
UTN
TÉCNICO UNIVERSITARIO EN SISTEMAS INFORMÁTICOS
PROGRAMACIÓN AVANZADA II

Nosotros

Comisiones y datos de contacto

Prof. Juan José Azar
juan.azar@gmail.com

Ayte. Ignacio Casales
casales.Ignacio@gmail.com

RESPETA A TUS COMPAÑEROS

ESTUDIA CON INTENSIDAD

JÚNTATE CON LOS QUE CONSIDERES MEJORES Y MAS EXPERIMENTADOS

“ESTO FUNCIONA” NO ES DONDE TE DETIENES, ES DONDE COMIENZAS

CONÉCTATE CON LOS DEMÁS, TRABAJA EN EQUIPO

APRENDE TÉCNICAS, NO HERRAMIENTAS

PERMANECE CURIOSO

PIDE AYUDA Y OFRECE AYUDA

AMA LO QUE HACES

DIVIÉRTETE!

Sobre esta materia

Cómo aprobarla?

· Aprobar los 2 (dos) parciales o recuperatorios con 6 (seis) o mas puntos en cada uno.

Cómo son los Recuperatorios?

· Hay una única instancia al final de la cursada.

· Todos los parciales pueden recuperarse en esta instancia.

Cómo es la Aprobación Directa?

· Todos los parciales deben ser aprobados con 8 (ocho) o mas puntos en cada uno.

· 75% de asistencia es requerida.

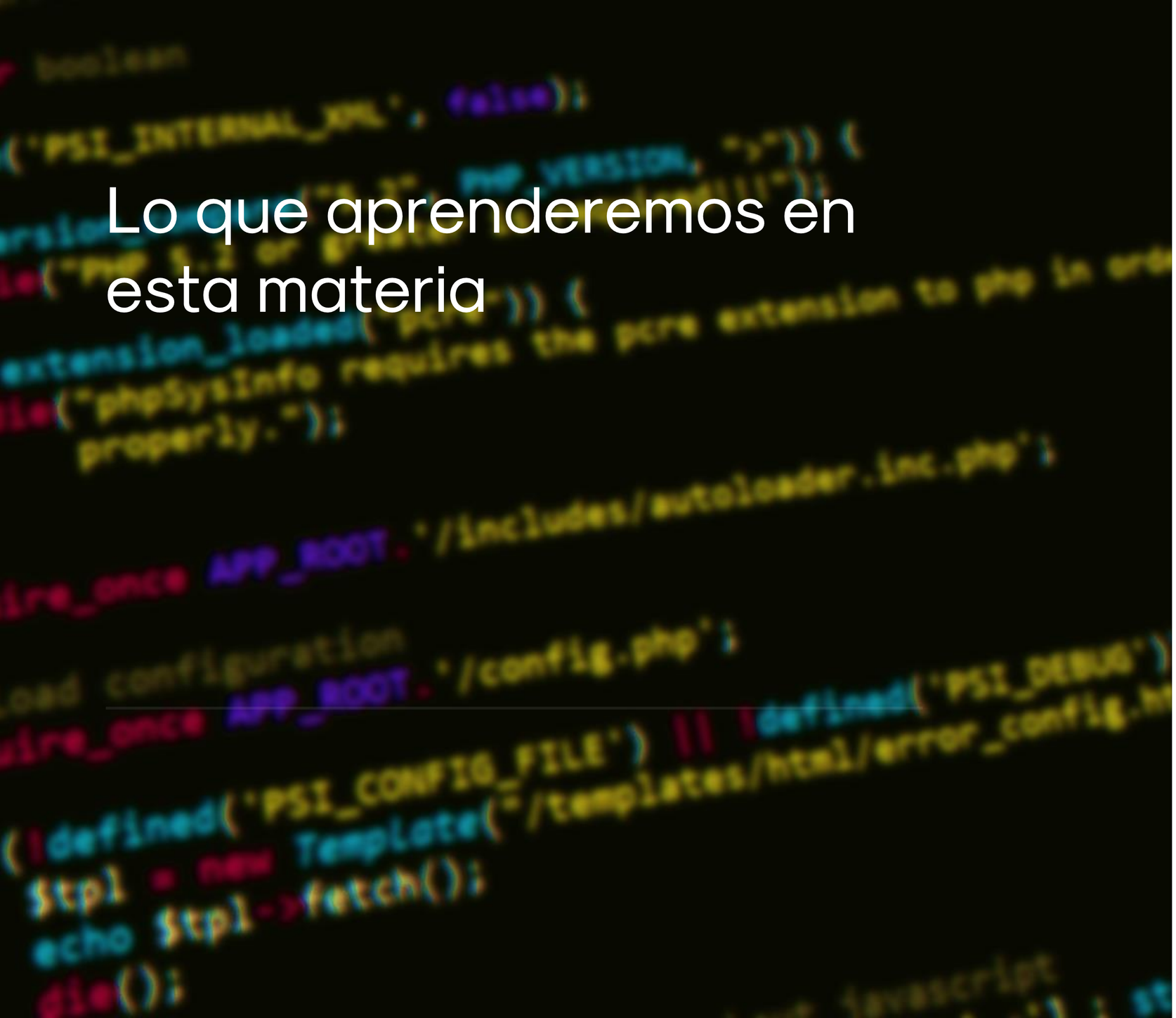
· El alumno podrá tener sólo 1 (uno) recuperatorio.

