DESARROLLO WEB EN ENTORNO CLIENTE

UT5.- INTERACCIÓN CON EL USUARIO. EVENTOS Y FORMULARIOS

CURSO 18/19

Profesora: Teresa Jiménez Nieto

Objetivos

- Reconocer las posibilidades de los lenguajes de marcas de capturar y gestionar los eventos producidos.
- Diferenciar los tipos de eventos que se pueden manejar.
- Crear código que capture y utilice eventos.
- Gestionar formularios web.
- Validar formularios web con eventos y expresiones regulares.
- Utilizar cookies y web storage.

1. MODELO DE GESTIÓN DE EVENTOS

- **EVENTO:**
  - ***Mecanismo que se acciona cuando el usuario realiza un cambio sobre la página web*** (clic con el ratón, pulsar una tecla, recargar la página, pulsar un botón de formulario, etc.)
  - Capturar un evento es programar una acción para que se realice una tarea.
  - El encargado de gestionar los eventos es el DOM (Document Object Model).
- **MANEJADOR**:
  - Palabra reservada que indica la acción que se va a manejar.
  - Por ejemplo, en el evento click el manejador es onClick.
- Eventos HTML:
  https://www.w3schools.com/tags/ref_eventattributes.asp

1. MODELO DE GESTIÓN DE EVENTOS

- **DOM**: Estándar que define como acceder a documentos HTML o XML.
  - Mediante JavaScript podemos acceder y actualizar el contenido dinámicamente.
  - El DOM crea la jerarquía de objetos que compone una página web.
  - Un documento HTML se estructura en forma de árbol invertido, siendo la raíz <html>, los nodos hijos <head> y <body>, …
- **Los eventos tienen un manejador asociado a un código JavaScript.**

1. MODELO DE GESTIÓN DE EVENTOS

- **Grupos de eventos de la especificación DOM:**
  - **Eventos del ratón.** Cuando el usuario hace uso del ratón para realizar una acción. (movimiento o pulsación del ratón), puede desencadenar un evento.
  - **Eventos del teclado.** Se originan cuando el usuario pulsa alguna tecla del teclado.
  - **Eventos HTML.** Se producen cuando hay algún cambio en la página del navegador. También pueden ocurrir cuando existe alguna interacción entre el cliente y el servidor.
  - **Eventos DOM,** o eventos de mutación, son los que se originan cuando existe algún cambio en la estructura DOM de la página.

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos del ratón:**
  - ***onclick***: Botón izquierdo del ratón.
  - ***ondblclick***: Doble clic.
  - ***onmousedown/onmouseup***: Pulsar/soltar un botón del ratón
  - ***onmouseover/onmouseout***. El puntero del ratón entra/sale fuera de un elemento
  - ***onmousemove***. Cuando se mueve el puntero dentro de un elemento

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos del ratón:**
  - El orden de ejecución de los eventos es mousedown, mouseup, click, mousedown, mouseup, click, dblclick.
  - Cuando se produce un evento **el objeto event se crea automáticamente.**
  - Para los eventos de ratón el objeto event tiene entre otras las siguientes **propiedades**
    - las coordenadas del ratón (screenX, screenY),
    - nombre del evento (type)
    - elemento que origina el evento (button)

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos del teclado:**
  - ***onkeydown*:** Cuando se pulsa o mantiene pulsada una tecla.
  - ***onkeypress*:** Cuando se pulsa o mantiene pulsada una tecla alfanumérica.
  - ***onkeyup*.:** Cuando se libera una tecla pulsada
  - Las propiedades de event para los eventos de teclado son:
    - El código numérico de la tecla pulsada (keyCode),
    - El código unicode del carácter correspondiente a la tecla pulsada (charCode),
    - El elemento que origina el evento (target),
    - Otras para identificar si hemos pulsado shift (shif tKey),control (ctrlKey), alt (altKey) o meta (metaKey).

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos del teclado:**
  - Cuando pulsamos una tecla que corresponda a un carácter alfanumérico, la secuencia de eventos es la siguiente: keydown, keypress, keyup.
  - En el caso de pulsar una tecla que no corresponda a un carácter alfanumérico, la secuencia de eventos es: keydown, keyup.
  - Además existe la posibilidad de que dejemos una tecla pulsada.
    - Si es con carácter alfanumerico, se repiten de forma continua los eventos keydown, keypress.
    - En caso de que no sea alfanumerico se repite de forma continuada el evento keydown solamente.

