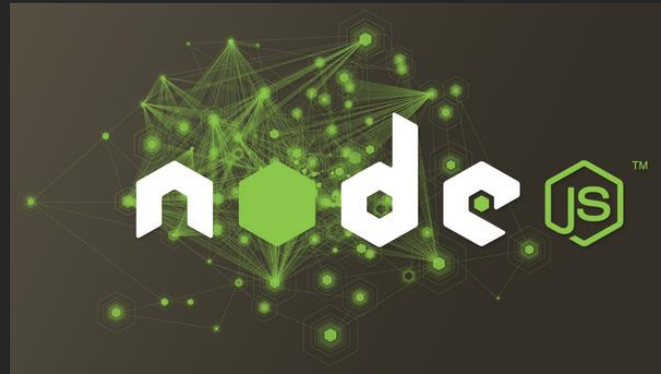


# Programa de Entrenamiento Intensivo

*Desarrollador Web Full Stack*

Marzo de 2019



Día 01 – HTML, CSS y JS

# Sobre el Instructor

**Becker Portilla (bportilla@baufest.com)**

Estudiante Ingeniería en Sistemas – UTN FRBA

*.Net Technical Expert*

5 Años en Baufest

Principales clientes en los que trabajé...

- *SIVD (Sistema Integrado de Venta Directa) - ISBAN*
- *PISCYS (Sistema de Suscripción y Afiliación) - ASOCIART*

# Sobre el Instructor

**Carla Eliana Vargas (cevargas@baufest.com)**

Tecnicatura en programación – UTN Avellaneda

*.Net Developer*

1 Año en Baufest

Principales clientes en los que trabajé...

- *Soporte/Factory – San Antonio*
  - *Dreyfus*
- *SuscripcionesGoogle/Autogestión - La Nación*

# Sobre el Instructor

**Aníbal Ezequiel Montalti (amontalti@baufest.com)**

Ingeniería en Informática - FIUBA

*.Net Developer*

2 años en Baufest

Principales clientes en los que trabajé...

- *Proyecto Interno – Baufest*
- *Gestión de macros – Johnson & Johnson*
- *PISCYS (Sistema de Suscripción y Afiliación) - ASOCIART*

# Objetivos del Módulo

1. Entender la **importancia del Front-End** para el correcto desarrollo de software
  2. Conocer el lenguaje **HTML** para realizar el **maquetado** de una página web
  3. Aprender el lenguaje **CSS** para imprimir **estilo y presentación**
  4. Conocer **Javascript** para darle **comportamiento** a una página web
- 
- ✓ **Combinar** HTML, CSS y Javascript para desarrollar páginas y aplicaciones web atractivas e interactivas
  - ✓ **Ejercitar** lo aprendido utilizando herramientas utilizadas diariamente en .Net

# Agenda – Por la mañana



- 09:00 hs- 10:30hs
  - Qué es HTML: Conceptos Básicos
  - Estructura del Documento HTML
  - Concepto de Semántica
  - Etiquetas Básicas
  - Formularios
- 10:30-10:50 hs Coffee Break
  - Laboratorio HTML básico
- 12:00-12:45 hs Almuerzo



# Agenda – Por la tarde

- 13:00 hs- 15:00hs
  - CSS
  - Maquetado por Tabla
  - Maquetado por Div
- 15:00-15:20 hs Coffee Break
- 15:20 hs- 18:00hs
  - Introducción a JavaScript Básico
  - Laboratorio sobre el ejercicio anterior
- 18:00 hs Fin del día



# Sobre ustedes

## **Queremos conocerlos un poco...**

- Nombre
- Carrera que estudian
- Experiencia profesional
- Banda favorita
- Hobbies o gustos...
- Etc.



```
sudo apt-get install code
```

# Separación de Estructura, Presentación y Comportamiento

## ¿Qué es una Página Web?

- Es un documento que contiene texto, sonido, video, imágenes, enlaces a otras páginas y es interpretada por navegadores web o browsers.
- Utilizadas en la Word Wide Web.

