HTML/CSS S2: Formularios y rutas

Después de esta lección podrás:

1. Comprender mejor los formularios.
2. Entender las entradas de datos más usadas en los forms.
3. Realizar validaciones de las entradas de datos.

Formularios

Los formularios son las piezas HTML que nos permiten recoger la información que introduce el usuario. Están compuestos por las siguientes etiquetas:

- **Etiqueta** `<form>`**: e**tiqueta principal que envuelve todo el contenido del formulario. En esta etiqueta se pueden definir dos atributos. “action” para definir la localización donde se enviará la información que se recoja en el formulario y “method” para definir el método HTTP en el que se enviará esta información (GET, POST).

```
<form action="/next-page" method="post">
</form>
```

- Etiqueta `<fieldset>`: etiqueta que define una agrupación de elementos comunes dentro del formulario. Siempre deberá ir acompañada de una etiqueta “**legend**” a la que se le asignará un título descriptivo. Este título siempre debe existir por motivos de accesibilidad, pero puede ser ocultado mediante css.

```
<form action="/next-page" method="post">
  <fieldset>
    <legend>Datos Básicos</legend>
  </fieldset>
</form>
```

- Etiqueta `<input>`: es la etiqueta más versátil, define un campo dentro de un formulario y tiene tres principales atributos:
  - El atributo **type** define el comportamiento que tendrá el input por defecto. Entre los tipos más importantes tenemos: text, checkbox, date, number, password, reset, radio, reset, submit, file...

```
<input type="text" name="name" id="name">
<input type="date" name="date" id="date">
<input type="number" name="price" id="price">
```

- **Name**: especifica un nombre a cada uno de los campos del formulario. Esto nos ayuda a la hora de enviar y recibir el contenido de los mismos cuando hagamos submit en el formulario.
- **For**: enlaza el input con la etiqueta **<label>** que tenga el mismo "id".

- Etiqueta `<label>`: etiqueta que acompaña a un campo del formulario con un título descriptivo. Siempre debemos de especificar el campo al que acompañará con el atributo "**for**".

```
<label for="name">Nombre</label>
```

- Etiqueta `<textarea>`: Esta etiqueta es similar a un input de tipo "**text**" con la diferencia de que un **textarea** no se auto cierra y se suele utilizar para recoger textos de mayor cantidad como comentarios, descripciones, observaciones... etc. Esta etiqueta también puede utilizar el atributo "id" para enlazarse con un `<label>`.

```
<label for="comments">Comentarios:</label>
<textarea id="comments"></textarea>
```

- Etiqueta `<button>`: Etiqueta que permitirá que nuestro usuario pueda enviarnos la información que ha introducido en los diferentes campos, para ello solo deberemos agregarle el `type="submit"`

```
<button type="submit">Enviar</button>
```

Ejemplo completo de formulario

Para finalizar, os compartimos un ejemplo de un formulario bien formado compuesto de los anteriores elementos:

```
<form action="/next-page" method="post">

    <fieldset>
        <legend>Datos Básicos</legend>
        <label for="name">Nombre</label>
        <input type="text" name="name" id="name">
        <label for="date">Nombre</label>
        <input type="date" name="date" id="date">
    </fieldset>

    <fieldset>
        <legend>Datos productos</legend>
        <label for="price">Precio</label>
        <input type="number" name="price" id="price">
        <label for="comments">Comments</label>
        <textarea id="comments"></textarea>
    </fieldset>

    <button type="submit">Enviar</button>
</form>
```

Validación de Formularios con HTML5

La **validación** de formularios **HTML5** permite comprobar cada entrada de datos, tiene un gran rendimiento ya que es nativa. No es personalizable y nos tenemos que ajustar a usar los atributos y opciones por defecto: ***minlength, maxlength, min, max, step, required...***

Un ejemplo sería:

```
<form method="post" action="/register">
  <h3>Formulario</h3>
  <div>
    <!-- Nombre de usuario. Campo obligatorio, entre 5-40 carácteres -->
    <label for="nombre">Usuario:</label>
    <input type="text" name="nombre" id="nombre" placeholder="Nombre de usuario" minlength="5" maxlength="40" required>
  </div>
  <div>
    <!-- Contraseña. Campo obligatorio, mínimo 11 carácteres -->
    <label for="pass">Password:</label>
    <input type="password" name="pass" id="pass" placeholder="Contraseña" minlength="11" required>
  </div>
  <p>Complete la información para su registro</p>
  <button type="submit">Registrarse</button>
</form>
```

En caso de que queramos customizar las validaciones nativas de HTML5, podemos añadir CSS utilizando las pseudoclases de validación ***:valid*** e ***:invalid***.