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos del DOM:**
  - Estos eventos hacen referencia a la especificación DOM
  - Se accionan cuando varía el árbol DOM.
  - ***DOMSubtreeModified***: cuando añadimos o eliminamos nodos en un subárbol de un elemento o documento.
  - **DOMNodeInserted**: cuando añadimos un nodo hijo a un nodo padre.
  - ***DOMNodeRemoved***: cuando eliminamos un nodo que tiene nodo padre.
  - ***DOMNodeRemovedFromDocument***: cuando eliminamos un nodo del documento.
  - ***DOMNodeInsertedIntoDocumento***: cuando añadimos un nodo al documento.

1. MODELO DE GESTIÓN DE EVENTOS

- **Manejadores de eventos HTML:**
  - **onload**: la página/elemento termina de cargarse.
  - **onabort**: cuando el usuario detiene la descarga de un elemento antes de que haya terminado.
  - **onerror:** cuando ocurre un error al cargar un elemento.
  - **onunload:** cuando el navegador cierra el documento.
  - **onresize:** la ventana del navegador cambia de tamaño.
  - **onblur:** cuando se abandona un campo de formulario.
  - **onchange:** cambia el contenido un campo de formulario.
  - **onfocus:** un elemento button, input, label, select, textarea o body obtiene el foco.
  - **onselect:** cuando un campo de texto es seleccionado.
  - **onsubmit:** cuando se pulsa el botón enviar de un formulario.
  - **onscroll:** cuando varía la posición del scroll.
  - **onreset:** cuando se pulsa el botón reset en un formulario.

1. MODELO DE GESTIÓN DE EVENTOS

1.1. Modelo de registro de eventos en línea

- Cada elemento XHTML tiene sus posibles eventos como atributos: puede tener un evento de cada tipo.
- El nombre del evento es "on" seguido del nombre de la acción.
- **Manejadores como atributo de una etiqueta XHMTL:**

```
<h3 id="cab1" onclick="this.innerHTML='Javascript en XHTML'"
onmouseover="this.style.background='red'"
onmouseout="this.style.background='white'">Pulsa aquí para ver lo
que se ejecuta</h3>
<!-- h3 id="cab1"
onclick="document.getElementById("cab1").innerHTML="...">...</h3>
-->
```

1. MODELO DE GESTIÓN DE EVENTOS

1.1. Modelo de registro de eventos en línea

- **Manejadores como funciones externas:**

```
<h3 onclick="cambiar(this)">Pulsa aquí para ver qué se
ejecuta</h3>
<script>
    function cambiar(elem) {
        elem.innerHTML = "Javascript en XHTML y función externa";
    }
</script>
```

1. MODELO DE GESTIÓN DE EVENTOS

1.1. Modelo de registro de eventos en línea

- Hay acciones que desencadenan varios eventos. Ej. submit desencadena onmousedown, onclick, onmouseup, onsubmit.
- Para evitar que el navegador ejecute la acción por defecto necesitamos añadir "return false;"

```
<a href="http://www.google.com" onclick="alertar(); return
false;">Pulsa aquí para ver qué se ejecuta</a>
<script>
    function alertar() {
        alert("Vamos a Google");
    }
</script>
```

1. MODELO DE GESTIÓN DE EVENTOS

1.1. Modelo de registro de eventos en línea

- Podemos pedir al usuario si quiere que se ejecute esa acción por defecto.

```
<a href="http://www.google.com" onclick="return
preguntar();">Pulsa aquí para ver qué se ejecuta</a>
<script>
    function preguntar() {
        return confirm("¿Deseas ir a Google?");
    }
</script>
```

- **Evento "onload":** nos permite indicarle a la página que no cargue ningún JavaScript hasta que no haya cargado todo el HTML.

```
<body onload="alert('La página se ha cargado correctamente')">
```

1. MODELO DE GESTIÓN DE EVENTOS

1.1. Modelo de registro de eventos en línea

- No se recomienda utilizarlo ya que estamos mezclando etiquetas HTML con código JavaScript.
- No separa la programación de la estructura.
- Lo recomendado es que no haya en un documento HMTL código JavaScript, sino que se incluya en archivos externos.

