UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

Desarrollo Web en HTML5, CSS3 y Javascript (avanzado)

Módulo 1:

HTML5 avanzado

Unidad 2:

DOM, formularios y API form

Objetivos:

Unidad 2: Que el alumno conozca las nuevas propiedades y atributos relacionados con el uso de formularios en HTML5.

Temario:

.: DOM
- .: Árbol de nodos
- .: Tipos de nodos
- .: Acceso directo a los nodos
- .: Acceso directo a los atributos HTML

.: Formularios
- .: Formularios web
  - .: El elemento form
  - .: El elemento input
    - .: Tipo email
    - .: Tipo search
    - .: Tipo url
    - .: Tipo tel
    - .: Tipo number
    - .: Tipo range
    - .: Tipo date
    - .: Tipo week
    - .: Tipo month
    - .: Tipo time
    - .: Tipo datetime
    - .: Tipo datetime-local
    - .: Tipo color
- .: Nuevos atributos
  - .: Atributo Placeholder
  - .: Atributo requiered
  - .: Atributo multiple
  - .: Atributo autofocus
  - .: Atributo pattern
  - .: Atributo form
- .: Nuevos elementos para formularios
  - .: Elemento datalist
  - .: Elemento progress
  - .: Elemento meter
  - .: Elemento output

CONSIGNAS PARA EL APRENDIZAJE COLABORATIVO

En esta Unidad los participantes se encontrarán con diferentes tipos de consignas que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

1. Los foros asociados a cada una de las unidades.
2. La Web 2.0.
3. Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen las actividades sugeridas y compartan en los foros los resultados obtenidos.

DOM

La creación del **Document Object Model** o **DOM** es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas.

DOM permite a los programadores web acceder y manipular las páginas XHTML como si fueran documentos XML. De hecho, DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.

A pesar de sus orígenes, DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript) y cuyas únicas diferencias se encuentran en la forma de implementarlo.

El DOM es el conjunto de objetos predefinidos que nos permite acceder a todos los elementos de una página y a ciertas características específicas del navegador.

Árbol de nodos

Una de las tareas habituales en la programación de aplicaciones web con JavaScript consiste en la manipulación de las páginas web. De esta forma, es habitual obtener el valor almacenado por algunos elementos (por ejemplo los elementos de un formulario), crear un elemento (párrafos, `<div>`, etc.) de forma dinámica y añadirlo a la página, aplicar una animación a un elemento (que aparezca/desaparezca, que se desplace, etc.).

Todas estas tareas habituales son muy sencillas de realizar gracias a DOM. Sin embargo, para poder utilizar las utilidades de DOM, es necesario "transformar" la página original.

Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular. Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla. El motivo por el que se muestra el funcionamiento de esta transformación interna es que condiciona el comportamiento de DOM y por tanto, la forma en la que se manipulan las páginas.

DOM transforma todos los documentos HTML en un conjunto de elementos llamados nodos, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos.

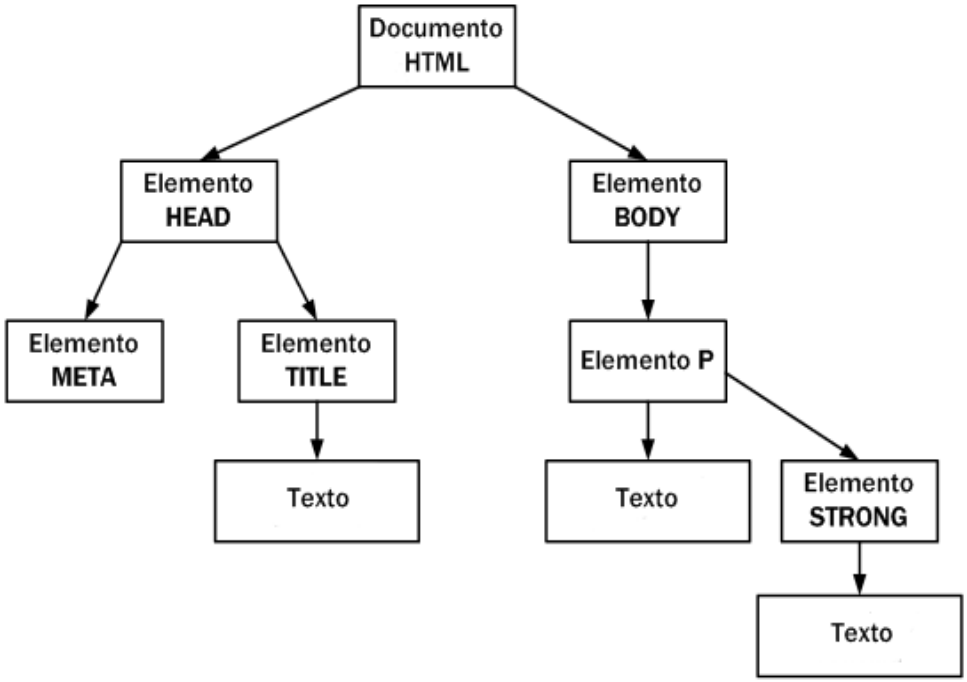
Por su aspecto, la unión de todos los nodos se llama "árbol de nodos".