· El alumno no podrá tener materias correlativas sin aprobar.

Se puede repetir un Parcial?

· Si, solo 1 (una) chance para repetir un Parcial con el objetivo de obtener un puntaje mayor, pero el alumno no deberá tener recuperatorios previos.

· La nota quedará firme aunque ésta sea menor que la de la primera instancia.



Lo que aprenderemos en esta materia

FUNDAMENTALS

JavaScript, CSS, HTML5, DOM

WEB UI FRAMEWORK

Angular, NPM

INTEGRACION

Integración Angular Application con REST API

HTML Basics

HTML es el lenguaje de etiquetado para Páginas Web.

Con HTML puedes crear tu propio Web Site.

Qué es HTML?

HTML es el lenguaje de etiquetado para la creación de Páginas Web.

- HTML viene de Hyper Text Markup Language
- HTML describe la estructura de una Página Web
- HTML consiste en una serie de elementos
- Los elementos HTML le dicen al browser cómo mostrar el contenido
- Los elementos HTML están representados por etiquetas (**tags**)
- Las HTML tags etiquetan partes del contenido como por ejemplo "heading", "paragraph", "table", etc.
- Los browsers no muestran las tags HTML, pero las usan para renderizar el contenido de la página

index.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>HTML fundamentals</title>
    </head>
    <body>

        <h1>My First Heading</h1>

        <p>My first paragraph.</p>

        <!-- This is a comment -->

    </body>
</html>
```

Estructura básica de un documento HTML

- **<!DOCTYPE html>** define que el documento es de tipo HTML5
- **<html>** es el elemento raíz de una página HTML
- **<head>** contiene meta información sobre el documento
- **<title>** especifica el título del documento (title bar)
- **<body>** posee el contenido visible de la página
- **<h1>** define un encabezado grande
- ** ** define un párrafo
- **<!–** y **-->** apertura y cierre de código comentado

HTML Tags: **<tagname>**contenido…**</tagname>**

- Generalmente van de a pares, ej.: ** ** y ** **
- Cada una se conoce como *opening tag* y *closing tag*

```html
<body>
    <h1>This is heading 1</h1>
    <h2>This is heading 2</h2>
    <h3>This is heading 3</h3>

    <a href="https://google.com">This is a link to Google</a>

    <br>

    <img
    src="https://raw.githubusercontent.com/JuanAzar/UTN-LabIV/master/Common/Assets/logo-landscape.jpg"
    alt="logo-landscape image" title="UTN Mar del Plata" width="340" height="68">
</body>
```

Estructura básica de un documento HTML

<h1> al **<h6>** definen encabezados de mayor a menor tamaño respectivamente

- **<a>** define un hipervínculo. El atributo **href** indica la URL de destino y entre las tags se especifica el texto a mostrar en el hipervínculo
- **
** salto de línea en HTML, no lleva closing tag
- **<img>** inserta una imagen en el documento, no lleva closing tag y posee los siguientes atributos:

  src origen de la imagen, puede ser un archivo local o un archivo remoto

  alt texto alternativo que se muestra cuando la imagen no logra cargarse

  title texto que se muestra al hacer un **hover over** en la imagen

  width y **height** ancho y alto respectivamente para cambiar el tamaño de forma manual. Si no se especifica ninguno, la imagen se muestra en tamaño original

```html
<body>
    <h2>Unordered List example</h2>
    <ul>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
    </ul>

    <h2>Ordered List example</h2>
    <ol>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
    </ol>

    <h2>Description list example</h2>
    <dl>
        <dt>Coffee</dt>
        <dd>- black hot drink</dd>
        <dt>Milk</dt>
        <dd>- white cold drink</dd>
    </dl>
</body>
```

Trabajando con listas

- **`<ul>`** define una lista desordenada
- **`list-style-type`** es una propiedad CSS que define el tipo de viñeta
- **`<ol>`** define una lista ordenada
- **`type`** es un atributo que define el tipo de numeración
- **`<li>`** corresponde a un list item o elemento de lista
- **`<dl>`** define una lista de descripción
- **`<dt>`** corresponde a un término
- **`<dd>`** corresponde a la descripción de un término
- Las listas pueden anidarse entre sí
- Las listas pueden contener otros elementos HTML
- Se pueden usar las propiedades CSS **`float:left`** o **`display:inline`** para mostrar una lista de manera horizontal

```html
<body>
    <b>This text is bold</b>

    <br>

    <strong>This text is strong</strong>

    <br>

    <i>This text is italic</i>

    <br>

    <em>This text is emphasized</em>

    <h2>HTML <small>Small</small> Formatting</h2>

    <h2>HTML <mark>Marked</mark> Formatting</h2>

    <p>My favorite color is <del>blue</del> red.</p>

    <p>My favorite <ins>color</ins> is red.</p>

    <p>This is <sub>subscripted</sub> text.</p>

    <p>This is <sup>superscripted</sup> text.</p>
</body>
```

Dando formato al texto

- **<b>** define texto en negrita
- **** define texto importante
- **<i>** define texto en cursiva
- **** define texto enfatizado
- **<small>** define texto mas pequeño
- **<mark>** define texto resaltado
- **<del>** define texto tachado
- **<ins>** define texto subrayado
- **<sub>** define texto en subíndice
- **<sup>** define texto en superíndice