Aplicaremos estilos a los campos **<input>** teniendo en cuenta su validación.

```
input:valid {
  background: lightgreen;
}

input:invalid {
  background: red;
}
```

También se pueden dar estilos haciendo uso de los atributos de las etiquetas HTML, es decir:

```
input[type="text"] {
  margin-bottom: 20px;
}
```

Dicho esto, ya sabéis que NO es recomendable generalizar los estilos sobre los elementos. Debemos siempre usar clases para ello:

```
.form-input:valid {
  background: lightgreen;
}

.form-input:invalid {
  background: red;
}

.form-input.form-input-text {
```

```
  margin-bottom: 20px;
}
```

Tipos de Validación básicos

required

La validación requerida indica que el campo es obligatorio en el formulario, y no podremos enviar los datos hasta rellenar el campo.

```
<form method="post" action="/nombre">
  <h3>Formulario</h3>
  <div>
    <label for="name">Nombre:</label>
    <input class="form-input form-input-text" type="text" name="name" placeholder="Su nombre" required>
  </div>
  <button type="submit">Enviar</button>
</form>
```

minlength/maxlength (text)

Esta validación es usada para indicar un tamaño mínimo o máximo de caracteres en una entrada de datos. Así podremos restringir un input para que al menos tenga N caracteres, o que no se exceda de N caracteres.

```
<form method="post" action="/user">
  <h3>Formulario</h3>
  <div>
    <label for="name">Nombre:</label>
    <input class="form-input form-input-text" type="text" name="name" placeholder="Su nombre" required>
  </div>
  <div>
    <label for="nick">Nick:</label>
    <input class="form-input form-input-text" type="text" name="nick" placeholder="Su nick" minlength="2" maxlength="8">
  </div>
  <button type="submit">Enviar</button>
</form>
```

min/max (number)

De igual manera que con los campos de texto, podemos restringir los campos números mediante un valor máximo y un valor mínimo.

```
<form method="post" action="/user">
  <h3>Formulario</h3>
  <div>
    <label for="name">Nombre:</label>
    <input class="form-input form-input-text" type="text" name="name" placeholder="Su nombre" required>
  </div>
  <div>
    <label for="nick">Nick:</label>
    <input class="form-input form-input-text" type="text" name="nick" placeholder="Su nick" minlength="2" maxlength="8">
  </div>
  <div>
    <label for="age">Edad:</label>
```

```
    <input class="form-input form-input-number" type="number" name="age" placeholder="Su edad" min="14" max="99">
  </div>
  <button type="submit">Enviar</button>
</form>
```

type (email, number, date, etc)

Un paso más avanzado sería realizar validación por tipo de input. En HTML5 existen de manera nativa validaciones sobre algunas entradas de dato, por defecto. Es el caso de la validación de emails o fechas, por ejemplo:

```
<form method="post" action="/nombre">
  <h3>Formulario</h3>
  <div>
    <label for="name">Nombre:</label>
    <input class="form-input form-input-text" type="text" name="name" placeholder="Su nombre" required>
  </div>
  <div>
    <label for="nick">Nick:</label>
    <input class="form-input form-input-text" type="text" name="nick" placeholder="Su nick" minlength="2" maxlength="8">
  </div>
  <div>
    <label for="age">Edad:</label>
    <input class="form-input form-input-number" type="number" name="age" placeholder="Su edad" min="14" max="99">
  </div>
  <div>
    <label for="email">Correo:</label>
    <input class="form-input form-input-number" type="email" name="email" placeholder="Su correo" required>
  </div>
  <button type="submit">Enviar</button>
</form>
```

Validación por Patrón

HTML5 nos **permite** usar **patterns**, vamos a ver algunos aspectos básicos de estos por si en algún momento os animáis a usarlos:

Se rigen por conjuntos de caracteres que cumplen con un patrón regex.