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Ejemplo árbol de nodos</title>
</head>

<body>
<p>Esto es un ejemplo de <strong>Árbol de nodos</strong></p>
</body>
</html>
```

Se transforma en el siguiente árbol de nodos:



La raíz del árbol de nodos de cualquier página HTML siempre es la misma: un nodo de tipo especial denominado *"Documento"*.

La transformación automática de la página en un árbol de nodos siempre sigue las mismas reglas:

- Las etiquetas HTML se transforman en dos nodos: el primero es la propia etiqueta y el segundo nodo es hijo del primero y consiste en el contenido textual de la etiqueta.
- Si una etiqueta HTML se encuentra dentro de otra, se sigue el mismo procedimiento anterior, pero los nodos generados serán nodos hijo de su etiqueta padre.

Tipos de nodos

La especificación completa de DOM define **12 tipos de nodos**, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- **Document**, nodo raíz del que derivan todos los demás nodos del árbol.
- **Element**, representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- **Attr**, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par atributo=valor.
- **Text**, nodo que contiene el texto encerrado por una etiqueta XHTML.
- **Comment**, representa los comentarios incluidos en la página XHTML.

Los otros tipos de nodos existentes que no se van a considerar son DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation.

Acceso directo a los nodos

getElementsByTagName()

La función ***getElementsByTagName(nombreEtiqueta)*** obtiene todos los elementos de la página HTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

El siguiente ejemplo muestra cómo obtener todos los párrafos de una página HTML:

```
var parrafos = document.getElementsByTagName("p");
```

getElementsByName()

La función ***getElementsByName()*** busca los elementos cuyo atributo name sea igual al parámetro proporcionado. En el siguiente ejemplo, se obtiene directamente el único párrafo con el nombre indicado:

```
var parrafoEspecial = document.getElementsByName("especial");


<p name="prueba">...</p>
<p name="especial">...</p>
<p>...</p>
```

getElementById()

La función ***getElementById()*** es la más utilizada cuando se desarrollan aplicaciones web dinámicas. Se trata de la función preferida para acceder directamente a un nodo y poder leer o modificar sus propiedades.

La función ***getElementById()*** devuelve el elemento HTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo id debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

```
var cabecera = document.getElementById("cabecera");

<div id="cabecera">
  <a href="#" id="logo">...</a>
</div>
```

Acceso directo a los atributos HTML

Una vez que se ha accedido a un nodo, el siguiente paso consiste en acceder y/o modificar sus atributos y propiedades. Mediante DOM, es posible acceder de forma sencilla a todos los atributos HTML y todas las propiedades CSS de cualquier elemento de la página.

Los atributos HTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo HTML detrás del nombre del nodo.

El siguiente ejemplo obtiene de forma directa la dirección a la que enlaza un vínculo:

```
var enlace = document.getElementById("enlace");
alert(enlace.href); // muestra http://www.google.com