```
<body>

    <div>Hello World</div>

    <span>Hello World</span>

</body>
```

Elementos de bloque

- `<div>` define una sección en un documento (**block-level**). Este elemento es utilizado a menudo como contenedor de otros elementos HTML.

- `<span>` define una sección en un documento (**inline**). Este elemento es utilizado a menudo como contenedor de texto.

Nota: Ambos elementos se utilizan con atributos de tipo **style**, **class** y **id**.

Un elemento de tipo **block-level** comienza siempre en una nueva línea y ocupa todo el ancho disponible (se extiende hacia la izquierda y hacia la derecha todo lo que pueda). Ej.: `<address>` `<article>` `<aside>` `<dd>` `<div>` `<dl>` `<dt>` `<footer>` `<form>` `<h1>-<h6>` `<header>` `<hr>` `<li>` `<main>` `<nav>` `<ol>` `<p>` `<section>` `<table>` `<ul>`

Un elemento de tipo **inline** no comienza en una nueva línea y ocupa sólo el ancho que sea necesario. Ej.: `<a>` `<b>` `<br>` `<button>` `<cite>` `<code>` `<em>` `<i>` `<img>` `<input>` `<label>` `<q>` `<script>` `<select>` `<small>` `<span>` `<strong>` `<sub>` `<sup>` `<textarea>`

```html
<body>
    <table>
        <caption>Employee Information</caption>
        <thead>
            <tr>
                <th>Firstname</th>
                <th>Lastname</th>
                <th>Age</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Jill</td>
                <td>Smith</td>
                <td>50</td>
            </tr>
            <tr>
                <td>Eve</td>
                <td>Jackson</td>
                <td>94</td>
            </tr>
            <tr>
                <td>John</td>
                <td>Doe</td>
                <td>80</td>
            </tr>
        </tbody>
        <tfoot>
            <tr>
                <td colspan="3"> </td>
            </tr>
        </tfoot>
    </table>
</body>
```

Tablas

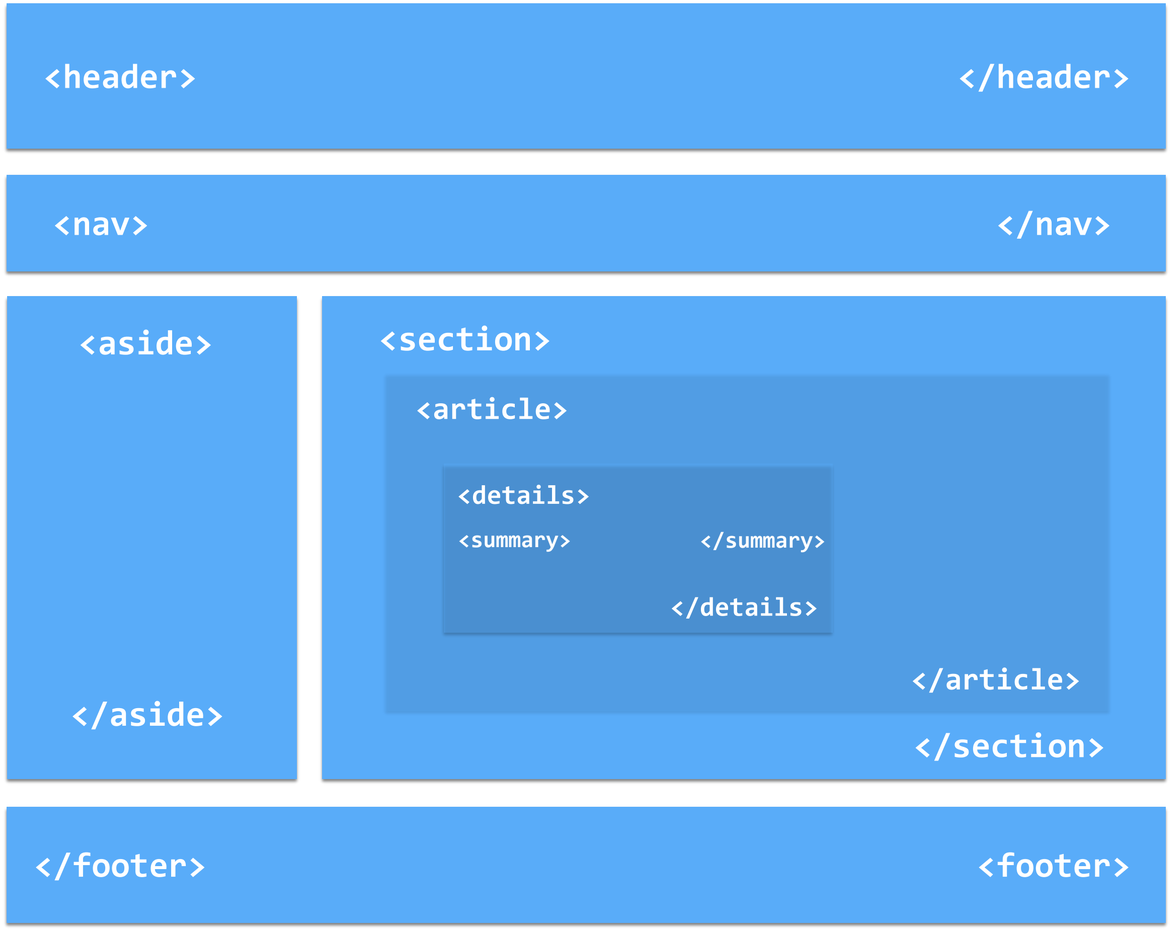
- **`<table>`** define una tabla
- **`<caption>`** define el título de una tabla
- **`<tr>`** define una fila de tabla
- **`<th>`** define una celda de encabezado de tabla
- **`<td>`** define una celda de tabla
- **`<thead>`** agrupa el contenido del **header** en una tabla
- **`<tbody>`** agrupa el contenido del **body** en una tabla
- **`<tfoot>`** agrupa el contenido del **footer** en una tabla

Nota: Los atributos **`colspan`** y **`rowspan`** permiten hacer que una columna o fila abarque mas de una columna o fila respectivamente