Validación por Patrón → Expresiones regulares

Aa Expresión regular	Carácter especial	Denominación	Descripción
.	Punto	Comodín	Cualquier carácter
A|B	Pipe	Opciones lógicas	Alternativa (o A o B)
C(A|B)	Paréntesis	Agrupaciones	Agrupación (o CA o CB)
[0-9]	Corchetes	Rango carácteres	Dígito (del 0 al 9)
[A-Z]	Corchetes	Rango carácteres	Letra mayúscula de la A a Z
[^A-Z]	^ en corchetes	Rango exclusión	No letra A a Z mayúscula
[0-9]*	Asterisco	Cierre	Un dígito repetido 0 ó más veces (vacío incluido)
[0-9]+	Signo más	Cierre positivo	Un dígito repetido 1 ó más veces
[0-9]{3}	Llaves	Coincidencia exacta	Cifra de 3 dígitos (dígito repetido 3 veces)
[0-9]{2,4}		Coincidencia (rango)	Cifra de 2 a 4 dígitos (rep. de 2 a 4 veces)
b?	Interrogación	Carácter opcional	El carácter b puede aparecer o puede que no

Expresión regular	Carácter especial	Denominación	Descripción
\.	Barra invertida	Escape	El carácter . literalmente (no como comodín)

Ahora veremos en HTML un ejemplo de uso de estas expresiones regulares o patterns, imaginemos un caso donde los requisitos serían:

- Tipo de campo: **Nombre de usuario**
- Campo obligatorio: **required**.
- Entre 5-40 carácteres: **minlength="5" maxlength="40"**
- Sólo se permiten letras (mayúsculas y minúsculas) o números: **pattern="[A-Za-z0-9]+"**

```
<form method="post" action="/register">
  <h3>Formulario</h3>
  <div>
    <label for="nombre">Usuario:</label>
    <input class="form-input form-input-text" type="text" name="nombre" placeholder="Su nombre de usuario" minlength="5" maxlength="40" req
      pattern="[A-Za-z0-9]+">
  </div>
  <button type="submit">Registrarse</button>
</form>
```

En caso de **no** querer utilizar el ***minlength*** y el ***maxlength*** para hacerlo todo con un **pattern**:

```
<form method="post" action="/register">
  <h3>Formulario</h3>
  <div>
    <label for="nombre">Usuario:</label>
    <input class="form-input form-input-text" type="text" name="nombre" placeholder="Su nombre de usuario" required
      pattern="[A-Za-z0-9]{5,40}" title="Letras y números. Tamaño mínimo: 5. Tamaño máximo: 40">
  </div>
  <button type="submit">Registrarse</button>
</form>
```

Veamos un último ejemplo, vamos a pedir al usuario que nos diga el modelo de Audi que tiene, esto sería un ejemplo claro de un formulario para un taller de Audi.

Los modelos posibles son A1, A3, A4 y A6. No queremos mostrar el típico select que se despliega y seleccionas una opción, podemos mostrar un campo de texto y colocar una validación como la siguiente:

- Tipo de campo: **Modelo de coche**
- Campo obligatorio: **required**.
- Sólo se permiten las opciones: **A1, A3, A4 y A6**

```
<form method="post" action="/coche">
  <h3>Formulario</h3>
  <div>
    <label for="coche">Coche:</label>
    <input class="form-input form-input-text" type="text" name="coche" placeholder="Su modelo de coche" required
      pattern="A|a(1|3|4|6)" title="Modelos posibles: A1, A3, A4 y A6">
```

```
  </div>
  <button type="submit">Enviar</button>
</form>
```

En este punto conocemos todo lo necesario para construir un formulario robusto con su validación correcta, para que el usuario introduzca siempre los datos.

Ejemplo GET/POST contra API

Por último, vamos a ver dos ejemplos prácticos, sobre el uso del GET para recuperar información:

```
<form method="get" action="https://reqres.in/api/users?page=2">
  <h3>Formulario</h3>
  <div>
    <label for="page">Pagina:</label>
    <input class="form-input form-input-text" type="text" name="page" placeholder="Pagina" required>
  </div>
  <button type="submit">Enviar</button>
</form>
```

Y sobre el uso del POST para persistir información en un servidor:

```
<form method="post" action="https://reqres.in/api/users">
  <h3>Formulario</h3>
  <div>
    <label for="name">Nombre:</label>
    <input class="form-input form-input-text" type="text" name="name" placeholder="Nombre" required>
  </div>
  <div>
    <label for="job">Trabajo:</label>
    <input class="form-input form-input-text" type="text" name="job" placeholder="Trabajo" required>
  </div>
  <button type="submit">Enviar</button>
</form>
```

Rutas en HTML

Antes de continuar avanzando, es muy importante que entendamos el concepto de rutas y cómo funcionan. Pero... ¿Dónde se usan las rutas?