## ➤ Separación de Estructura, Presentación y Comportamiento

### 1. Estructura

- Documento HTML
- Contiene la información de la página
- Markup Semántico: Los tags o etiquetas no proveen presentación, proveen significado

### 2. Presentación

- Hoja de estilo CSS
- Contiene cómo se va a presentar o ver la información del documento HTML

### 3. Comportamiento

- Archivo JS
- Contiene sentencias en Javascript que manipulan los elementos HTML para que se comporten de determinada manera según los estímulos del usuario

# HTML

- ✓ Lenguaje HTML y Conceptos Básicos
- ✓ Estructura del documento HTML
- ✓ Elementos HTML Básicos



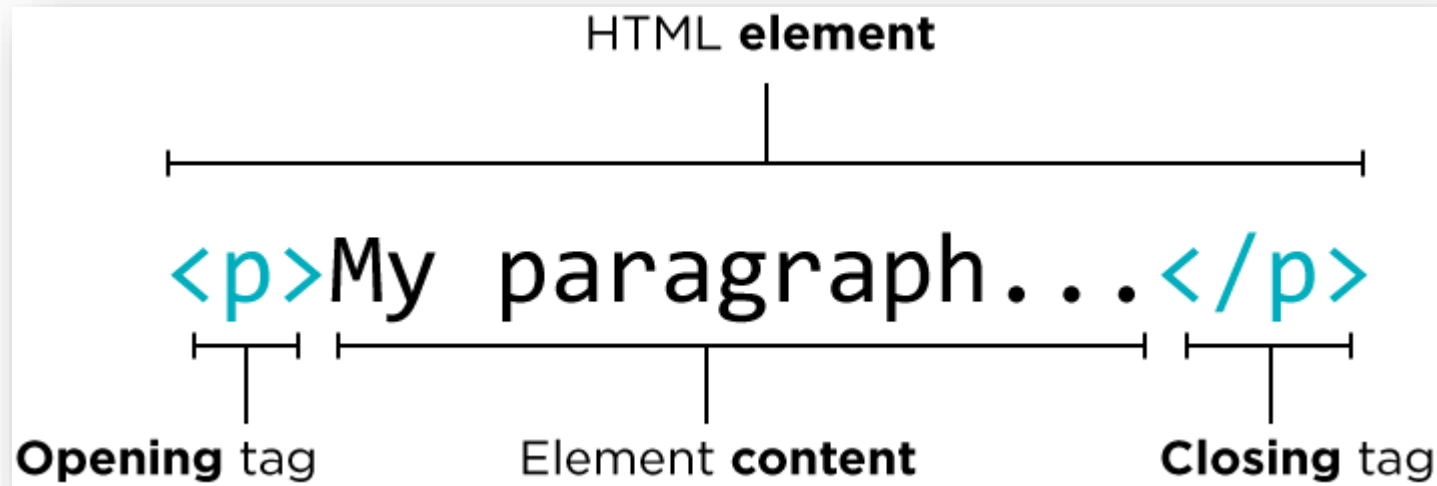
# Lenguaje HTML y Conceptos Básicos

## ¿Qué es HTML?

- HTML, siglas de **H**yper**T**ext **M**arkup **L**anguage: «lenguaje de marcado de hipertexto»
- Se utiliza principalmente para la elaboración de páginas web y aplicaciones Windows 8
- **Se escribe en forma de etiquetas**
- Creado y administrado por el W3C (World Wide Web Consortium).