```html
<head>
    <title>HTML fundamentals</title>
    <base href="https://raw.githubusercontent.com/JuanAzar/UTN-LabIV/master/Common/Assets/">
    <link rel="stylesheet" href="mystyles.css">
    <meta charset="UTF-8">
    <script>
        function myFunction() {
            document.getElementById("demo").innerHTML = "Hello JavaScript!";
        }
    </script>
    <style>
        body {
            background-color: powderblue;
        }

        h1 {
            color: red;
        }

        p {
            color: blue;
        }
    </style>
</head>
<body>
    <h1>This is a heading</h1>

    <p id="demo">This is a paragraph</p>

    <button onclick="myFunction()">Click here</button>

    <br><br>

    <img src="logo-portrait.jpg" width="230" height="383">
</body>
```

Head

- **`<head>`** define información sobre el documento
- **`<title>`** especifica el título del documento (title bar)
- **`<base>`** define una dirección/destino por defecto para todos los links relativos de la página
- **`<link>`** especifica una relación entre el documento y un recurso externo
- **`<meta>`** define metadata sobre un documento HTML
- **`<script>`** define un script client-side
- **`<style>`** especifica estilos para un documento



Layout

- **<header>** define el encabezado para un documento o sección
- **<nav>** define un contenedor para links de navegación (ej.: menú)
- **<section>** define una sección en un documento
- **<article>** define un artículo independiente y auto contenido
- **<aside>** define contenido que no forma parte del contenido principal (ej.: sidebar)
- **<footer>** define el pie de un documento o sección
- **<details>** especifica detalles adicionales. Permite expandir o colapsar su contenido sin necesidad de agregar lógica adicional (*no soportado en IE y Edge*).
- **<summary>** especifica un encabezado para el elemento **<details>**

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	double quotation mark	&quot;	&#34;
'	single quotation mark	&apos;	&#39;
á	small letter a with accent	&aacute;	&#225;
é	small letter e with accent	&eacute;	&#233;
í	small letter i with accent	&iacute;	&#237;
ó	small letter o with accent	&oacute;	&#243;
ú	small letter u with accent	&uacute;	&#250;
ñ	n tilde	&ntilde;	&#241;
©	copyright	&copy;	&#169;
®	registered trademark	&red;	&#174;

Entidades

- Algunos caracteres son reservados en HTML
- Las entities se utilizan para poder mostrar caracteres reservados en HTML
- Se utiliza **`&entityName`** o **`&#entityNumber`**. Ej.: para mostrar (<) se usa **`<`** o **`&#60;`**
- Ventajas de usar un entity name: Es fácil de recordar
- Desventajas de usar un entity name: Los Browsers pueden no soportar todos los entity names, pero el soporte de entity numbers es muy bueno.

