4.

ANEXO: Formularios web

Contenidos

1. Formularios web

1.1. Formularios

Los formularios web son la interface con el usuario más importante de todas, pues permiten a los usuarios insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación.

Durante los últimos años, se crearon códigos personalizados y librerías para procesar formularios en el ordenador del usuario. HTML5 vuelve a estas funciones estándar agregando nuevos atributos, elementos y una API completa, la API Forms, que no trataremos en este anexo. Ahora la capacidad de procesamiento de información insertada en formularios en tiempo real ha sido incorporada en los navegadores y completamente estandarizada.

1.1.1. El elemento <form>

Los formularios en HTML no han cambiado mucho. La estructura sigue siendo la misma, pero HTML5 ha agregado nuevos elementos, tipos de campo y atributos para expandirlos tanto como sea necesario y proveer así las funciones actualmente implementadas en aplicaciones web.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
  <section>
    <form name="miformulario" id="miformulario" method="get">
      <input type="text" name="nombre" id="nombre">
      <input type="submit" value="Enviar">
    </form>
  </section>
</body>
</html>
```

Listado. Estructura normal de un formulario.

En el Listado creamos una plantilla básica para formularios. Como se puede ver, la estructura del formulario y sus atributos siguen siendo igual que en especificaciones previas. Sin embargo, existen nuevos atributos para el elemento <**form**>:

- **autocomplete** Este es un viejo atributo que se ha vuelto estándar en esta especificación. Puede tomar dos valores: **on** y **off**. El valor por defecto es **on**. Cuando es configurado como **off** los elementos **<input>** pertenecientes a ese formulario tendrán la función de autocompletar desactivada, sin mostrar entradas previas como posibles valores.

  Puede ser implementado en el elemento **<form>** o en cualquier elemento **<input>** independientemente.

- **novalidate** Una de las características de formularios en HTML5 es la capacidad propia de validación. Los formularios son validados automáticamente. Para evitar este comportamiento, podemos usar el atributo **novalidate**. Para lograr lo mismo para elementos **<input>** específicos, existe otro atributo llamado **formnovalidate**. Ambos atributos son booleanos, y no tiene que ser especificado ningún valor (su presencia es suficiente para activar su función).

1.1.2. El elemento <input>

El elemento más importante en un formulario es **<input>**. Este elemento puede cambiar sus características gracias al atributo **type** (tipo). Este atributo determina qué clase de entrada se espera desde el usuario. Los tipos disponibles hasta el momento eran el multipropósitos **text** (para textos en general) y sólo unos pocos más específicos, como **password** o **submit**. HTML5 ha expandido las opciones incrementando las posibilidades para este elemento.

En HTML5 estos nuevos tipos no solo están especificando qué clase de entrada es esperada sino también diciéndole al navegador qué debe hacer con la información recibida. El navegador procesará los datos ingresados de acuerdo al valor del atributo **type** y validará la entrada o no.

El atributo **type** trabaja junto con otros atributos adicionales para ayudar al navegador a limitar y controlar en tiempo real lo introducido por el usuario.

Para comprobar cómo funciona cada tipo de campo estudiado de aquí en adelante, puede crearse un nuevo archivo HTML como la plantilla del Listado anterior e ir reemplazando los elementos **<input>** en la plantilla por aquellos que se quiere probar y abrir nuevamente el archivo en el navegador. En este momento la forma en la que son tratados los tipos de campo puede variar entre distintos navegadores, por lo que se recomienda probar el código en cada navegador disponible.

1.1.3. Tipo email

Casi todo formulario en la web ofrece un campo para ingresar una dirección de e-mail, pero hasta ahora el único tipo de campo disponible para esta clase de datos era **text**. El tipo **text** representa un texto general, no un dato específico, por lo que teníamos que controlar la entrada con código JavaScript para estar seguros de que el texto ingresado correspondía a un e-mail válido. Ahora el navegador se hace cargo de esto con el nuevo tipo **email**:

```
<input type="email" name="miemail" id="miemail">
```

El texto insertado en el campo de tipo **email** será controlado por el navegador y validado como un e-mail. Si la validación falla, el formulario no será enviado.

Cómo cada navegador responderá a una entrada inválida no está determinado en la especificación de HTML5. Por ejemplo, algunos navegadores mostrarán un borde rojo alrededor del elemento **<input>** que produjo el error y otros lo mostrarán en azul.

Existen formas de personalizar esta respuesta, pero no las veremos en este texto.

1.1.4. Tipo search

El tipo **search** (búsqueda) no controla la entrada, es sólo una indicación para los navegadores. Al detectar este tipo de campo algunos navegadores cambiarán el diseño del elemento para ofrecer al usuario un indicio de su propósito.