## ¿Qué es una etiqueta o tag HTML?

- Palabra reservada encerrada entre los símbolos < y >
- Junto con su **contenido**, **atributos** y **etiqueta de cierre** forman un Elemento HTML



# Lenguaje HTML y Conceptos Básicos

## Elementos HTML

### 1. Atributos

- Es un par nombre="valor"
- Se ubican en la etiqueta de inicio
  - <nombre\_elemento atributo1="valor" atributo2="valor">...
- El nombre no puede tener espacios, el valor si
- Hay ya definidos o se pueden crear nuevos según las necesidades. Los nuevos se crean solo para almacenar información, ya que los navegadores no sabrán interpretarlos

```
<a href="http://www.google.com" id="ir-google">Ir a Google</a>
```

```
<input type="radio" value="Curso" name="radio-curso"/>
```

### 2. Contenido

- Si tiene contenido, debe tener una etiqueta de inicio y una etiqueta de cierre :
  - <nombre\_elemento>...</nombre\_elemento>
- Si no tiene contenido, es una sola etiqueta que se "autocierra" con el símbolo /
  - <nombre\_elemento />

## Comentarios

- Se pueden incluir cualquier texto dentro de los comentarios:
- <!-- comentario -->

# Elementos HTML Básicos

## <HTML>

- Define el inicio del documento HTML

## <HEAD>

- Define la cabecera del documento HTML
- La cabecera contiene información no visible para el usuario pero que es necesaria para una buena interpretación de parte del navegador
  - <TITLE>
    - Es el título de la página, que se ve en la ventana del navegador o en la barra de tareas.
  - <LINK>
    - Se utiliza para hacer referencia a las hojas de estilo CSS utilizados por la página

## <BODY>

- Define la contenido del documento HTML
- Dentro de esta etiqueta se incluirá todos los elementos e información que se mostrará por pantalla.
  - <SCRIPT>
    - Se utiliza para incrustar código Javascript o hacer referencia a un archivo externo de Javascript

# Elementos HTML Básicos

## <A>

- Link a un sitio web o a una página del sitio web
- **Atributos**
  - href : se define la url a donde apunta el link
- ej.: `<a href="www.google.com.ar">Google</a>`

## <IMG>

- Indica una imagen a mostrar
- **Atributos**
  - src: ruta a la imagen
- ej.: ``

## <TABLE>

- Define una tabla
- **<TR>** Define una fila
- **<TD>** Define una celda
- **<TH>** Define un header o cabecera de columna.

# Elementos HTML Básicos

## <INPUT>

- Define un control en el cual el usuario puede ingresar información o datos
- **No tiene contenido**, por lo que es una sola etiqueta que se "auto cierra"
- Hay de varios tipos y varían mucho
- El tipo se indica con el atributo **type**
- Tipos más comunes:
  - text (por defecto)
  - password
  - checkbox
  - Radio (deben tener el mismo atributo name)
  - button
  - date
  - file
  - hidden
  - image
  - submit



# Elementos HTML Básicos

## <BUTTON>

- Define un botón, que es usado para ejecutar algún script en javascript
- Dentro del elemento se puede incluir contenido como texto o imágenes, lo que lo diferencia de los botones creados con `<input type="button">`
- El tipo se indica con el atributo **type**:
  - button
  - reset
  - submit

```
<button type="button" onclick="msg()">Click Me!</button>
```

# Elementos HTML Básicos

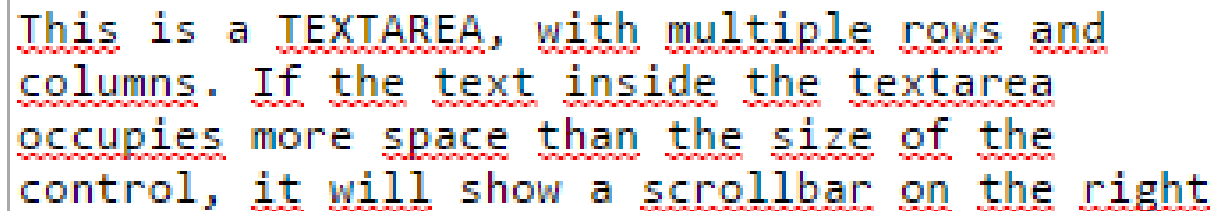
## <TEXTAREA>

- Define un control de ingreso de texto amplio y de varias líneas
- El tamaño del control se puede especificar con los atributos **cols** (cantidad de caracteres horizontalmente) y **rows** (cantidad de líneas verticalmente)

```
<textarea rows="4" cols="60">
```