1. MODELO DE GESTIÓN DE EVENTOS

1.2. Modelo de registro de eventos tradicional

- Separa el código HTML del JavaScript.
- El evento se convierte en una propiedad del elemento.
- Tampoco se recomienda utilizarlo.

```
<h1>Modelo de registro de eventos tradicional</h1>
<h3 id="tradicional">Pulsa aquí para ver lo que se ejecuta</h3>
<script>
    document.getElementById("tradicional").onclick = cambiar; //¡¡Sin paréntesis!!
    function cambiar() {
        alert("Entramos en cambiar");
        document.getElementById("tradicional").innerHTML = "Modelo de registro de eventos
        tradicional";
    }
</script>
```

1. MODELO DE GESTIÓN DE EVENTOS

1.2. Modelo de registro de eventos tradicional

- Si en el ejemplo anterior queremos que el elemento html deje de tener el evento onclick activado:

```
<h1>Modelo de registro de eventos tradicional</h1>
<h3 id="tradicional">Pulsa aquí para ver lo que se ejecuta</h3>
<script>
    document.getElementById("tradicional").onclick = cambiar; //¡¡Sin paréntesis!!
    function cambiar() {
        alert("Entramos en cambiar");
        document.getElementById("tradicional").innerHTML = "Modelo de registro de eventos
        tradicional";
        document.getElementById("tradicional").onclick = null;
    }
</script>
```

1. MODELO DE GESTIÓN DE EVENTOS

1.2. Modelo de registro de eventos tradicional

- La página debe estar completamente cargada para que se utilicen las funciones del DOM que asignan manejadores a los elementos HTML.

```
<h3 id="tradicional2">Pulsa aquí para ver qué se ejecuta</h3>
<script>
    window.onload = function () {
        alert("La página ha cargado correctamente");
        document.getElementById("tradicional2").onclick = miMensaje; //¡¡Sin paréntesis!!
    }
    function miMensaje() {
        document.getElementById("tradicional2").innerHTML = "Modelo de registro de eventos
        tradicional";
    }
</script>
```

1. MODELO DE GESTIÓN DE EVENTOS

1.3. Modelo de registro de eventos avanzado del W3C

- Es uno de los más extendidos.
- Ya es soportado por todos los navegadores modernos (incluido Microsoft Edge).
- No funciona en versiones antiguas de navegadores de IE.
- Si queremos que funcione en navegadores con versiones antiguas, hay que utilizar librerías *CrossBrowser*.
- Separa HTML de JavaScript.
- Podemos añadir todas las acciones o manejadores que queramos para cada evento.

1. MODELO DE GESTIÓN DE EVENTOS

1.3. Modelo de registro de eventos avanzado del W3C

elemento.addEventListener("<evento_sin_on>", <función>, <true | false>);

- Utiliza tres argumentos:
  - Tipo de evento (sin "on" delante).
  - Función a ejecutar (sin comillas ni paréntesis).
  - Valor booleano para elegir la fase de captura (true) o burbujeo (false).

1. MODELO DE GESTIÓN DE EVENTOS

1.3. Modelo de registro de eventos avanzado del W3C


```
<h1>Modelo de eventos avanzados del W3C</h1>
<h3 id="w3c">Modelo del W3C</h3>
<h3 id="w3canonima">Modelo del W3C con funciones anónimas</h3>

<script>
    document.getElementById("w3c").addEventListener("click", saludarUnaVez, false);
    document.getElementById("w3c").addEventListener("click", colorearse, false);
    document.getElementById("w3c").addEventListener("mouseover", fondo, false);
    function saludarUnaVez() {
        alert("¡Hola, caracola!");
        document.getElementById("w3c").removeEventListener("click", saludarUnaVez); //para
        que sólo se ejecute la primera vez.
    }
    function colorearse() {
        document.getElementById("w3c").style.color = "red";
    }
    function fondo() {
        document.getElementById("w3c").style.background = "blue";
    }
```

