna librería como por ejemplo jQuery, lo podríamos hacer de una forma aún mucho más sencilla:

```
$("#formulario ").show();
$("#formulario ").hide();
```

# 4. ACCESO A LOS MIEMBROS DEL FORMULARIO. PROPIEDADES Y MÉTODOS DE UN FORMULARIO. CONTROLES

En el capítulo anterior hemos visto como acceder a los formularios de HTML desde **JavaScript**, utilizando diferentes métodos. En este haremos inciso en cómo acceder a los miembros de un formulario, **propiedades** y **métodos**.

Para las explicaciones seguiremos el siguiente ejemplo de **formulario HTML**:

```
<form name = "formContacto" action =
"mailto:direccion@dominio.com" enctype="text/plain">
    <input type="TextArea" name="mensaje" value="" size="12">
    <input type="button" value="Enviar" onclick = "validar()">
</form>
```

Se trata de un pequeño formulario con una **caja de texto**, donde pondremos el **mensaje** y un **botón** para enviar.

- **Acceder al formulario**: podemos usar este formulario desde **JavaScript**, accediendo, por ejemplo, por su nombre y desde allí habría varias maneras de acceder a algún elemento, como la **caja de texto**:

```
document.formContacto.elements[0]
document.formContacto.elements["mensaje"]
document.formContacto.mensaje
```

- **Acceder al contenido**: con eso estaríamos accediendo al elemento donde guardamos el mensaje. Si ahora queremos acceder a su contenido, bastaría con usar la propiedad **value**, con cualquiera de sus variantes:

```
alert(document.formContacto.mensaje.value)
```

- **Propiedades**: la propiedad "**elements**" es una propiedad del formulario que permite acceder a todos sus elementos. Además de ésta, las principales propiedades de un formulario son:
  - ✔ **action**: acción a realizar al enviar el formulario.

- ✔ **length**: número de campos del formulario.
- ✔ **name**: nombre del formulario.
- ✔ **method**: método de envío (GET/POST).
- ✔ **target**: la ventana a la que va dirigido el formulario.

➔ **Métodos**: en cuanto a los métodos del formulario, basícamente se trata de:

- ✔ **submit**: envía el formulario.
- ✔ **reset**: reinicia el formulario a los valores por defecto.

Para acceder tanto a las propiedades como a los métodos lo haremos con la notación por punto habitual.

*Ejemplo: formulario*

Así, un ejemplo de uso del formulario podría ser:

```
<script>
    function validar() {
        if (document.formContacto.mensaje.value == "") {
            alert("Debe rellenar el mensaje")
        } else {
            document.formContacto.submit()
        }
    }
</script>
```

*Vídeo: acceso a formulario*


*Visualiza este vídeo en el que vemos cómo acceder a formularios.*

## 5. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

Una vez creado un formulario en HTML, éste tiene un cierto **aspecto** y **comportamiento**, en función del valor de sus **etiquetas** y **propiedades**.


*Si en cualquier momento queremos cambiar su apariencia, lo ideal sería hacerlo mediante estilos CSS, lo que nos permite un mayor control sobre el diseño. Sin embargo, desde JavaScript también podríamos modificar estos aspectos en tiempo de ejecución de la web, pudiendo, por ejemplo, modificar el comportamiento en función de valores del formulario.*

A continuación, veremos cómo **añadir un campo a un formulario** y cómo **enviar el mensaje**:


- **Añadir campo**: imaginemos el caso en que tenemos un formulario de contacto, similar al del capítulo anterior, pero ahora añadimos un campo para seleccionar el departamento con el cual queremos contactar.

```
<form name="formContacto" action =
"mailto:direccion@dominio.com" enctype="text/plain">
    <input type="Text" name="mensaje" value="" size="12">
    <input type="button" value="Enviar" onclick="enviar()">
    <input list="dept" name="Departamento">
    <datalist id="dept">
        <option value="Atención al cliente"></option>
        <option value="Servicio técnico"></option>
    </datalist>
</form>
```