This is a TEXTAREA, with multiple rows and columns. If the text inside the textarea occupies more space than the size of the control, it will show a scrollbar on the right side of the control to scroll to the hidden text.

```
</textarea>
```

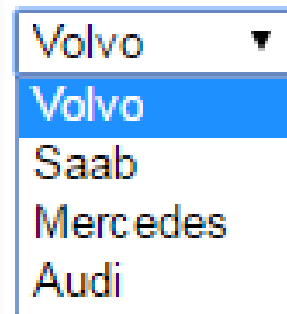


# Elementos HTML Básicos

## <SELECT>


- Define una lista desplegable (dropdown list) con un número limitado de alternativas
- Contiene elementos **option** dentro para cada una de las opciones o alternativas disponibles


```
<select>  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="mercedes">Mercedes</option>  
  <option value="audi">Audi</option>  
</select>
```





# Formularios


## Contact us











Enter your message for us here. We will get back to you within 2 business days.

Submit

# Formularios

## <FORM>

- Define un formulario, que se utiliza para enviar datos al servidor
- Elementos del form
  - input
  - checkbox
  - radiobutton
  - button
  - textarea
  - select
  - y otros...
- Atributos **name** vs **id**:
  - **Name**: *se usa en los elementos de un formulario* para identificar el valor del control cuando se envía al servidor web para su procesamiento. **Solo tiene sentido en los controles de un form**. Varios elementos pueden tener el mismo name: Ejemplo, varios radio-buttons con mismo name que al enviarse al servidor tendrá el valor del radio-button seleccionado.
  - **Id**: *se usa en todos los elementos html* y es para identificar un elemento único en la página para usarlo en scripts, setearle estilos con css.

# Formularios

## **<INPUT type="submit">**

➤ Define un botón, que se usa para enviar (submitir) el formulario al servidor

```
<input type="submit"/>
```



```
<input type="submit" value="Save"/>
```

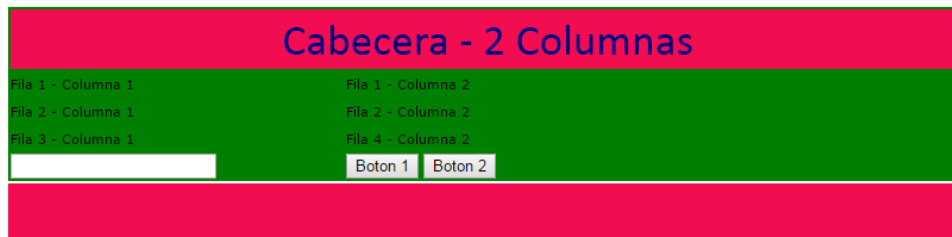
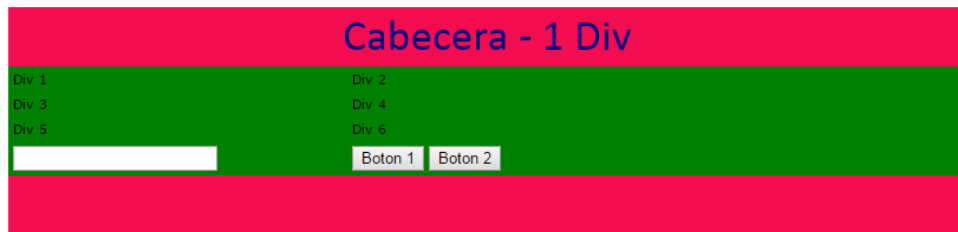


# Maquetado de páginas – Tables vs Divs

- El layout de toda la pantalla se puede realizar utilizando elementos **<TABLE>** o elementos **<DIV>**
- En primeras versiones de HTML los desarrolladores maquetaban con TABLE:
  - Eran más fáciles de entender y ubicar que los DIVS
- **Ahora maquetar con TABLE se considera una mala práctica de desarrollo**, y se recomienda usar DIV
  - Markup Semántico: Los tags o etiquetas no proveen presentación, proveen significado:
    - Semánticamente TABLE se usa para la presentación de información tubular
    - DIV no tiene significado semántico
  - La **tabla** no se renderiza o ve en pantalla hasta que se haya leído completa, por lo que la página **tardará más en ser vista por el usuario**. Los **DIVS** en cambio **van apareciendo** en pantalla a medida que son leídos por el browser y de forma independiente.
- Además de DIV, para maquetar se deben usar los nuevos tags semánticos (HEADER, FOOTER, ASIDE, ARTICLE, NAV – menus)
  - Para lo que no existe explicación semántica y hay que aplicar estilo utilizar DIV o SPAN