1. MODELO DE GESTIÓN DE EVENTOS

1.3. Modelo de registro de eventos avanzado del W3C

- **Crear un evento anónimo (no asociado a ninguna función:**

elemento.addEventListener("<evento_sin_on>",function(){
<codigo_funcion}),false);

```
<h1>Modelo de eventos avanzados del W3C</h1>
<h3 id="w3canonima">Modelo del W3C con funciones anónimas</h3>
<script>
    document.getElementById("w3canonima").addEventListener("click", function () {
        this.style.background = "#C0C0C0";
    });
</script>
```

Estilos con style:
https://www.w3schools.com/jsref/dom_obj_style.asp

1. MODELO DE GESTIÓN DE EVENTOS

1.4. Modelo de registro de eventos avanzado de Microsoft

- Utiliza un evento ***attachEvent()*** con dos argumentos:
  - Evento entre comillas. (Sí se utiliza "on" delante).
  - Función a ejecutar sin paréntesis.

  elemento.attachEvent("<evento_con_on>",funcion);

- ***Eliminar un evento de un elemento*:**
  - elemento.detachEvent("onclick", accion);
- Es posible crear más de un evento para el mismo elemento.

1. MODELO DE GESTIÓN DE EVENTOS

1.4. Modelo de registro de eventos avanzado de Microsoft


```
<h1>Modelo de eventos avanzados de Microsoft</h1>
<h3 id="ms">Modelo de Microsoft</h3>
<h3 id="msanonima">Modelo de Microsoft</h3>
<script>
    document.getElementById("ms").attachEvent("onclick", saludarUnaVez);
    document.getElementById("ms").attachEvent("onclick", colorearse);
    document.getElementById("ms").attachEvent("onmouseover", fondo);
    function saludaUnaVez() {
        alert("¡Hola, caracola!");
        document.getElementById("ms").detachEvent("onclick", saludarUnaVez);
    }
    function colorearse() {
        document.getElementById("ms").style.color = "red";
    }
    function fondo() {
        document.getElementById("ms").style.backgroundColor = "blue";
    }
```

1. MODELO DE GESTIÓN DE EVENTOS

1.4. Modelo de registro de eventos avanzado de Microsoft

- **Crear un evento anónimo (no asociado a ninguna función:**

  elemento.attachEvent("<evento_con_on>",function(){...});

```
documnet.getElementById("msanonima").attachEvent("onclick", function () {
    this.style.backgroundColor = "#C0C0C0";
});
```

1. MODELO DE GESTIÓN DE EVENTOS

1.5. Obtención de información de un evento

*Utilizando el modelo de eventos del W3C

- Automáticamente cuando se produce un evento, el navegador crea un objeto de tipo evento.
- El objeto evento almacena el evento que se ha producido.
- En los navegadores de Microsoft (versiones anteriores), capturamos el evento con window.event.

1. MODELO DE GESTIÓN DE EVENTOS

1.5. Obtención de información de un evento

- **evento.type** nos devuelve el tipo de evento que se ha generado.

```
<h1 id="eventos">Obtener información de un evento</h1>
<h2 id="parrafo1">Párrafo 1</h2>
<script>
    document.getElementById("eventos").addEventListener("mouseover", manejador);
    document.getElementById("eventos").addEventListener("mouseout", manejador);
    function manejador(e) {
        //Valoramos la posibilidad de que se utilice un navegador de Microsoft
        if (!e) e = window.event;
        switch (e.type) {
        case "mouseover":
            this.style.color = "purple";
            break;
        case "mouseout":
            this.style.color = "yellow";
            break;
        }
    }
```

1. MODELO DE GESTIÓN DE EVENTOS

1.5. Obtención de información de un evento

- **evento.target** nos permite extraer el elemento que ha generado el evento.

```
<h1 id="eventos">Obtener información de un evento</h1>
<h2 id="parrafo1">Párrafo 1</h2>
<h2 id="parrafo2">Párrafo 2</h2>
<script>
    document.getElementById("eventos").addEventListener("mouseover", manejador);
    document.getElementById("eventos").addEventListener("mouseout", manejador);
    document.getElementById("parrafo1").addEventListener("click", saludo);
    document.getElementById("parrafo2").addEventListener("click", saludo);
    function saludo(e) {
        //Valoramos la posibilidad de que se utilice un navegador de Microsoft
        if (!e) e = window.event;
        if (e.target.id == "parrafo1") alert("Has pulsado el primer párrafo");
        else if (e.target.id == "parrafo2") alert("Has pulsado el segundo párrafo");
        alert("Has pulsado el " + e.target.id);
    }
```

2.1. Elementos de un formulario

- **form**:

```
<form action="pagina.php" method= "post">
   //elementos
</form>
```

  - **action**: url de la página a la que redirige el formulario.
  - **method**: puede ser POST o GET, dependiendo del enmascaramiento de los datos enviados.