➔ **Función enviar**: lo que podemos hacer en este caso es comprobar el departamento al cuál se quiere enviar el mensaje, mediante el combobox, y en función de eso cambiar la dirección de correo del envío.

```
function enviar () {
    var comboDept = document.formContacto.Departamento

    if (comboDept.value == "Atención al cliente") {
        document.formContacto.action =
"mailto:atencion@dominio.com"
    } else {
        document.formContacto.action = "mailto:sat@dominio.com"
    }
    document.formContacto.submit()
}
```


*Aquí las líneas importantes son las que cambian la propiedad action del formulario, que es precisamente la que cambia el comportamiento del formulario, haciendo que se envíe el correo a uno u otro departamento.*

## Cuestionario

**Lee el enunciado e indica la opción correcta**:

¿Cuál de los siguientes no es un parámetro de addEventListener?

**a**. Nombre.

**b**. Tiempo.

**c**. Función.

**Lee el enunciado e indica la opción correcta**:

¿Con qué propiedad podemos saber la cantidad de campos de un formulario?

**a**. size.

**b**. name.

**c**. lentgth.

**Lee el enunciado e indica la opción correcta**:

¿Cuál es el lenguaje más común para cambiar la apariencia de un documento?

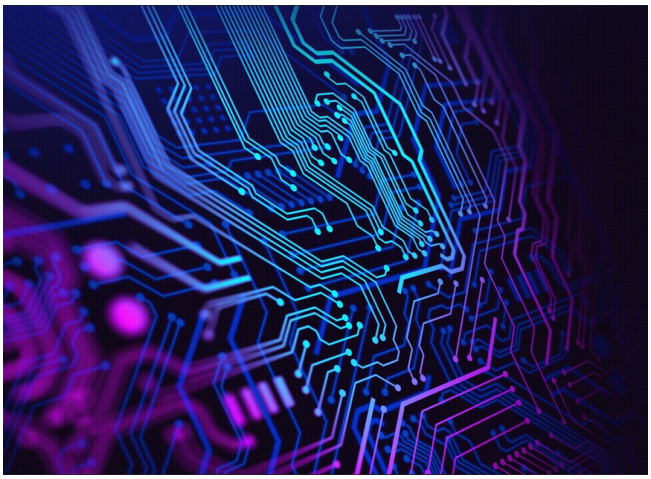
**a**. CSS.

**b**. JavaScript.

**c**. HTML.

# 6. VALIDACIÓN Y ENVÍO. PROCESAMIENTO DE DATOS DE UN FORMULARIO. FORMULARIOS DINÁMICOS

Hemos visto como acceder a los **formularios de HTML**, manejar sus **miembros**, **propiedades** y **métodos**. A partir de aquí podemos centrarnos en uno de los usos principales de **JavaScript** en los formularios, como es la validación.



Cuando un usuario rellena un formulario en una página web y acepta procesa, éste se envía directamente al **servidor**, que mediante algún **lenguaje de entorno servidor**, como puede ser PHP, por ejemplo, lo procesa, recogiendo los valores de los distintos campos.

DESTACADO

*La validación en JavaScript consiste en un paso previo al envío, que consiste en evitar errores en los datos, bien por falta de ellos o por inconsistencias.*

- **Principales errores**: los principales **errores** que pueden requerir de validación son:
  - ✔ Campos **vacíos** o en **blanco**.
  - ✔ Campos obligatorios **sin rellenar**.
  - ✔ Campos con **formato incorrecto**, por ejemplo, un NIF o un número de teléfono.
  - ✔ Campos que dependen directamente de otros.
- **Función**: el proceso de validación se encarga, mediante una función creada por el desarrollador de **avisar al usuario** de que debe corregir los **errores** antes de poder hacer el envío, de tal manera que los datos lleguen de forma correcta al servidor.

```
function validaFecha() {
    var dia = document.formulario.dia.value;
    var mes = document.formulario.mes.value;
    var ano = document.formulario.ano.value;

    var fecha = new Date(ano, mes, dia);
    if(!isNaN(fecha)){
        return false;
    )
    return true;
}
```

En el ejemplo anterior vemos cómo podemos validar una fecha, accediendo a los campos de día, mes y año, e intentando crear una fecha (Date) con éstos. Si la fecha es incorrecta, podremos avisar al usuario.