# Maquetado de páginas – Tables vs Divs

```
<div class="header">Cabecera - 1 Div</div>
<div class="tbl">
  <div class="cell_35">Div 1</div>
  <div class="cell_65">Div 2</div>
  <div class="cell_35">Div 3</div>
  <div class="cell_65">Div 4</div>
  <div class="cell_35">Div 5</div>
  <div class="cell_65">Div 6</div>
  <div class="cell_35">
    <input id="text1" type="text" />
  </div>
  <div class="cell_65">
    <input type="button" value="Boton 1" id="button1">
    <input type="button" value="Boton 2" id="button2">
  </div>
</div>
<div class="footer">
  <span id="Span1"></span>
</div>
```



```
<table class="tbl" cellspacing="0">
  <tbody>
    <tr>
      <th colspan="2">Cabecera - 2 Columnas</th>
    </tr>
    <tr>
      <td class="space_height21 cell_35">Fila 1 - Columna 1</td>
      <td class="space_height21 cell_65">Fila 1 - Columna 2</td>
    </tr>
    <tr>
      <td class="space_height21 cell_35">Fila 2 - Columna 1</td>
      <td class="space_height21 cell_65">Fila 2 - Columna 2</td>
    </tr>
    <tr>
      <td class="space_height21 cell_35">Fila 3 - Columna 1</td>
      <td class="space_height21 cell_65">Fila 4 - Columna 2</td>
    </tr>
    <tr>
      <td class="space_height21 cell_35">
        <input id="text1" type="text" />
      </td>
      <td class="space_height21 cell_65">
        <input type="button" value="Boton 1" id="button1">
        <input type="button" value="Boton 2" id="button2">
      </td>
    </tr>
    <tr>
      <td colspan="2" class="footer" >
        <strong><span id="Span1"></span></strong>
      </td>
    </tr>
  </tbody>
</table>
```



# Ejercicio



# CSS

- ✓ Conceptos Básicos
- ✓ Reglas de Estilo CSS
- ✓ Estilos en Cascada



# CSS y Conceptos Básicos

## ¿Qué es CSS?

- CSS, siglas de **C**ascading **S**tyle **S**heets: «hojas de estilo en cascada»
- Se usa para definir la presentación visual de las páginas web y aplicaciones Windows 8
- Se escribe en forma reglas de estilo
- Creado y administrado por el W3C (World Wide Web Consortium).

## ➤ ¿Qué es una regla de estilo CSS?

- Una regla define el estilo que tendrá uno o varios elementos HTML
- 2 partes:
  - **Selector:** permite identificar el o los elementos HTML
  - **Bloque de Declaraciones:** conjunto de instrucciones de formateo, separadas entre si por punto y coma ( ; )
    - Instrucción de formateo: propiedad CSS, seguida de dos puntos ( : ), seguida de un valor

## ➤ Ejemplo de regla de estilo

```
body {  
    background-color: white;  
    color: gray;  
}
```

# Reglas de Estilo CSS – Declaración

## Reglas Inline (Inline Style)

- Se declaran dentro de la etiqueta de inicio de un elemento, en el atributo **style**
- No tienen selector, ya que se aplican sólo al elemento donde se declaró, por lo que solo son las definiciones de formato separadas por ;

```
<p style="text-align:center;color:red;">Un párrafo.</p>
```

### ➤ Evitar esta forma de declaración

- Viola el principio de separación de responsabilidades
- No permite reusabilidad
- Perjudica la legibilidad del archivo HTML
- Hace al archivo HTML más pesado por lo que tardará más tiempo en cargar