```html
<body>
    <form action="action.php" method="post">
        <input type="text" name="firstName" placeholder="First Name">
        <input type="text" name="lastName" value="Your Last Name">
        <input type="password" name="password" placeholder="Enter password">
        <br>
        <input type="radio" name="answer" value="yes" checked> Yes
        <input type="radio" name="answer" value="no"> No
        <input type="radio" name="answer" value="na"> N/A
        <br>
        <select name="cars">
          <option value="peugeot">Peugeot</option>
          <option value="chevrolet">Chevrolet</option>
          <option value="ford">Ford</option>
          <option value="volkswagen">Volkswagen</option>
        </select>
        <br>
        <textarea name="comments" cols="50" rows="10"></textarea>
        <br>
        <input type="checkbox" name="vehicle1" value="Bike"> I have a bike<br>
        <input type="checkbox" name="vehicle2" value="Car"> I have a car
        <br>
        <button type="submit">Send</button>
        <button type="reset">Reset</button>
        <button type="button" onclick="alert('Hello World!')">Say Hello</button>
      </form>
</body>
```

Formularios

- **<form>** define un formulario para recolectar información.
- **<input>** depende del atributo type para determinar cómo se muestra:
  - **text** especifica un input text de una sola línea
  - **password** define un input de tipo password
  - **radio** define un radio button. Aquellos radio buttons con mismo **name** trabajan en conjunto
  - **checkbox** permite elegir cero o mas opciones
- **<select>** define un drop-down list. Contiene varios **<option>**
- **<textarea>** define un input de tipo multilínea
- **<submit>** envía el contenido de los inputs de un form a un **form-handler**
- **<reset>** restablece el contenido de los inputs de un form
- **<button>** botón genérico al cual se le puede añadir acción

ECMAScript - JavaScript

JavaScript es el lenguaje de programación de HTML y la Web inventado en 1995.

ECMAScript es el nombre oficial del lenguaje.

ECMA (*European Computer Manufacturer's Association*) es un estándar para *scripting languages*

A partir de 2015 ECMAScript se llama según el año (Ej.: ECMAScript 2015)

Qué es ECMAScript?

ECMAScript es el lenguaje de scripting para páginas Web.

- Trabaja del lado del cliente (*client-side)*, es decir en el browser.
- Puede cambiar contenido HTML
- Puede cambiar valores de atributos HTML
- Puede cambiar estilos CSS
- En síntesis puede cambiar cualquier cosa dentro del **DOM** (***Document Object Model***)

ECMAScript Basics

Actualmente ECMAScript se encuentra en la versión 2018.

Veremos las versiones principales con sus diferencias y su sintaxis básica

ECMAScript Versiones

ECMAScript y sus versiones mas relevantes

- **JavaScript ES5(2009)**: Soporte para, **`strict mode`**, **`JSON support`**, **`String.trim()`**, **`Aray.isArray()`**, **`Array iteration methods`**
- **JavaScript ES6 o ECMAScript 2015**: Soporte para **`let`** y **`const`**, **`default parameter values`**, **`Array.find()`**, **`Array.findIndex()`**
- **ECMAScript 2016:** Soporte para **`exponential operator (**)`**, **`Array.prototype.includes`**
- **ECMAScript 2017:** Soporte para **`string padding`**, **`new Object proeprtie`**s, **`async functions`**, **`shared memory`**
- **ECMAScript 2018:** Soporte para **`rest/spread properties`**, **`async itearion`**, **`Promise.finally()`**, agregados a **`RegExp`**

```html
<html>
    <head>
        <title>JavaScript Fundamentals</title>
        <script src="myScript.js"></script>
        <script src="https://www.w3schools.com/js/myScript1.js"></script>
    </head>
    <body>
        <p id="demo"></p>

        <button type="button" onclick="myFunction()">Say Hello</button>

        <script>
            var x, y, z;
            x = 5;
            y = 6;
            z = x + y;

            document.getElementById("demo").innerHTML =
            "The value of z is " + z + ".";

            function myFunction()
            {
                document.getElementById("demo").innerHTML =
                "Hello World!.";
            }

            var intValue = parseInt("1234");
            var floatValue = parseFloat("12.34");
            var parseError = parseInt("John");

            //Single line Commented

            /*
            Multiline commented
            block
            */
        </script>
    </body>
</html>
```

Sintaxis básica de JavaScript

- Debe ser insertado entre tags **<script>** y **</script>**
- Las funciones JavaScript pueden ser llamadas a través de **eventos**, por ejemplo cuando un usuario hace click en un botón
- Los scripts pueden ir dentro del **<head>** o el **<body>** y se pueden colocar todos los scripts que queramos
- Los scripts pueden colocarse en archivos externos. Esto mejora entre otras cosas que las páginas carguen más rápido al ser cacheados. Ej.: **myScript.js**
- **getElememntById()**: permite acceder a un elemento HTML a través de su **id** y luego ser manipulado.
- Se puede comentar una línea con **//** y en bloque con **/* */**
- Las variables pueden contener letras, números, underscores (_) y signos de dólar $
- Las variables no pueden comenzar con un número y son case sensitive
- Es case sensitive
- Una variable que no tiene valor, tendrá el valor **undefined**
- **parseInt() y parseFloat()** permiten convertir texto en **integer** o **float**

```html
<html>
    <head>
        <title>JavaScript Fundamentals</title>
    </head>
    <body>
        <h1>My First Web Page</h1>
        <p>My First Paragraph</p>

        <p id="demo"></p>

        <script>
            document.getElementById("demo").innerHTML = 5 + 6;

            document.write(5 + 6);