En general en el mundo web se incluyen muchos ficheros externos como por ejemplos haremos con el CSS y JavaScript, en estos casos hará falta que indiquemos la ruta en la que se encuentran estos ficheros.

También en ocasiones haremos uso de enlaces, de imágenes, vídeos... etc, en estos casos también debemos indicar las rutas en las que se encuentran estos ficheros.

Hay distintos tipos de rutas, vamos a verlas:

Rutas absolutas

Se trata de rutas que no dependen de la ruta sobre en la que estemos navegando actualmente.

Generalmente incluyen toda la información de la ruta: protocolo, puerto, dominio y ubicación.

Estas rutas pueden incluirse en cualquier página sin riesgo de error al copiar/pegar, ya que como hemos dicho estas rutas no dependen del documento donde se encuentren. Siempre apuntan al mismo sitio. Ejemplo de rutas absolutas:

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Es es mi blog</title>
    <script type="text/javascript" src="https://maps.googleapis.com/maps/api/js"></script>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap.css">
</head>

<body>
    <a href="http://getbootstrap.com/">Página web de Bootstrap</a>
    <img src="http://placekitten.com/400/400" alt="Imágenes de gatos">
</body>

</html>
```

En el ejemplo anterior hemos incluido dos librerías externas mediante sus URL's absolutas.

También hemos incluido un enlace y una imagen de forma absoluta.

Este tipo de rutas debe usarse para incluir o enlazar con elementos externos o que se encuentren en otros dominios, pero no se recomiendan para enlazar con elementos que se encuentren bajo el mismo dominio, ya que es menos sostenible.

Rutas relativas

Las rutas relativas se usan para enlazar referenciar elementos que se encuentran bajo el mismo dominio que la página HTML en la que nos encontramos. Esto tiene una ventaja muy grande: no importa dónde se desplieguen nuestras páginas (dominio o carpeta).

En las rutas relativas no se incluye el dominio y podemos encontrar de varios tipos:

Rutas relativas que empiezan por /

La ruta se generará a partir del dominio en el que nos encontramos. Por ejemplo, si nos encontramos en la página web: "http://midominio.com/revista/deporte" y encontramos el siguiente enlace en nuestro documento HTML: "/videos/tormentas", nuestro navegador interpretará este enlace como "http://midominio.com/videos/tormentas"

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8">
    <title>Es es mi blog</title>
    <script type="text/javascript" src="/scripts/maps/api/js"></script>
    <link rel="stylesheet" href="/styles/bootstrap.css">
</head>

<body>
    <a href="/">Ir a la home</a>
    <a href="/videos/tormentas">Ir a la sección de videos de tormentas</a>
    <img src="/images/400/400" alt="Imágenes de gatos">
</body>

</html>
```

Rutas relativas que empiezan por ./

El navegador considera que la ruta parte desde el mismo punto en el que nos encontramos, en el ejemplo anterior el documento “deporte” se encontraba bajo la ruta “http://midominio.com/revista/” y por tato construye la ruta a partir de ese punto.

La ruta se generará a partir del dominio y de la carpeta en la que nos encontramos actualmente. Siguiendo con el ejemplo anterior, si nos encontramos en la ruta “http://midominio.com/revista/deporte” y encontramos el siguiente enlace en nuestro documento HTML: “./videos/tormetas”, nuestro navegador en este caso interpretará el enlace como “http://midominio.com/revista/videos/tormentas”

Rutas que empiezan por ../

Este caso es muy parecido al anterior, salvo que los dos puntos en vez de indicar que se trata del mismo nivel, nos indica que se encuentra en un nivel superior.

Si deseamos subir más niveles se puede realizar concatenando más “../”

Siguiendo con ejemplo anterior, partimos de la ruta "http://midominio.com/revista/deportes" y encontramos un enlace en el HTML con la siguiente ruta "../economía" Nuestro navegador interpretará este enlace como "http://midominio.com/revista/economia".

Rutas relativas que son solo el nombre del fichero

Es el caso que hemos usado durante los ejemplos. Una ruta que no especifica ningún comienzo, está indicando que debe construirse a partir del punto en el que nos encontramos.

Por tanto funciona por tanto del mismo modo que “./”, salvo que usar el punto y la barra es más correcto que no usar nada, ya que de esta manera estamos indicando una ruta de forma explícita.