### ➤ Ventajas

- Sobrescribe los estilos que se hayan declarado en cualquier otro lado, tiene la mayor **especificidad**
- Esta especificidad puede resolver problemas de estilo aislados cuando un estilo global afecta de forma no deseada un determinado elemento

# Reglas de Estilo CSS – Declaración

## Reglas Embebidas (Embedded Style)

- Se declaran en el archivo HTML dentro del elemento html **<STYLE>**
- Deben tener selector, ya que se pueden aplicar a varios elementos de la página donde se encuentran embebidas.
- Se declara dentro del elemento **<HEAD>**
- Aplican a los elementos de la *página* que cumplen con el selector
- **Evitar esta forma de declaración**
  - Separa las responsabilidades de estilo del cuerpo html, pero no permite tener archivos separados
  - Permite reusabilidad solo en los elementos de la página, pero no entre páginas
  - Hace al archivo HTML pesado por lo que tardará más tiempo en cargar
- **Ventajas**
  - Permite reusabilidad entre los elementos de la página

```
<head>
  <style>
    p {
      text-align: center;
      color: red;
    }
  </style>
</head>
<body>
  <p>Un párrafo.</p>
</body>
```

# Reglas de Estilo CSS – Declaración

## Hojas de Estilo Externas (Archivos CSS)

- Se declaran en uno o varios archivos CSS, que puede ser referenciado por algunos o todos los archivos HTML de la aplicación.
- El archivo CSS lo único que tiene son los selectores con los bloques de declaración uno debajo del otro
- Deben tener selector para poder determinar los elementos a los que aplicar
- Con el elemento **<LINK>** dentro del elemento **<HEAD>** se agrega una referencia al archivo CSS.

```
<link href="css/style.css" rel="stylesheet" type="text/css"/>
```

- **La mejor práctica para declarar estilos**

- Se pueden linkear varios archivos CSS con diferentes "porciones" del estilo de la aplicación
- Separa las responsabilidades de presentación de la de estructura
- Permite reusabilidad en los elementos de toda la aplicación, de varias páginas
- Al ser el archivo HTML más liviano, se carga más rápidamente

```
p {  
    text-align: center;  
    color: red;  
}
```

# Reglas de Estilo CSS – Selectores

## Selector

- Palabra o conjunto de palabras que permiten “encontrar” (o seleccionar) elementos HTML basándose en el tipo de elemento, id, clase, atributos y más.

### ➤ Selector por Tipo de Elemento

- Selecciona elementos usando el nombre de la etiqueta
- Directamente se escribe el tipo:

```
<p>Un párrafo.</p>  
<p id="parrafo2">Otro párrafo.</p>
```

```
p {  
  text-align: center;  
  color: red;  
}
```

Afecta a ambos párrafos

### ➤ Selector por Id

- Selecciona un *único* elemento con el atributo **id** del elemento
- Se escribe el símbolo numeral (#) seguido del id:

```
#parrafo2 {  
  text-align: center;  
  color: red;  
}
```

Afecta solo al  
segundo párrafo

# Reglas de Estilo CSS – Selectores

## ➤ Selector por Clase

- Selecciona elementos usando el atributo **class** del elemento
- Se escribe el símbolo punto (.) seguido del nombre de la clase:

```
<p class="centrar">Un párrafo.</p>
<p id="parrafo2">Otro párrafo.</p>
<button class="centrar" id="grabar">
    Grabar</button>
```

- También se puede especificar qué elementos con qué clase son afectados por el estilo
- Se escribe el nombre del elemento, seguido de punto (.) seguido del nombre de la clase:
- Y se puede indicar que sea afectado por más de una clase:

```
<p class="centrar azul">Un párrafo.</p>
<p class="centrar">Otro párrafo.</p>
```

```
.centrar {
    text-align: center;
    color: red;
}
```

Afecta al primer párrafo  
y al botón

```
p.centrar {
    text-align: center;
    color: red;
}
```