            console.log("Logging some data");
        </script>

        <button type="button" onclick="document.write('This is JavaScript!')">Write</button>
        <button type="button" onclick="window.alert('This is JavaScript!')">Alert</button>
    </body>
</html>
```

Output

- **`innerHTML:`** utilizando **`getElementById()`** podemos acceder a un elemento HTML y con la propiedad **`innerHTML`** acceder a su contenido HTML. Esta es una manera tradicional de mostrar información
- **`document.write()`**: permite mostrar contenido, se usa mas comúnmente para propósitos de testing. Si usamos **`document.write()`** luego de que el documento HTML se haya cargado, se elimina todo el HTML existente.
- **`window.alert():`** Permite disparar un **`alert box`** para mostrar información.
- **`console.log():`** es utilizado para debugging y permite mostrar por consola determinada información. Es muy utilizado en el proceso de desarrollo.

Operadores Aritméticos

Operador	Nombre	Ejemplo	Resultado
+	Suma	x + y	Suma de x e y
-	Resta	x - y	Diferencia entre x e y
*	Multiplicación	x * y	Producto de x e y
/	División	x / y	Cociente de x dividido y
%	Módulo	x % y	Resto de x dividido y
**	Potenciación	x ** y	Resultado de elevar x a la y potencia

Operadores de Asignación

Asignación	Equivalente	Descripción
x = y	x = y	La variable de la izquierda recibe el valor de la expresión de la derecha
x += y	x = x + y	Suma de x e y con asignación en x
x -= y	x = x - y	Resta entre x e y con asignación en x
x *= y	x = x * y	Multiplicación de x e y con asignación en x
x /= y	x = x / y	Cociente x dividido y con asignación en x
x %= y	x = x % y	Resto de x dividido y con asignación en x

Operadores de Comparación

Operador	Nombre	Ejemplo	Resultado
==	Igualdad	x == y	Retorna true si x es igual a y
===	Identidad	x === y	Retorna true si x es igual a y, y además son del mismo tipo
!=	Desigualdad	x != y	Retorna true si x no es igual a y
<>	Desigualdad	x <> y	Retorna true si x no es igual a y
!==	No Identidad	x !== y	Retorna true si x no es igual a y, o si no son del mismo tipo
>	Mayor	x > y	Retorna true si x es mayor que y
<	Menor	x < y	Retorna true si x es menor que y
>=	Mayor o igual	x >= y	Retorna true si x es mayor o igual que y
<=	Menor o igual	x <= y	Retorna true si x es menor o igual que y

Operadores de Incremento y Decremento

Operador	Nombre	Descripción
++x	Pre-incremento	Incrementa x en 1, luego retorna x
x++	Post-incremento	Retorna x, luego incrementa x en uno
--x	Pre-decremento	Decrementa x en 1, luego retorna x
x--	Post-decremento	Retorna x, luego decrementa x en uno

Operadores Lógicos

Operador	Nombre	Ejemplo	Resultado
`&&`	And	`x && y`	True si `x` e `y` son true
`||`	Or	`x || y`	True si `x` o `y` es true
`!`	Not	`!x`	True si `x` no es true

Operadores de Strings

Operador	Nombre	Ejemplo	Resultado
`+`	Concatenación	`txt1 + txt2`	Concatenación de `txt1` y `txt2`
`+=`	Asignación de concatenación	`txt1 += txt2`	Agrega `txt2` a `txt1`

Operadores de Tipo

Operador	Descripción
`typeOf`	Retorna el tipo de una variable
`instanceOf`	Retora true si un objecto es una instancia de un tipo específico

```
<script>
    var length = 16;

    var lastName = "Johnson";

    var person = { firstName :"John", lastName : "Doe"};

    //Output: 16Volvo
    var a = "16" + "Volvo";
    console.log(a);

    //Output: 16Volvo
    var b = 16 + "Volvo";
    console.log(b);

    //Output: Volvo16
    var c = "Volvo" + 16;
    console.log(c);

    //Output: 20Volvo
    var d = 16 + 4 + "Volvo";
    console.log(d);

    //Output: Volvo164
    var e = "Volvo" + 16 + 4;
    console.log(e);

    var carName1 = "Volvo XC60";
    var carName2 = 'Volvo XC60';

    var answer1 = "It's alright";
    var answer2 = "He is called 'Johnny'";
    var answer3 = 'He is called "Johnny"';
</script>
```

Tipos de Datos

- Sumando número y string, se tratará como la concatenación de dos strings
- Concatenando un string a un número, se tratará como la concatenación de dos strings
- JavaScript evalúa las expresiones de izquierda a derecha, por lo tanto diferentes secuencias pueden producir diferentes resultados.
- Sumar dos números y un string, producirá la suma algebraica y posteriormente la concatenación
- Concatenar un string a la suma de dos números, producirá la concatenación de los tres términos
- Los strings pueden declararse con comillas simples o dobles
- Se pueden poner comillas simples dentro de dobles y viceversa

```html
<body onload="myFunction1()">
    <script>
        function myFunction1() {
            alert("Hello World!")
        }

        function myFunction2() {
            console.log("Hello World!");
        }
    </script>

    <p onmouseover="myFunction1()">I love JavaScript!! Best language ever!!</p>

    <form onsubmit="myFunction1()" onreset="myFunction2()">
        KeyUp: <input type="text" name="a" onkeyup="myFunction1()"/>
        <br><br>
        Change: <input type="text" name="b" onchange="myFunction1()"/>
        <br><br>
        Select: <input type="text" name="c" onselect="myFunction1()"/>
        <br><br>
        Focus: <input type="text" name="d" onfocus="myFunction2()"/>
        <br><br>
        Blur: <input type="text" name="d" onblur="myFunction2()"/>
        <br><br>
        <button type="button" onclick="myFunction1()">Click</button>
        <button type="submit">Submit</button>
        <button type="reset">Reset</button>
        <br><br>
        <span>This is a span with an onclick event bound to a function</span>
        <div id="myDiv">This div has a onclick function bound to it</div>
    </form>
    <script>
        document.getElementById("myDiv").onclick = myFunction1;

        var element = document.getElementsByTagName('span')[0];
        element.addEventListener('click', myFunction1, false);
    </script>
</body>
```

Funciones y Eventos

- Las funciones pueden llamarse entre sí o al producirse un evento
- Un evento detecta la acción de un usuario y permite disparar una acción determinada
- Tipos de manejadores de eventos: en línea, propiedad y método
  `AddEventListener()`
- Eventos comunes:

  `onLoad:` Terminar de cargar una página o frame (entrar)

  `onMouseOver:` Pasar el mouse por encima de un elemento

  `onMouseOut:` Quitar el mouse de encima del elemento

  `onMouseMove:` Mover el mouse sobre el documento

  `onKeyUp:` Presionar una tecla

  `onClick:` Hacer click con el mouse

  `onChange:` Modificar texto en un control de edición. Sucede al perder el foco

  `onSelect:` Seleccionar texto en un control de edición

  `onFocus:` Situar el foco en un control

  `onBlur:` Perder el foco un control

  `onSubmit:` Enviar un formulario

  `onReset:` Restablecer un formulario