## Ejemplo: validación

Otro ejemplo de validación lo encontramos en el capítulo anterior, donde comprobamos si un campo de texto está vacío antes de realizar a acción del formulario.

Además, algo también muy útil que se puede hacer en un formulario mediante JavaScript, es hacerlo dinámico. Esto consiste principalmente en ir modificando el formulario a medida que se va rellenando.

Un ejemplo típico podría ser un selector de País, que afecta a otro en el cual podamos seleccionar las provincias, en función del primero:

```
function actualizarProvincias() {

    var comboPais = document.formulario.pais
    var comboProvincias = document.f1.provincias
```

```
    var pais = comboPais[comboPais.selectedIndex].value

    // Seleccionamos el array de provincias en función del país
    var mis_provincias = todasProvincias[pais]
    var num_provincias = mis_provincias.length

    comboProvincias.length = num_provincias

    // Introducimos en el select cada provincia del array
    for (i = 0; i < num_provincias; i++) {
        comboProvincias.options[i].value = mis_provincias[i]
        comboProvincias.options[i].text = mis_provincias[i]
    }

    comboProvincias.options[0].selected = true
}
```

## Actividad de aprendizaje 2: validación de un formulario

El siguiente punto del nuevo proyecto tiene que ver con los formularios, por lo que nos piden hacer una pequeña prueba. Se trata de hacer la validación de un formulario de contacto.

Nos piden:

- Crear un pequeño formulario, con nombre, dirección de correo y mensaje.
- Validar el formulario de tal manera que avise al usuario si cualquiera de los 3 campos está vacío.

Entrega el archivo .html en el espacio habilitado para ello.

# 7. EXPRESIONES REGULARES

Otra de las herramientas utilizadas en la validación de formularios, que acabamos de ver en la unidad anterior son las llamadas **expresiones regulares**.


*Una expresión regular es una cadena alfanumérica que conforma un patrón de búsqueda. Si este patrón se cumple, se procederá a hacer una serie de acciones.*

Normalmente, las **expresiones regulares** se usan para buscar cadenas de texto y reemplazarlas por otras cadenas, o, como ya hemos dicho, para validar estas cadenas.

A continuación, veremos algunos **ejemplos típicos de expresiones regulares** y cómo **validar el formato** de algunas cadenas:

- **Ejemplos de expresiones regulares**: ejemplos típicos podrían ser:
  - Comprobar que una **dirección** de correo contenga la arroba.
  - Comprobar que una **fecha** sea correcta.
  - Comprobar que un **NIF** tenga un formato correcto.

  Todo lo que se puede conseguir con las expresiones regulares se puede conseguir también mediante **código**, pero su uso produce código más eficiente y fácil de comprender.

- **Cadena I**: un ejemplo de una expresión regular para comprobar el **formato del NIF** sería la siguiente:

```
/^[0-9]{8}[a-z]$/i
```

Veamos esta expresión en profundidad:

- ✔ Las "/" sirven para **definir** la expresión regular.
- ✔ ^ - Que la cadena **empiece** por…
- ✔ [0-9] – Con **números** del 0 al 9.
- ✔ {8} – De 8 dígitos de **longitud**.
- ✔ [a-z] – Seguido de una **letra** de la A a la Z, en minúsculas.
- ✔ $ - **Final** de la cadena.
- ✔ i – Modifica el patrón para hacerlo **case-insensitive**, así validará el NIF tanto si está escrito en mayúsculas como en minúsculas.

Puedes ver todos los detalles de la composición de las expresiones regulares en los **recursos para ampliar**. A continuación, veremos como usarlas en JavaScript.

