

# DOM y formularios

Laura Isabel López Naves

# Formularios

- **forms[ ]**: es un array que contiene los formularios del documento. Aunque hay muchas formas de acceder a un formulario, las más sencillas y estándar es utilizar:
  - **“document.forms[i]”**
  - **“document.getElementById(“x”)”** siendo “x” el atributo “id”.
  - **“document.querySelector(“x”)”** siendo “x” el selector css.
- **elements[ ]**: array de controles (**input**, **textarea** o **select**) de cada formulario.
- Además, se puede acceder a cada control:  
(ej: suponiendo un formulario “foo” y `<input type=“text” name=“email”>`) (\*):
  - Con un índice o nombre del campo (**foo.elements[i]** o **foo.elements[‘email’]**)
  - Con un índice o nombre de campo sobre el formulario (**foo[i]** o **foo[‘email’]**)

(\*): Por supuesto, también se puede asignar un id a cada elemento y buscar con `querySelector()` o `getElementById()`.

# Formularios

- Los elementos o controles típicos de un formulario son:
  - **input**: su tipo (“type”) puede ser: “text” (caja de texto), “button” (botón), “checkbox” (opción múltiple), “radio” (opción múltiple exclusiva), “submit” (botón para enviar el formulario), etc...
  - **textarea**: para textos de varias líneas (se puede especificar su tamaño con los atributos “cols” y “rows”)
  - **select**: para listas desplegables (etiqueta “select” con subetiquetas “option”).
- Cada elemento tiene las propiedades como:
  - **type**: en los elementos input indica el tipo de control (text, checkbox, button, etc.).
  - **name**: nombre del campo (representa la variable que se enviará al servidor al hacer un “submit” en el formulario).
  - **value**: valor del atributo value (o del texto escrito por el usuario).
  - **checked**: valor boolean que indica un elemento seleccionado (radiobutton y checkbox).
  - **selected**: atributo que indica la opción seleccionada en un control option.

# Formulario. Ejemplo:

```
<body>
  <form id="cliente" action="#">
    <label>Nombre: <input name="nombre" value="Pedro" type="text"></label>
    <label>Telefono: <input name="telefono" value="687232332" type="text"></label>
    <label>Datos: <textarea name="datos" cols="30" rows="10">Lorem ipsum</textarea></label>
    <input type="submit" value="Enviar">
  </form>
  <script>
    const fo = document.querySelector('#cliente');
    console.log(fo['nombre'].value); // Pedro
    console.log(fo[0].value); // Pedro
    console.log(fo.elements['telefono'].value); // 687232332
    console.log(fo.elements[1].value); // 687232332
    let res = "";
    for (const e of fo) {
      res += e.value + " ";
    }
    console.log(res); // Pedro 687232332 Lorem ipsum Enviar
  </script>
```

# Eventos en formularios

- **focus:** se produce cuando un elemento obtiene el foco.
- **blur:** se produce cuando un elemento pierde el foco.
- **change:** se produce cuando el usuario pierde el foco de un elemento “input” y ha modificado el campo. En elementos de tipo “select” o “input type=checkbox/radio” se produce cuando se selecciona una opción.
- **input:** se produce cuando cambia el contenido de cualquier elemento de tipo texto.
- **cut/copy/paste:** cuando se producen las acciones de cortar, copiar y pegar.
- **click:** cuando se hace click en cualquier elemento.
- **submit:** se produce cuando se envía el formulario, es decir, cuando se hace click en el botón “submit” o cuando se pulsa return sobre un elemento “input”. Antes de producirse el evento “submit” sobre el formulario siempre se produce el evento “click” sobre el botón “submit”.

# Enviar un formulario

- Cuando se envía un formulario se envían los datos de los controles del formulario. Cada control con atributo “name” es una variable a enviar.
- El atributo “action” del formulario indica la URL donde se envían los datos.
- El atributo “method” indica el modo de envío:
  - GET: es la opción por defecto, menos seguro y limitado en tamaño (8k aprox.).
  - POST: mas seguro, no limitado.
- También es posible enviar un formulario con el método “fo.submit()”. Ej:

```
let form = document.createElement('form');
form.action = 'https://google.com/search';
form.method = 'GET';
form.innerHTML = '<input name="q" value="jcyl">';
document.body.append(form);
form.submit();
```

Envía a Google una URL como:  
[www.google.com/search?q=jcyl](https://www.google.com/search?q=jcyl)  
El buscador visualiza los resultados de buscar “jcyl”.