```html
<script>
    function getById() {
        document.getElementById('myTextBox').value = 'Hello World!';
    }

    function getByName() {
        document.getElementsByName('firstName')[0].value = 'Hello World';
    }

    function getBygetByClassName() {
        document.getElementsByClassName('formClass')[0].value = 'Hellow World!';
    }

    function getByTagName() {
        var htmlCollection = document.getElementsByTagName('span');

        var elements = Array.from(htmlCollection);

        elements.forEach(function(element) {
            element.innerHTML = 'Hello World!';
        });
    }

    function getBySelector() {
        document.querySelector('h1, h2').innerHTML = 'Hello World';
    }

    function getBySelectorAll() {
        var htmlCollection = document.querySelectorAll('h2, h3');

        var elements = Array.from(htmlCollection);

        elements.forEach(function(element) {
            element.innerHTML = 'Hello World!';
        });
    }
</script>
```

Referenciando elementos

- getElementById(): Obtiene un elemento a través de su id. Los id deberían ser únicos para cada elemento, pero si hubiese mas de uno con el mismo id, este método retorna el primero
- **`getElementsbyName():`** Obtiene una colección de elementos que tienen el **`name`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos y poder iterar sobre estos
- **`getElementsByClassName():`** Obtiene una colección de elementos que tienen el **`name`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos
- **`getElementsByTagName():`** Obtiene una colección de elementos que corresponden al **`tag`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos. Si se especifica el parámetro '*' se retorna todos los elementos del documento
- **`querySelector():`** Obtiene el primer elemento que coincide con el selector CSS especificado. Se pueden seleccionar elementos HTML a través de **`id`**, **`class`**, **`type`**, **`attribute`**, **`attribute values`**, etc.
- **`querySelectorAll()`**: Similar al anterior, solo que en lugar de retornar la primer ocurrencia, retorna todas

```html
<body onload="myFunction1()">
    <script>
        function myFunction1() {
            alert("Hello World!")
        }

        function myFunction2() {
            console.log("Hello World!");
        }
    </script>

    <p onmouseover="myFunction1()">I love JavaScript!! Best language ever!!</p>

    <form onsubmit="myFunction1()" onreset="myFunction2()">
        KeyUp: <input type="text" name="a" onkeyup="myFunction1()"/>
        <br><br>
        Change: <input type="text" name="b" onchange="myFunction1()"/>
        <br><br>
        Select: <input type="text" name="c" onselect="myFunction1()"/>
        <br><br>
        Focus: <input type="text" name="d" onfocus="myFunction2()"/>
        <br><br>
        Blur: <input type="text" name="d" onblur="myFunction2()"/>
        <br><br>
        <button type="button" onclick="myFunction1()">Click</button>
        <button type="submit">Submit</button>
        <button type="reset">Reset</button>
        <br><br>
        <span>This is a span with an onclick event bound to a function</span>
        <div id="myDiv">This div has a onclick function bound to it</div>
    </form>
    <script>
        document.getElementById("myDiv").onclick = myFunction1;