Afecta solo al  
primer párrafo

```
.azul {
    color: blue;
}
```

El primer párrafo es  
afectado por clase  
centrar y azul



# Reglas de Estilo CSS – Selectores

## ➤ Selector de Descendencia

- Si se quiere aplicar estilos a un elemento solo si es descendiente de otro elemento.
- Un descendiente es un elemento que está contenido dentro de otro, ya sea inmediatamente o dentro de otros elementos contenidos en él
- Se debe indicar una *cadena de selectores*, separados por espacios, siendo de izquierda a derecha *ancestro de ...* y de derecha a izquierda *descendiente de ...*

```
body header {  
  color: #FFFFFF;  
}
```

Ambos headers tendrán  
letra blanca

```
article header {  
  background-image: url('../img/home-bg.jpg');  
}
```

Solo "titulo2" tendrá una  
imagen de fondo

```
<body>  
  <header id="titulo1">  
    Introducción a HTML, CSS y Javascript  
  </header>  
  <article id="articulo">  
    <header id="titulo2">  
      Este es el título del artículo  
    </header>  
  </article>  
</body>
```

- En documentos HTML grandes, usar estos selectores puede causar problemas de performance por la cantidad de búsquedas de elementos necesaria. En ese caso, implementar un selector más específico, como un selector de hijos.

# Reglas de Estilo CSS – Selectores

## ➤ Selector de Hijos

- Si se quiere aplicar estilos a un elemento solo si es descendiente *directo* (o hijo) de otro elemento.
- Se debe indicar una *cadena de selectores*, separados por el símbolo mayor (>), siendo de izquierda a derecha *padre de ...* y de derecha a izquierda *hijo de ...*

```
body > header {  
  color: #FFFFFF;  
}
```

Sólo "titulo1" tendrá  
letra blanca

```
article header {  
  background-image: url('../img/home-bg.jpg');  
}
```

Solo "titulo2" tendrá una imagen  
de fondo, pero la tercera regla es  
más *eficiente*

```
article > header {  
  background-image: url('../img/home-bg.jpg');  
}
```

```
<body>  
  <header id="titulo1">  
    Introducción a HTML, CSS y Javascript  
  </header>  
  <article id="articulo">  
    <header id="titulo2">  
      Este es el título del artículo  
    </header>  
  </article>  
</body>
```

# Reglas de Estilo CSS – Selectores

## ➤ Otros tipos de Selectores

- Por pseudo-clase (**selector:pseudo-class**) : [http://www.w3schools.com/css/css\\_pseudo\\_classes.asp](http://www.w3schools.com/css/css_pseudo_classes.asp)
- Por pseudo-elementos (**selector::pseudo-element**): [http://www.w3schools.com/css/css\\_pseudo\\_elements.asp](http://www.w3schools.com/css/css_pseudo_elements.asp)
- De hermanos inmediatamente adyacentes (**element1 + element2**):  
[http://www.w3schools.com/cssref/sel\\_element\\_pluss.asp](http://www.w3schools.com/cssref/sel_element_pluss.asp)
- De hermanos seguidos, pero no adyacentes (**element1 ~ element2**):  
[http://www.w3schools.com/cssref/sel\\_gen\\_sibling.asp](http://www.w3schools.com/cssref/sel_gen_sibling.asp)
- Por atributo (**element[attribute]**): [http://www.w3schools.com/cssref/sel\\_attribute.asp](http://www.w3schools.com/cssref/sel_attribute.asp)
- Por valor de atributo (**element[attribute=value]**): [http://www.w3schools.com/cssref/sel\\_attribute\\_value.asp](http://www.w3schools.com/cssref/sel_attribute_value.asp)

# Reglas de Estilo CSS – Selectores

## ➤ Agrupar Selectores

- Los selectores pueden agruparse, si es que comparten definiciones de estilo
- Se agrupan todos en un bloque de definición, separando los selectores con coma (,)

```
h2 {  
    text-align: center;  
    color: red;  
}  
p.centrar {  
    text-align: center;  
    color: red;  
}
```



```
h2, p.centrar {  
    text-align: center;  
    color: red;  
}
```