```
<input type="search" name="busqueda" id="busqueda">
```

1.1.5. Tipo url

Este tipo de campo trabaja exactamente igual que el tipo **email**, pero es específico para direcciones web. Está destinado a recibir sólo URLs absolutos, y retornará un error si el valor es inválido.

```
<input type="url" name="miurl" id="miurl">
```

1.1.6. Tipo tel

Este tipo de campo es para números telefónicos. A diferencia de los tipos **email** y **url**, el tipo **tel** no requiere ninguna sintaxis en particular. Es sólo una indicación para el navegador en caso de que necesite hacer ajustes de acuerdo al dispositivo en el que la aplicación se ejecuta.

```
<input type="tel" name="telefono" id="telefono">
```

1.1.7. Tipo number

Como su nombre indica, el tipo **number** sólo es válido cuando recibe una entrada numérica. Existen algunos atributos nuevos que pueden ser útiles para este campo:

- **min**. El valor de este atributo determina el mínimo valor aceptado para el campo.
- **max**. El valor de este atributo determina el máximo valor aceptado para el campo.

- **step**. El valor de este atributo determina el tamaño en el que el valor será incrementado o disminuido en cada paso. Por ejemplo, si declara un valor de 5 para **step** en un campo que tiene un valor mínimo de 0 y máximo de 10, el navegador no permitirá especificar valores entre 0 y 5 o entre 5 y 10.

No es necesario especificar ambos atributos (**min** y **max**), y el valor por defecto para **step** es 1.

```
<input type="number" name="numero" id="numero" min="0" max="10" step="5">
```

1.1.8. Tipo range

Este tipo de campo hace que el navegador construya una nueva clase de control que no existía previamente. Este nuevo control le permite al usuario seleccionar un valor a partir de una serie de valores o rango. Normalmente se muestra en pantalla como una puntero deslizable o un campo con flechas para seleccionar un valor de entre los predeterminados, pero no existe un diseño estándar hasta el momento.

El tipo **range** usa los atributos **min** y **max** vistos previamente para configurar los límites del rango. También puede utilizar el atributo **step** para establecer el tamaño en el cual el valor del campo será incrementado o disminuido en cada paso.

```
<input type="range" name="numero" id="numero" min="0" max="10" step="5">
```

Podemos declarar el valor inicial utilizando el viejo atributo **value** y usar JavaScript para mostrar el número seleccionado en pantalla como referencia.

1.1.9. Tipo date

Este es otro tipo de campo que genera una nueva clase de control. En este caso fue incluido para ofrecer una forma mejor de introducir una fecha. Algunos navegadores muestran en pantalla un calendario que aparece cada vez que el usuario hace clic sobre el campo. El calendario le permite al usuario seleccionar un día que será introducido en el campo junto con el resto de la fecha. Un ejemplo de uso es cuando necesitamos proporcionar un método para seleccionar una fecha para un vuelo o la entrada a un espectáculo.

Gracias al tipo **date** ahora es el navegador el que se encarga de construir un almanaque o las herramientas necesarias para facilitar la introducción de este tipo de datos.

```
<input type="date" name="fecha" id="fecha">
```

La interface no fue declarada en la especificación. Cada navegador provee su propia interface y a veces adaptan el diseño al dispositivo en el cual la aplicación está siendo ejecutada. Normalmente el valor generado y esperado tiene la sintaxis **año-mes-día**.

1.1.10. Tipo week

Este tipo de campo ofrece una interface similar a **date**, pero sólo para seleccionar una semana completa. Normalmente el valor esperado tiene la sintaxis **2019-W50** donde **2019** es al año y **50** es el número de la semana.