➔ **Cadena II**: en el siguiente ejemplo tenemos una sencilla página con una caja de texto, donde introducir una dirección de correo electrónico y un botón para validar que es correcta.

```
<html>
<body>
    <input id="email">
    <button>Validate</button>
    <script>
        document.getElementsByTagName( "button" )
[0].addEventListener("click", validateEmail, false);
        function validateEmail() {
            var regexp = /^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-
z\.]{2,6})$/i;

alert(regexp.test(document.getElementById("email").value));
        }
    </script>
</body>


</html>
```

Lo que hacemos es crear una **expresión regular** que contemple que una dirección de correo está formada por caracteres alfanuméricos (dirección), seguidos de una arroba, otra serie de caracteres (sitio), un punto y finalmente 2 o más caracteres (dominio).

Para validar esta expresión regular usamos el método "**test()**", que nos devolverá cierto o falso si el texto que cogemos de la dirección de correo cumple con esta expresión.


*Con estos sencillos pasos, la validación del formato de cualquier cadena se puede realizar de forma rápida y eficaz.*

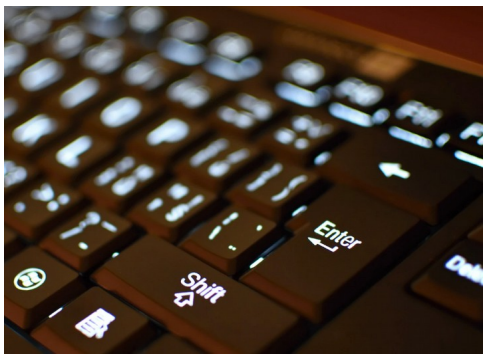
# 8. UTILIZACIÓN DE COOKIES. UTILIZACIÓN DE PARÁMETROS ENTRE DOCUMENTOS

En la unidad 3 hablamos vagamente acerca de las **cookies** y su utilización. Como se trata de una herramienta indispensable para la interacción con el usuario, haremos más hincapié durante los siguientes dos capítulos.


*El uso de cookies es una tecnología a través de la cual se guarda cierta información, enviada o recibida por el cliente, de forma local, de tal manera que queda almacenada en la máquina del cliente durante un cierto tiempo.*


*La idea es que una página pueda recordar cierta información del cliente cuando éste se vuelve a conectar a la página, de manera que se ahorra proceso y tiempo.*

A continuación, veremos algunos **aspectos relevantes** sobre las **cookies**:




- **Cadena de texto**: una cookie no es más que una cadena de texto, con varias parejas clave-valor, separadas por punto y coma.

```
<nombre>=<valor>; expires=<fecha>;
max-age=<segundos>; path=<ruta>; domain=<dominio>; secure;
httponly;
```

- **Enviar al servidor**: cuando se crea una cookie, ésta será enviada al servidor, con cada nueva petición de la página.
- **Almacenamiento**: una página puede almacenar tantas cookies como desea, y lo que es importante recalcar es que estas cookies siempre estarán almacenadas en la máquina del servidor.


*En el siguiente capítulo veremos su uso en JavaScript, como crearlas y recuperarlas, para su utilización en una siguiente visita a la página web por parte del cliente.*

## Cuestionario

**Lee el enunciado e indica la opción correcta**:

¿Qué es una cookie en el contexto de la interacción con el usuario?

**a.** Un archivo de imagen utilizado para mejorar la apariencia de la página.

**b.** Una tecnología que almacena información en la máquina del cliente.

**c.** Un código de programación utilizado para realizar cálculos en el servidor.

**Lee el enunciado e indica la opción correcta**:

¿Cómo se guarda la información en una cookie?

**a.** Como una lista de números separados por comas.

**b.** Como una cadena de texto con parejas clave-valor separadas por punto y coma.

**c.** Como un archivo adjunto en el servidor.

**Lee el enunciado e indica la opción correcta**:

¿Dónde se almacenan las cookies?

**a.** En la máquina del cliente.

**b.** En el servidor.

**c.** En una base de datos externa.

## 9. ESCRITURA Y LECTURA DE COOKIES

Desde JavaScript es posible tanto crear como recuperar **cookies**, que es lo que veremos a continuación.