<a id="enlace" href="http://www.google.com">Enlace</a>
```

Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo style.

El siguiente ejemplo obtiene el valor de la propiedad margin de la imagen:

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);

<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio. Por ejemplo:

- **font-weight** se transforma en **fontWeight**
- **line-height** se transforma en **lineHeight**
- **border-top-style** se transforma en **borderTopStyle**
- **list-style-image** se transforma en **listStyleImage**

El único atributo HTML que no tiene el mismo nombre en HTML y en las propiedades DOM es el atributo **class**. Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento HTML. En su lugar, DOM utiliza el nombre **className** para acceder al atributo class de HTML:

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.class); // muestra "undefined"
alert(parrafo.className); // muestra "normal"

<p id="parrafo" class="normal">...</p>
```

FORMULARIOS

Formularios Web

La Web 2.0 está completamente enfocada en el usuario. Y cuando el usuario es el centro de atención, todo está relacionado con interfaces, en cómo hacerlas más intuitivas, más naturales, más prácticas, y por supuesto más atractivas. Los formularios web son la interface más importante de todas, permiten a los usuarios insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación. Durante los últimos años, códigos personalizados y librerías fueron creados para procesar formularios en la máquina del usuario.

HTML5 vuelve a estas funciones estándar agregando nuevos atributos, elementos y una API completa. Ahora la capacidad de procesamiento de información insertada en formularios en tiempo real ha sido incorporada en los navegadores y completamente estandarizada.

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="POST" action="#">
<input type="text" name="nombre" id="nombre" value="Ingresa tu nombre">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

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 automáticamente validados. 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, ningún valor tiene que ser especificado (su presencia es suficiente para activar su función).

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 es esperada desde el usuario. Los tipos disponibles hasta el momento eran el multipropósitos text (para textos en general) y solo unos pocos más específicos como password o submit.

HTML5 ha expandido las opciones incrementando de este modo 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 ingresado por el usuario.

Tipo email

Casi todo formulario en la web ofrece un campo para ingresar una dirección de email, 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 email 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 generado por el código del ejemplo será controlado por el navegador y validado como un email. 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.

Tipo search

El tipo search (búsqueda) no controla la entrada, es solo 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">
```

Tipo url

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

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

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 solo una indicación para el navegador en caso de que necesite hacer ajustes de acuerdo al dispositivo en el que la aplicación es ejecutada.

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

Tipo number

Como su nombre lo indica, el tipo number es sólo 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 le permitirá especificar valores entre 0 y 5 o entre 5 y 10.

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

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 es mostrado en pantalla como un puntero deslizable o un campo con flechas para seleccionar un valor entre los predeterminados, pero no existe un diseño estándar hasta el momento.

El tipo range usa los atributos min y max estudiados 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">
```

Tipo date

Este es otro tipo de campo que genera una nueva clase de control. En este caso fue incluido para ofrecer una mejor forma de ingresar 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 ingresado 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 el ingreso de este tipo de datos.

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

Tipo week

Este tipo de campo ofrece una interface similar a date, pero solo para seleccionar una semana completa. Normalmente el valor esperado tiene la sintaxis 2013-W20 donde 2013 es al año y 20 es el número de la semana.

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

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">
```

Tipo time

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

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

Tipo datetime

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

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

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">
```

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.

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

NUEVOS ATRIBUTOS

Algunos tipos de campo requieren de la ayuda de atributos, como los anteriormente detallados 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.

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 ingresar la información correcta. El valor de este atributo es presentado en pantalla por los navegadores dentro del campo, como una marca de agua que desaparece cuando el elemento es enfocado.

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

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 email, el navegador comprueba si la entrada es un email válido o no, pero validará la entrada si el campo está vacío. Cuando el atributo required es incluido, 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>
```

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 el ingreso de entradas múltiples en el mismo campo.

Los valores insertados deben estar separados por coma para ser válidos.

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

Atributo autofocus

Esta es una función que muchos desarrolladores aplicaban anteriormente utilizando el método focus() de Javascript. Este método era efectivo pero forzaba el foco sobre el elemento seleccionado, incluso cuando el usuario ya se encontraba posicionado en otro diferente. Este comportamiento era irritante pero difícil de evitar hasta ahora.

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>
```

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 estudiados 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">
```

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 nombre 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 name="formulario" id="formulario" method="get">
<input type="text" name="nombre" id="nombre">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

NUEVOS ELEMENTOS PARA FORMULARIOS

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>
```

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">
```

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. Vamos a usar <progress> en futuros ejemplos.

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.

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>.