# Obtener valores de los campos. Input y textarea

- En cuadros de texto y textarea se utiliza la propiedad **value**. Ejemplo:

```
<form id="cliente" action="#" method="GET">
...<label>Nombre:<input name="nombre" value="Pedro" type="text"></label>
...<label>Edad:<input name="edad" value="68" type="text"></label>
...<label>Texto:<textarea name="txt">Escribe aquí</textarea></label>
...<input type="submit" name="enviar" value="Enviar">
</form>

<script>
...const fo = document.querySelector('#cliente');
...fo.addEventListener('submit', function() {
...  let res = `Nombre: ${fo['nombre'].value} \nEdad: ${fo['edad'].value} \nTexto: ${fo['txt'].value}`;
...  alert(res);
...});
</script>
```

- En los controles 'input type="text"' la propiedad "value" contiene el texto inicial o el texto que escribe el usuario.
- En los controles 'textarea' la propiedad "value" contiene el texto contenido en la etiqueta o el texto que escribe el usuario.



# Obtener valores de campos. RadioButtons.

```
<form id="cliente" action="#">
  ...<label><input type="radio" name="pregunta" value="si" checked >Afirmativo</label>
  ...<label><input type="radio" name="pregunta" value="no">Negativo</label>
  ...<input type="submit" name="enviar" value="Enviar">
</form>

<script>
  ...const fo = document.querySelector('#cliente');
  ...fo.addEventListener('submit', function() {
  ...  ...let rnl=fo['pregunta'];
  ...  ...console.log(rnl.value); // si
  ...  ...console.log(`${rnl[0].value} --> ${rnl[0].checked}`); // si --> true
  ...  ...console.log(`${rnl[1].value} --> ${rnl[1].checked}`); // no --> false
  ...  ...});
</script>
```

- Todos los radiobuttons con el mismo nombre forman un mismo grupo.
- La propiedad “value” del grupo contiene el “value” del radio seleccionado.
- También se puede acceder a cada radiobutton para leer su valor y comprobar si está seleccionado (propiedad “checked”).



# Obtener valores de campos. Checkbox.

```
<form id="formulario" action="#">
  ....<label><input type="checkbox" name="color" value="a" checked>Azul</label>
  ....<label><input type="checkbox" name="color" value="v">Verde</label>
  ....<label><input type="checkbox" name="color" value="r">Rojo</label>
  ....<input type="submit" name="enviar" value="Enviar">
</form>

<script>
  ....const fo = document.querySelector('#formulario');
  ....fo.addEventListener('submit', function() {
  ....  ....let cb=fo['color'];
  ....  ....console.log(`${cb[0].value}--> ${cb[0].checked}`); // a--> true
  ....  ....console.log(`${cb[1].value}--> ${cb[1].checked}`); // v--> false
  ....  ....console.log(`${cb[2].value}--> ${cb[2].checked}`); // r--> false
  ....  });
</script>
```

- Todos los checkbox con el mismo nombre forman un grupo.
- Obteniendo la referencia al grupo (ej: fo['color']) se puede acceder a cada elemento como un array de nodos y obtener las propiedades 'value' y 'checked'.

# Obtener valores de campos. Listas desplegables.

```
<form id="formulario" action="#">
  <select name="colores">
    <option value="" selected>-Seleccionar un color-</option>
    <option value="rojo">Rojo</option>
    <option value="verde">Verde</option>
  </select>
  <input type="submit" name="enviar" value="Enviar">
</form>
<script>
  const fo = document.querySelector('#formulario');
  fo.addEventListener('submit', function() {
    let lista=fo['colores'];
    console.log(lista.selectedIndex); // 0
    console.log(`${lista[0].value} --> ${lista[0].selected}`); // rojo --> true
    console.log(`${lista[1].value} --> ${lista[1].selected}`); // verde --> false
    console.log(`${lista[2].value} --> ${lista[2].selected}`); // verde --> false
  });
</script>
```

- Se construyen con las etiquetas “select” y “option”.
- El control “select” admite la propiedad “selectedIndex” y contiene un array de nodos con cada una de las opciones (con las propiedades “value” y “selected”.

# Mas controles.

- Existen muchos más controles de interés:
  - “<input>” permite otros atributos “type” muy utilizados como: “password”, “image”, “hidden” o “file”.
  - Otros atributos “type” de “<input>” que se incluyeron en el HTML5: “color”, “date”, “time”, “tel”, “email”, “range”, etc.

Más información con ejemplos de diferentes <input>:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>




# Validación de formularios. Atributos.

- La validación de los datos que provienen de formularios se puede hacer en el cliente, en el servidor y en ambos sitios. Esta última opción es la recomendada.
- Tradicionalmente la validación en el cliente se realizaba en javascript pero desde la aparición de HTML5 existen atributos que permiten validaciones básicas.