        var element = document.getElementsByTagName('span')[0];
        element.addEventListener('click', myFunction1, false);
    </script>
</body>
```

Funciones y Eventos

- Las funciones pueden llamarse entre sí o al producirse un evento
- Un evento detecta la acción de un usuario y permite disparar una acción determinada
- Tipos de manejadores de eventos: en línea, propiedad y método
  `AddEventListener()`
- Eventos comunes:

  `onLoad:` Terminar de cargar una página o frame (entrar)

  `onMouseOver:` Pasar el mouse por encima de un elemento

  `onMouseOut:` Quitar el mouse de encima del elemento

  `onMouseMove:` Mover el mouse sobre el documento

  `onKeyUp:` Presionar una tecla

  `onClick:` Hacer click con el mouse

  `onChange:` Modificar texto en un control de edición. Sucede al perder el foco

  `onSelect:` Seleccionar texto en un control de edición

  `onFocus:` Situar el foco en un control

  `onBlur:` Perder el foco un control

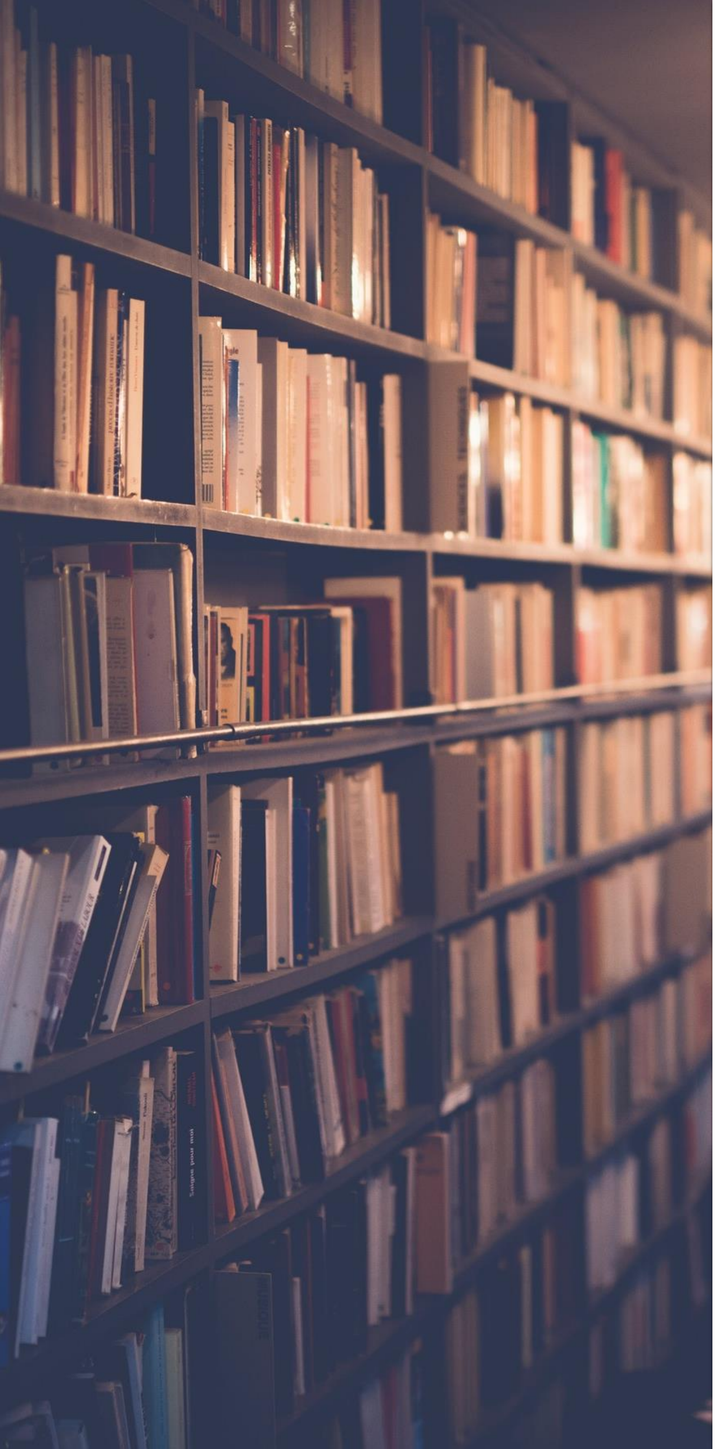
  `onSubmit:` Enviar un formulario

  `onReset:` Restablecer un formulario

```html
<script>
    function getById() {
        document.getElementById('myTextBox').value = 'Hello World!';
    }

    function getByName() {
        document.getElementsByName('firstName')[0].value = 'Hello World';
    }

    function getBygetByClassName() {
        document.getElementsByClassName('formClass')[0].value = 'Hellow World!';
    }

    function getByTagName() {
        var htmlCollection = document.getElementsByTagName('span');

        var elements = Array.from(htmlCollection);

        elements.forEach(function(element) {
            element.innerHTML = 'Hello World!';
        });
    }

    function getBySelector() {
        document.querySelector('h1, h2').innerHTML = 'Hello World';
    }

    function getBySelectorAll() {
        var htmlCollection = document.querySelectorAll('h2, h3');

        var elements = Array.from(htmlCollection);

        elements.forEach(function(element) {
            element.innerHTML = 'Hello World!';
        });
    }
</script>
```

Referenciando elementos

- getElementById(): Obtiene un elemento a través de su id. Los id deberían ser únicos para cada elemento, pero si hubiese mas de uno con el mismo id, este método retorna el primero
- **`getElementsbyName():`** Obtiene una colección de elementos que tienen el **`name`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos y poder iterar sobre estos
- **`getElementsByClassName():`** Obtiene una colección de elementos que tienen el **`name`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos
- **`getElementsByTagName():`** Obtiene una colección de elementos que corresponden al **`tag`** especificado. Se puede usar la propiedad **`length`** para determinar la cantidad de elementos devueltos. Si se especifica el parámetro '*' se retorna todos los elementos del documento
- **`querySelector():`** Obtiene el primer elemento que coincide con el selector CSS especificado. Se pueden seleccionar elementos HTML a través de **`id`**, **`class`**, **`type`**, **`attribute`**, **`attribute values`**, etc.
- **`querySelectorAll()`**: Similar al anterior, solo que en lugar de retornar la primer ocurrencia, retorna todas



Bibliografía y recursos

HTML Basics, JavaScript, CSS : https://www.w3schools.com/

Campus: http://campus.mdp.utn.edu.ar/course/edit.php?id=169

GitHub Repository: https://github.com/JuanAzar/UTN-Adv2


¿Preguntas?


END
GRACIAS POR ACOMPAÑARNOS
EN ESTE CAMINO