```
<input type="week" name="semana" id="semana">
```

1.1.11. Tipo month

Similar al tipo de campo previo, éste es específico para seleccionar meses. Normalmente el valor esperado tiene la sintaxis **año-mes**.

```
<input type="month" name="mes" id="mes">
```

1.1.12. Tipo time

El tipo de campo time es similar a **date**, pero sólo para la hora. Toma el formato de horas y minutos, pero en este momento su comportamiento depende de cada navegador. Normalmente el valor esperado tiene la sintaxis **hora:minutos:segundo**s, pero también puede ser solo **hora:minutos**.

```
<input type="time" name="hora" id="hora">
```

1.1.13. Tipo datetime

El tipo de campo **datetime** es para introducir fecha y hora completa, incluyendo la zona horaria.

```
<input type="datetime" name="fechahora" id="fechahora">
```

1.1.14. Tipo datetime-local

El tipo de campo **datetime-local** es como el tipo **datetime** sin la zona horaria.

```
<input type="datetime-local" name="tiempolocal" id="tiempolocal">
```

1.1.15. Tipo color

Además de los tipos de campo para fecha y hora existe otro tipo que provee una interface predefinida similar para seleccionar colores. Normalmente el valor esperado para este campo es un número hexadecimal, como #00FF00.

No se especificó ninguna interface como estándar en HTML5 para el tipo de campo **color**, pero es posible que los navegadores adopten o incorporen una rejilla con un conjunto básico de colores sea adoptada e incorporada en los navegadores.

```
<input type="color" name="micolor" id="micolor">
```

1.2. Nuevos atributos

Algunos tipos de campo requieren de la ayuda de atributos, como los atributos **min**, **max** y **step**. Otros tipos de campo requieren la asistencia de atributos para mejorar su rendimiento o determinar su importancia en el proceso de validación. Ya vimos algunos de ellos, como **novalidate** para evitar que el formulario completo sea validado o **formnovalidate** para hacer lo mismo con elementos individuales. El atributo **autocomplete**, también visto, provee medidas de seguridad adicionales para el formulario completo o elementos individuales. Aunque útiles, estos atributos no son los únicos incorporados por HTML5.

1.2.1. Atributo placeholder

Especialmente en tipos de campo **search**, pero también en entradas de texto normales, el atributo **placeholder** representa una sugerencia corta, una palabra o frase provista para ayudar al usuario a introducir la información correcta. El valor de este atributo es mostrado en pantalla por los navegadores dentro del campo, como una marca de agua que desaparece cuando el elemento adquiere el foco.

```
<input type="search" name="busqueda" id="busqueda" placeholder="Escriba su
búsqueda">
```

1.2.2. Atributo required

Este atributo booleano no dejará que el formulario sea enviado si el campo se encuentra vacío. Por ejemplo, cuando usamos el tipo **email** para recibir una dirección de e-mail, el navegador comprueba si la entrada es un e-mail válido o no, pero validará la entrada si el campo está vacío. Cuando se incluye el atributo **required**, la entrada será válida sólo si se cumplen las dos condiciones: que el campo no esté vacío y que el valor ingresado esté de acuerdo con los requisitos del tipo de campo.

```
<input type="email" name="miemail" id="miemail" required>
```

1.2.3. Atributo multiple

El atributo **multiple** es otro atributo booleano que puede ser usado en algunos tipos de campo (por ejemplo, **email** o **file**) para permitir introducir múltiples entradas en el mismo campo. Los valores insertados deben estar separados por coma para ser válidos.

```
<input type="email" name="miemail" id="miemail" multiple>
```

El código anterior permite la inserción de múltiples valores separados por coma, y cada uno de ellos será validado por el navegador como una dirección de **e-mail**.

1.2.4. Atributo autofocus

El atributo **autofocus** enfocará la página web sobre el elemento seleccionado pero considerando la situación actual. No moverá el foco cuando ya haya sido establecido por el usuario sobre otro elemento.

```
<input type="search" name="busqueda" id="busqueda" autofocus>
```

1.2.5. Atributo pattern

El atributo **pattern** es para propósitos de validación. Usa expresiones regulares para personalizar reglas de validación. Algunos de los tipos de campo ya vistos validan cadenas de texto específicas, pero no permiten hacer validaciones personalizadas como, por ejemplo, un código postal, que consiste en 5 números. No existe ningún tipo de campo predeterminado para esta clase de entrada.

El atributo **pattern** nos permite crear nuestro propio tipo de campo para controlar esta clase de valores no ordinarios. Puede incluso incluir un atributo **title** para personalizar mensajes de error.