Atributo	Descripción
minlength	Para campos de tipo texto indica la longitud mínima requerida.
maxlength	Para campos de tipo texto indica la longitud máxima permitida.
min	Para campos de tipo numérico, fecha u hora indica el valor mínimo permitido
max	Para campos de tipo numérico, fecha u hora indica el valor máximo permitido
step	Para campos de tipo numérico, fecha u hora indica el salto permitido (por defecto es 1)
required	Indica un campo que se debe completar obligatoriamente
disable	Campo desactivado. No se puede modificar. No se envía.
readonly	No se puede modificar. Se envía.
pattern	Expresión regular que determina los valores válidos en un campo.

# Validación de formularios. Pseudoselectores CSS.

- Además existen pseudoselectores CSS de validación. Por ejemplo:
- “:valid” se activan cuando el campo es válido
- “:invalid” se activa cuando el campo no es válido.
- “:required” se activa cuando el campo es requerido.

```
input:required {  
    background:  palegreen;  
}  
input:valid, textarea:valid {  
    background:  lightskyblue;  
}  
input:invalid, textarea:invalid {  
    background:  orange;  
}
```

Usuario:

Usuario:

Usuario:

Realmente en este caso no se podría activar “:required” y “:valid” (o “:invalid”) al mismo tiempo.



# Validación de formularios. Atributos. Ejemplo:

```
<form name="formulario" method="post" action="#">
  <!-- Nombre de usuario. Obligatorio, entre 5-20 caracteres -->
  Usuario:
  <input type="text" name="nombre" placeholder="Nombre" minlength="5" maxlength="20" required />
  <!-- Contraseña. Obligatorio, mínimo 10 caracteres -->
  Password:
  <input type="password" name="pass" placeholder="Contraseña" minlength="10" required />
  <!-- N.I.F. Obligatorio con expresión regular -->
  N.I.F:
  <input type="text" name="nif" placeholder="N.I.F." required pattern="^\d{7,8}[a-zA-Z]$" />

  <input type="submit" Value="Enviar"></input>
</form>
```

Usuario:  Password:  N.I.F:

Si la validación no es correcta se visualiza un mensaje. Ejemplo:

Usuario:

! Completa este campo

N.I.F:

! Utiliza un formato que coincida con el solicitado

# Validación de formularios. Atributos.

- Si un campo es **required** no se puede enviar el formulario sin introducir el dato y que sea correcto.
- Si el campo no es **required** se puede enviar el formulario con el campo vacío aunque no sea correcto.
- Aunque menos utilizado, se puede:
  - Comprobar en cualquier momento si un campo (elemento HTML) es válido con **checkValidity()**.
  - Establecer un mensaje personalizado de error con **setCustomValidity()**.



# Validación de formularios. Javascript. Ejemplo:

```
<form name="formulario" method="post" action="#">
  Usuario:
  <input type="text" name="nombre" placeholder="Nombre" minlength="5" maxlength="20" required />
  Password:
  <input type="password" name="pass" placeholder="Contraseña" minlength="10" required />
  N.I.F:
  <input type="text" name="nif" placeholder="N.I.F." required pattern="\d{7,8}[a-zA-Z]$" />

  <input type="submit" Value="Enviar"></input>
</form>
```

```
function validaNif(nif) { ...
const btn = document.querySelector("form");
btn.addEventListener("submit", (e) => {
  e.preventDefault();
  if (!validaNif(e.target["nif"])) {
    alert("El NIF no es correcto");
    return;
  }
  e.target.submit();
});
```

Una vez que los campos cumplen con los atributos de validación podemos hacer la última validación en javascript antes de mandar el formulario al servidor.

Comprobamos si el NIF es correcto con una función de validación y decidimos si enviar o no el formulario al servidor.

**e.preventDefault()** evita el comportamiento por defecto.  
**e.target** contiene el formulario y **e.target["nif"]** el nif  
**e.target.submit()** envía el formulario.

# Ejercicio 6 (ej6.html)

- Codificar una página web con un formulario que permita introducir los siguientes campos (todos requeridos) de un trabajador de una empresa:
  - Nombre: con una longitud mínima de 20 caracteres (solo caracteres).
  - NIF: 7 u 8 dígitos y un carácter mayúscula.
  - Edad: valor numérico entre 18 y 65.
  - Fecha de alta: en la empresa: con formato válido.
  - Email: que contenga al menos 3 caracteres seguido de “@” seguido de un mínimo de tres caracteres por cada dominio y terminando con uno de los dominios “.es”, “.pt” o “.fr”.
- Cuando se cumplan todos los criterios de validación de los atributos se deberá comprobar mediante javascript:
  - Si el NIF es correcto o no (en función de su parte numérica y la letra final).
  - Si la fecha de alta es correcta: solo se permiten fechas de alta en días laborables (de lunes a viernes).
- En todo momento se visualizarán los mensajes apropiados.