Para trabajar con **cookies** dentro de **JavaScript** haremos uso de la propiedad cookie del objeto document. Así, por ejemplo, para crear una cookie podemos hacer:

```
document.cookie = "nombre=Jaime; max-age=3600";
document.cookie = "edad=42";
```


*En el ejemplo anterior estamos creando dos cookies diferentes, en las cuales guardaremos el nombre y la edad respectivamente. Además, en la primera le ponemos una fecha de caducidad, de tal manera que durará sólo 1 hora (3.600 segundos). Si no especificamos fecha de caducidad, como en la segunda, ésta solo es válida para la sesión actual.*

- Para **modificar el valor de una cookie** en cualquier momento, basta con sobreescribirla.

  ```
  document.cookie = "edad=42";
  ```

- Para **eliminar una cookie**, podemos poner una fecha de caducidad a 0. Por ejemplo:

  ```
  document.cookie = "nombre=; max-age=0";
  ```

- Para **recuperar las cookies** de una página, las almacenaremos en una variable de texto.

  ```
  var todasLasCookies = document.cookie;
  ```

  Obteniendo así un String con todas las cookies, con sus correspondientes parejas clave-valor.

  ```
  "cookie1=valor1;cookie2=valor2;cookie3=valor3[;...]"
  ```

➔ Si queremos **recorrer todas las cookies** que acabamos de leer, podemos hacerlo mediante el método split del String, que nos permite separar un texto, en subcadenas, indicando el carácter separador, en este caso el punto y coma.

A partir de ahí, le daremos el uso requerido a las cookies, igual que haríamos en cualquier otra página web.

*Vídeo: cookies*


*Visualiza este vídeo en el que hablamos sobre las Cookies.*

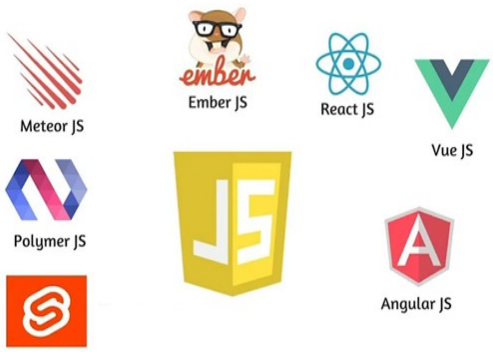
# 10. USO DE FRAMEWORKS. DESARROLLO RÁPIDO DE APLICACIONES

El creciente uso de JavaScript durante los últimos años ha proporcionado que muchos desarrolladores creen herramientas para que la creación de aplicaciones sea mucho más rápida y directa.

En una unidad anterior vimos las **librerías de terceros**, y lo que suponían en ayuda al desarrollo. En este capítulo vamos a ir un paso más allá, viendo lo que se denominan **frameworks**.


*Podemos encontrar muchas definiciones acerca de lo que es un framework, pero diremos que no es más que un conjunto de librerías, plantillas y código que ayudan al desarrollador a crear contenido de forma mucho más rápida, eficiente y sencilla.*

Además, en los **frameworks** más populares podemos encontrar multitud de información, tutoriales y foros, por lo que su aprendizaje no será complicado.





*La elección de un framework en concreto dependerá en gran medida de las necesidades del proyecto en que nos encontramos. Pese a eso, hay tal variedad actualmente que no será difícil encontrar el que se adecúe más en cada momento.*

Los principales **frameworks** de **JavaScript** actualmente son:

- **Angular**: creado por **Google** en 2009. Es uno de los frameworks de **código abierto** más utilizados actualmente. Ideal para la creación de sitios web de manera sencilla.

  Utiliza TypeScript como lenguaje de desarrollo, lo cual lo hace muy potente, ya que se adapta de manera fácil a los lenguajes tradicionales.

- **React.js**: algunos lo consideran una **librería** en lugar de un framework, sin embargo, también es de los más utilizados, por ejemplo, por Facebook o Instagram.

  Permite crear **webs dinámicas, estables y escalables**, con un gran rendimiento. Además, como hemos dicho, la comunidad de desarrolladores que apoyan al framework hace más sencillo su uso.