2. UTILIZACIÓN DE FORMULARIOS DESDE CÓDIGO

2.1. Elementos de un formulario

- **<form>**:
  - ***Propiedades***:
    - https://www.w3schools.com/jsref/dom_obj_form.asp
  - ***Métodos***:
    - reset(): resetea un formulario.
    - submit(): envía un formulario.
  - ***form.elements[]:*** devuelve un array con todos los input de un formulario.

    Ej. var x = document.getElementById("myForm").elements[0].value;

    https://www.w3schools.com/jsref/coll_form_elements.asp

2. UTILIZACIÓN DE FORMULARIOS DESDE CÓDIGO

2.1. Elementos de un formulario

- **<input>:**
  - **Propiedades**
    - **type:**
      - text: cuadro de texto.
      - password: cuadro de contraseña.
      - checkbox: casilla de verificación.
      - radio: opción de entre dos o más.
      - submit: botón de envío de formulario.
      - reset: botón de vaciado de campos.
      - file: botón para buscar fichero.
      - hidden: campo oculto.
      - image: botón de imagen en el formulario.
      - button: botón del formulario.
    - **name**: asigna un nombre al elemento. Necesario para que el servidor pueda trabajar con él .
    - **value**: inicializa el valor del elemento.

https://www.w3schools.com/jsref/
HTML Objets

2. UTILIZACIÓN DE FORMULARIOS DESDE CÓDIGO

2.1. Elementos de un formulario

- **<input>:**
- Con HTML5 se introdujeron nuevos tipos pero no son soportados por todos los navegadores
  - color Debería mostrar una rueda de color
  - date Debería mostrar un selector de fecha
  - datetime-local
  - email Debería validar automáticamente al enviarlo
  - month Debería mostrar un selector de fecha
  - number Permite restricciones
  - range Define un control tipo slider
  - search Para campos de búsqueda
  - tel Debería validar automáticamente al enviarlo
  - time Debería mostrar un selector de hora
  - url Debería validar automáticamente al enviarlo
  - week Debería mostrar un selector de fecha

2.1. Elementos de un formulario

- **<input>:**
  - **Propiedades:**
    - size: tamaño inicial. En campos text y password se refiere al número de caracteres.
    - maxlenght: nº máximo de caracteres que pueden conteer text y password.
    - checked: para elementos chexkbox y radio. indica si el checkbox está marcado, o qué botón de radio está pulsado.
    - disabled: el elemento aparece deshabilitado. el dato no se envía al servidor.
    - readonly: bloquea el contenido de control.
    - src: asigna la url de una imagen que se colocará como botón.
    - alt: descripción del elemento que se muestra al poner el cursor encima.
    - required: indica si el campo debe ser completado antes de enviar el formulario.
  - **Métodos:**
    - select(): selecciona el contenido de un campo de textarea.

2.1. Elementos de un formulario

- **<label>:**
  - La propiedad **for** en HTML permite indicar el elemento del formulario al que está asociada.
  - **Propiedad htmlFor:** devuelve o modifica el valor del atributo for.
  - **Propiedad form:** devuelve la referencia al formulario que contiene la etiqueta.

```
<form id="miForm">
    <label id="label1" for="nombre">Nombre:</label>
    <input type="text" name="nombre" id="nombre1"><br>
</form>

<script>
    document.getElementById("label1").htmlFor = "nombre1";
    alert(document.getElementById("label1").form.id);
</script>
```

2.1. Elementos de un formulario

- **<textarea>:**

  https://www.w3schools.com/jsref/dom_obj_textarea.asp

  - **Propiedades:**
    - name: asigna o devuelve el atributo name.
    - rows: número de líneas.
    - cols: ancho.
    - maxLenght: número máximo de caracteres.
    - disabled: elemento deshabilitado.
    - readOnly: bloquea el contenido.
    - required: si el campo debe ser completado antes de enviar el formulario.