```
<input pattern="[0-9]{5}" name="codigopostal" id="codigopostal" title="Inserte
los 5 números de su código postal">
```

Las expresiones regulares son un tema no relacionado directamente con HTML5. Para obtener información al respecto, puede visitarse cualquier sitio web destinado a tal efecto.

1.2.6. Atributo form

El atributo **form** es una adición útil que nos permite declarar elementos para un formulario fuera del ámbito de las etiquetas **<form>**. Hasta ahora, para construir un formulario teníamos que escribir las etiquetas **<form>** de apertura y cierre y luego declarar cada elemento del formulario entre ellas. En HTML5 podemos insertar los elementos en cualquier parte del código y luego hacer referencia al formulario que pertenecen usando su atributo **id** y el atributo **form**:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Formularios</title>
</head>
<body>
  <nav>
    <input type="search" name="busqueda" id="busqueda" form="formulario">
  </nav>
  <section>
    <form id="formulario" name="formulario" method="get">
      <input type="text" name="nombre" id="nombre">
      <input type="submit" value="Enviar">
    </form>
  </section>
</body>
</html>
```

1.3. Nuevos elementos para formularios

Ya hemos visto los nuevos tipos de campos disponibles en HTML5. Veamos ahora los nuevos elementos HTML incorporados con la intención de mejorar o expandir las posibilidades de los formularios.

1.3.1. El elemento <datalist>

El elemento <**datalist**> es un elemento específico de formularios usado para construir una lista de ítems que luego, con la ayuda del atributo **list**, será usada como sugerencia en un campo del formulario.

```
<datalist id="informacion">
  <option value="123123123" label="Teléfono 1">
  <option value="456456456" label="Teléfono 2">
</datalist>
```

Listado. Construyendo la lista.

Este elemento utiliza el elemento <**option**> en su interior para crear la lista de datos a sugerir. Con la lista ya declarada, lo único que resta es referenciarla desde un elemento <**input**> usando el atributo **list**:

```
<input type="tel" name="telefono" id="telefono" list="informacion">
```

Listado. Ofreciendo una lista de sugerencias con el atributo **list**.

El elemento en el Listado anterior mostrará posibles valores para que el usuario elija.

NOTA: El elemento <**datalist**> puede no haber sido implementado por todos los navegadores en este momento.

1.3.2. El elemento <progress>

Este elemento no es específico de formularios, pero debido a que representa el progreso en la realización de una tarea, y usualmente estas tareas son comenzadas y procesadas a través de formularios, puede ser incluido dentro del grupo de elementos para formularios.

El elemento <**progress**> utiliza dos atributos para configurar su estado y límites. El atributo **value** indica qué parte de la tarea ya ha sido procesada, y **max** declara el valor a alcanzar para que la tarea se considere finalizada.

1.3.4. El elemento <meter>

Similar a <**progress**>, el elemento <**meter**> es usado para mostrar una escala, pero no de progreso. Este elemento tiene la intención de representar una medida, como el tráfico del sitio web, por ejemplo.

El elemento <**meter**> cuenta con varios atributos asociados: **min** y **max** configuran los límites de la escala, **value** determina el valor medido, y **low**, **high** y **optimum** son usados para segmentar la escala en secciones diferenciadas y marcar la posición que es óptima.

1.3.5. El elemento <output>

Este elemento representa el resultado de un cálculo. Normalmente ayudará a mostrar los resultados del procesamiento de valores provistos por un formulario. El atributo **for** asocia el elemento <**output**> con el elemento fuente que participa del cálculo, pero este elemento deberá ser referenciado y modificado desde código JavaScript. Su sintaxis es <**output**>**valor**</**output**>.

2. Resumen

Los formularios constituyen el principal medio de comunicación entre usuarios y aplicaciones web. HTML5 incorpora nuevos tipos para el elemento <**input**>, una API completa para validar y procesar formularios, y atributos para mejorar esta interface.