- **Vue.js**: creado en 2013, no está creado por una gran empresa, si no que depende de donaciones individuales y corporativas.

  Es de los frameworks más **sencillo** de aprender, similar a React.js y Angular en muchos aspectos.

*Vídeo: frameworks*


*Visualiza el siguiente vídeo sobre los frameworks de JavaScript.*
*https://www.youtube.com/embed/p9OpBxpSM5c*

Actividad de aprendizaje 3: denominar expresiones regulares

Otro punto del proyecto semi-secreto en que nos encontramos es el uso de expresiones regulares. Por eso quieren que las dominemos por completo antes de poder implementar la aplicación definitiva.

Se nos pide:

- Crear dos cadenas de texto que cumplan con la siguiente expresión regular:

```
/^\d+\s[\w\s]+\w{4,6},\s[\w\s]+,\s\w{2}\s\d{5}$/
```

- Las dos cadenas han de ser diferente entre sí. Juega con las longitudes máximas y mínimas y con la combinación de caracteres.
- Por ejemplo, la cadena 56 aa YTBNO 09875468890 r2 no sería válida.

Entrega un archivo .pdf en el espacio habilitado para ello.

## Cuestionario

**Lee el enunciado e indica la opción correcta**:

¿Qué método se usa para enviar un formulario?

**a**. send.

**b**. submit.

**c**. reset.

**Lee el enunciado e indica la opción correcta**:

¿En una expresión regular como indicamos inicio y fin?

**a**. [ ]

**b**. { }

**c**. / /

**Lee el enunciado e indica la opción correcta**:

¿Cuál de los siguientes frameworks está creado por Google?

**a**. Angular.

**b**. React.js.

**c**. Vue.js.

## *RESUMEN*

Hemos centrado toda la unidad en el **proceso de interacción** con el usuario de la página web.

Empezamos definiendo el concepto de **evento**, y su utilización en **JavaScript**, para controlar las acciones que realiza el usuario y actuar en consecuencia.

Después entramos en la parte esencial de las aplicaciones web, como son los **formularios**. Éstos se crean en **HTML** y desde **JavaScript** somos capaces de modificar su **apariencia**, **comportamiento**, y lo que es más importante, **validar su contenido**.

Otra herramienta que nos ha servido para **validar parte de formularios**, sobre todo en cuanto a formatos son las **expresiones regulares**, de las cuales también hemos visto su uso en JavaScript.

Hemos ampliado el concepto de **cookies**, que vimos en una unidad anterior, viendo cómo se crean, leen y escriben, y sus principales usos dentro de una página.

Por último, hemos conocido unas herramientas que facilitan la tarea del desarrollador, como son los **frameworks**, viendo cuáles son los más utilizados.

## *RECURSOS PARA AMPLIAR*

### PÁGINAS WEB

- Event - Web APIs | MDN: *https://developer.mozilla.org/en-US/docs/Web/API/Event* [Consulta julio 2022].
- Event Bubbling and Capturing in JavaScript – javatpoint: *https://www.javatpoint.com/event-bubbling-and-capturing-in-javascript* [Consulta julio 2022].
- Expresiones Regulares - JavaScript | MDN: *https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions* [Consulta julio 2022].
- HTML DOM Event Objects: *https://www.w3schools.com/jsref/obj_events.asp* [Consulta julio 2022].

# BIBLIOGRAFÍA

## PÁGINAS WEB


- JSON - JavaScript | MDN: *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON* [Consulta julio 2022].
- W3C: *https://www.w3.org/* [Consulta julio 2022].


## GLOSARIO

- **Case-sensitive**: se dice cuando en un lenguaje de programación se distingue una palabra escrita en mayúsculas o en minúsculas.
- **DOM**: el modelo de objeto de documento es una representación completamente orientada al objeto de la página web y puede ser modificado con un lenguaje de script.
- **HTML5**: estándar que sirve como referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones.
- **jQuery**: librería de terceros para la ayuda del desarrollo en JavaScript.