2.1. Elementos de un formulario

- **<select> y <option>:**
  - Dentro del <select> tenemos la etiqueta <option> con el campo value para almacenar el valor y en su interior lo que se visualizará.
  - **Propiedades:**
    - name: asigna un nombre. necesario para que el servidor.
    - disabled: elemento deshabilitado. No se envía al servidor.
    - multiple: permite marcar más de una fila visible. Requiere del atributo size.
    - size: si multiple está marcado, muestra cuántas filas estarán visibles.
    - required: si el campo debe ser completado antes de enviar el formulario.

2.1. Elementos de un formulario

Desplegable. Ejemplo:

```
<p>Comida: </p>
<select name="comida">
    <option value="pasta">Pasta</option>
    <option value="carne">Carne</option>
    <option value="pescado">Pescado</option>
    <option value="postre">Postre</option>
</select>
```

Desplegable con elección múltiple. Ejemplo:

```
<p>Comida: </p>
<select multiple size="3" name="comida[]">
    <option value="pasta">Pasta</option>
    <option value="carne">Carne</option>
    <option value="pescado">Pescado</option>
    <option value="postre">Postre</option>
</select>
```

2.1. Elementos de un formulario

- **<fieldset> y <legend>**
  - fieldset agrupa elementos del formulario mediante una línea y permite asignar un título con la etiqueta.
  - **Propiedades:**
    - name: (fieldset) asigna un nombre al elemento.
    - align: (legend) permite elegir la alineación del texto del legend sobre el fieldset.

```
<form>
 <fieldset>
  <legend>Datos:</legend>
  Nombre: <input type="text" size="30"><br>
  Email: <input type="text" size="30"><br>
 </fieldset>
</form>
```

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Seleccionar un formulario**:

  Ej. <form id="miForm" name="miForm" action="enviar.php"…>

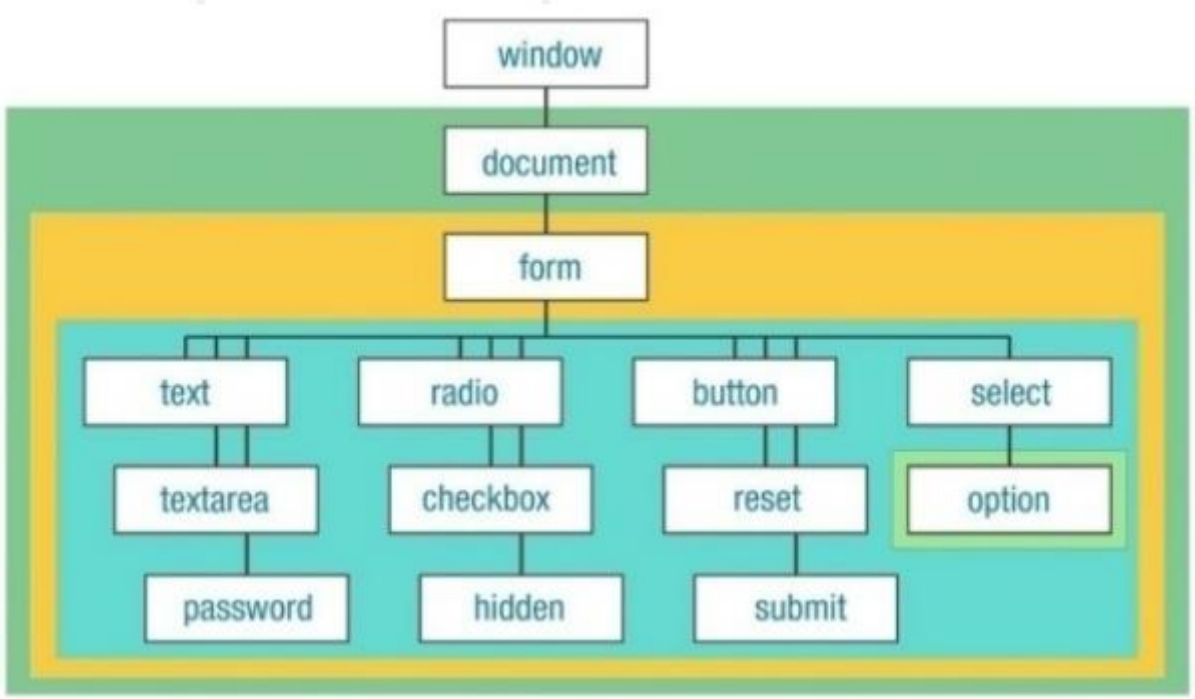
  - **Si conocermos el id:**
    - **document.getElementByID("id")**
      - var formulario = document.getElementById("miForm");
    - **document.forms["id"]**
      - var formularios = document.forms;
      - var primerForm = formularios["miForm"];
      - O var primerForm = document.forms["miForm"];