2.1. Tipos

Algunos de los nuevos tipos de campo introducidos por HTML5 tienen condiciones de validación implícitas. Otros sólo declaran un propósito para el campo que ayudará a los navegadores a presentar el formulario en pantalla.

- **email**. Este tipo de campo valida la entrada como una dirección de e-mail.
- **search**. Este tipo de campo da información al navegador sobre el propósito del campo (búsqueda) para ayudar a presentar el formulario en pantalla.
- **url**. Este tipo de campo valida la entrada como una dirección web.
- **tel**. Este tipo de campo da información al navegador sobre el propósito del campo (número telefónico) para ayudar a presentar el formulario en pantalla.
- **number**. Este tipo de campo valida la entrada como un número. Puede ser combinado con otros atributos (como **min**, **max** y **step**) para limitar los números permitidos.
- **range**. Este tipo de campo genera un nuevo control en pantalla para la selección de números. La entrada es limitada por los atributos **min**, **max** y **step**. El atributo **value** establece el valor inicial para el elemento.
- **date**. Este tipo de campo valida la entrada como una fecha en el formato **año-mes-día**.
- **month**. Este tipo de campo valida la entrada como una fecha en el formato **año-mes**.
- **week**. Este tipo de campo valida la entrada como una fecha en el formato **año-semana** donde el segundo valor es representado por una letra **W** y el número de la semana.
- **time**. Este tipo de campo valida la entrada como una hora en el formato **hora:minutos:segundos**. También puede tomar otras sintaxis como **hora:minutos**.
- **datetime**. Este tipo de campo valida la entrada como fecha y hora completa, incluyendo zona horaria.
- **datetime-local**. Este tipo de campo valida la entrada como una fecha y hora completa, sin zona horaria.
- **color**. Este tipo de campo valida la entrada como un valor de color.

2.2. Atributos

En HTML5 se agregaron también nuevos atributos para mejorar la capacidad de los formularios y ayudar al control de validación.

- **autocomplete**. Este atributo especifica si los valores insertados serán almacenados para futura referencia. Puede tomar dos valores: **on** y **off**.

- **autofocus**. Este es un atributo booleano que asigna el foco al elemento en el que se encuentra cuando la página es cargada.
- **novalidate**. Este atributo es exclusivo para elementos **<form>**. Es un atributo booleano que establece si el formulario será validado por el navegador o no.
- **formnovalidate**. Este atributo es exclusivo para elementos de formulario individuales. Es un atributo booleano que establece si el elemento será validado por el navegador o no.
- **placeholder**. Este atributo ofrece información que orientará al usuario sobre la entrada esperada. Su valor puede ser una palabra simple o un texto corto, y será mostrado como una marca de agua dentro del campo hasta que el elemento adquiera el foco.
- **required**. Este atributo declara al elemento como requerido para validación. Es un atributo booleano que no dejará que el formulario sea enviado hasta que una entrada para el campo sea provista.
- **pattern**. Este atributo especifica una expresión regular contra la cual la entrada será validada.
- **multiple**. Este es un atributo booleano que permite ingresar múltiples valores en el mismo campo (como múltiples cuentas de e-mail, por ejemplo). Los valores deben ser separados por coma.
- **form**. Este atributo asocia el elemento a un formulario. El valor provisto debe ser el valor del atributo **id** del elemento **<form>**.
- **list**. Este atributo asocia el elemento con un elemento **<datalist>** para mostrar una lista de posibles valores para el campo. El valor provisto debe ser el valor del atributo **id** del elemento **<datalist>**.

2.3. Elementos

HTML5 también incluye nuevos elementos que ayudan a mejorar y expandir formularios.

- **<datalist>**. Este elemento hace posible incluir una lista de opciones predefinidas que será mostrada en un elemento **<input>** como valores sugeridos. La lista es construida con el elemento **<option>** y cada opción se declara con los atributos **value** y **label**. Esta lista de opciones se relaciona con un elemento **<input>** por medio del atributo **list**.
- **<progress>**. Este elemento representa el estado en la evolución de una tarea (por ejemplo, una descarga).
- **<meter>**. Este elemento representa una medida, como el tráfico de un sitio web.
- **<output>**. Este elemento presenta un valor de salida para aplicaciones dinámicas.