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Seleccionar un formulario:**
  - **Si conocemos el nº de formulario que es en la página:**
    - **getElementByTagName("tag")[posicion]**
      - var formularios = document.getElementByTagNme("form");
      - var primerForm = fomularios[0];
      - O var primerForm = document.getElementsByTagName("form")[0];
    - **document.forms[posicion]**
      - var formularios = document.forms; //devuelve un array de formularios
      - var primerForm = formularios[0];
      - O var primerForm = document.forms[0];

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **El formulario como contenedor de objetos**



3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Selección de objetos de un formulario:**

Formulario.elements[]
devuelve un array con todos los input del formulario

getElementById()
devuelve un elemento que tenga un id determinado

getElementsByTagName
devuelve un array de elementos de un tipo de etiqueta

getElemensByName()
Devuelve un array con elementos que tienen el mismo nombre (por ejemplo, radiobutton)

* Hay más maneras que veremos en el DOM.

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Selección de objetos de un formulario:**
  - Ej:

```
<form id="formBusqueda" action="buscar.php">
    <input type="text" id="entrada" name="cEntrada">
    <input type="submit" id="enviar" name="enviar"
    value="Buscar">
</form>
```

  - Para hacer referencia al input texto podemos hacerlo mediante:

```
document.forms[0].elements[0];
document.getElementsByTagName("input")[0];
document.getElementsByName("cEntrada")
document.forms["formBusqueda"].elements["cEntrada"];
document.getElementById("entrada");
```

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Objetos input de tipo texto:**
  - Es recomendable usar un id igual que el name para poder referirnos a ellos con getElementById.
  - Podemos referirnos a cualquiera de sus propiedades para cambiarlas.

```
<form id="formBusqueda" action="buscar.php">
    <input type="text" id="entrada" name="entrada">
    <input type="submit" id="enviar" name="enviar"
    value="Buscar">
</form>
<script>
    document.getElementById("entrada").value="Probando
    value";
    document.getElementById("enviar").style.border =
    "2px solid blue";
</script>
```

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Objetos input de tipo checkbox:**
  - La propiedad value es el texto asociado al objeto.
  - La propiedad checked devuelve true si está marcado y false en caso contrario. Ej. podemos hacer que un formulario sólo se envíe si el checkbox está marcado.

```
<form id="formBusqueda" action="buscar.php">
    <input type="text" id="entrada" name="entrada">
    <input type="checkbox" id="cantidad" name="cantidad"
    value="100">
    <input type="submit" id="enviar" name="enviar"
    value="Buscar">
</form>
<script>
    document.getElementById("cantidad").value="200";
    document.getElementById("cantidad").checked = true;
</script>
```

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Objetos input de tipo radio:**
  - Cuando utilizamos varios radiobutton con el mismo nombre, se crea un array de objetos.
  - Podemos consultar el número de ratio buttom con formulario.gruporadio.lenght
  - Podemos consultar si un radio button está marcado con formulario.gruporadio[num].checked y devolverá true o false.

3. MODIFICACIÓN DE APARIENCIA Y COMPORTAMIENTO

- **Objetos select:**
  - La propiedad selectedIndex devuelve el índice de la opción que ha sido seleccionada.
  - Cada opción tiene dos propiedades accesibles: text y value que indican el valor visible y el valor interno, respectivamente.

```
<option value="ZA">Zamora</option>
formulario.nombreCampoSelect.options[n].text; //Devuelve Zamora
formulario.nombreCampoSelect.options[n].value; //Devuelve ZA
```