- Para minimizar código
- Hacer más livianos los archivos

# Reglas de Estilo CSS – Selectores

## ➤ Varios bloques de definición con mismo Selector

- Pueden haber varios bloques de definición con el mismo selector.
- Si las definiciones son para diferentes propiedades en los grupos, se aplican todas las definiciones al elemento seleccionado por el selector.

```
p.centrar {  
    text-align: center;  
}  
p.centrar {  
    color: red;  
}
```

El párrafo resultará  
centrado y de color rojo

- Si las definiciones son para propiedades repetidas, se aplicará la definición que se haya indicado de forma posterior

```
p.centrar {  
    text-align: center;  
    color: red;  
}  
p.centrar {  
    color: blue;  
}
```

El párrafo resultará  
centrado y de color azul  
(el rojo es "pisado" por el  
azul)

# Reglas de Estilo CSS – Propiedades CSS

## Propiedades CSS más comunes

- background-color
  - background-image
  - border-color
  - border-image
  - border-width
  - bottom
  - color
  - display
  - float
  - font-family
  - font-size
  - font-style
  - font-weight
  - height
  - left
  - margin
  - padding
  - position
  - right
  - text-decoration
  - top
  - width
- Lista completa de propiedades CSS: <http://www.w3schools.com/cssref/default.asp>

# Estilos en Cascada

## ¿Cómo resuelve los conflictos de estilo el browser?

- Los browsers usan un orden de precedencia para poder determinar qué estilo aplicara un determinado elemento:
- **Important:** Es un modificador, que se escribe luego de la definición de formato y tiene la mayor prioridad. No se recomienda su uso en exceso, debe usarse solo en el estilo *definido por el usuario*.

```
p {  
    text-align: center;  
    color: red !important;  
}
```

- **Especificidad (Specificity):** El browser determina cuál es la regla más específica (siguiendo determinadas pautas de CSS).
- **Orden Textual:** Para reglas que tienen la misma especificidad, la regla que se encuentra última en el orden textual es la que tiene precedencia.  
El orden textual significa que la regla está escrita en el archivo css después de otra, o que si está en un archivo css diferente, se encuentra en el archivo referenciado último.

# Estilos en Cascada

## Orden de evaluación de los estilos

- La *hoja de estilo* incorporada en el *browser*
- Declaraciones *normales* en la *hoja de estilo de usuario*
- Declaraciones *normales* en la hoja de estilo de la página web (*hoja de estilo de autor*)
- Declaraciones importantes (!important) en la *hoja de estilo de autor*
- Declaraciones importantes en la *hoja de estilo de usuario*

### CSS de Autor

```
p {  
    color: red;  
}
```

### CSS de Autor

```
p {  
    color: green; !important  
}
```

### CSS de Browser

```
p {  
    color: black;  
}
```

### CSS de Usuario

```
p {  
    color: blue;  
}
```

### CSS de Usuario

```
p {  
    color: yellow !important;  
}
```

El párrafo resultará de color amarillo




# Estilos en Cascada

## Especificidad

- Aspecto más confuso para determinar la precedencia de las reglas
- Hay que determinar la regla que es más específica:
  - Se calculan 3 valores para cada selector:
    - **A)** cantidad de selectores por **Id** en el selector
    - **B)** cantidad de selectores por **Clase**, por **Atributo** y por **Pseudo-clase** en el selector
    - **C)** cantidad de selectores por **Tipo de Elemento** en el selector
- Se concatenan los 3 valores, lo que da el *valor de especificidad*.
- A tiene la mayor importancia, B la siguiente y C tiene la menor importancia.

Lowest Precedence			
Selector	a	b	c
*	0	0	0
li	0	0	1
ol + li	0	0	2
div ol + li	0	0	3
div .content	0	1	1
div .content ol + li	0	1	3
div .content ol + li .selected	0	2	3
#main	1	0	0
#main .selected	1	1	0
#main ul + li .selected	1	1	2
Highest Precedence			



# Libros y cursos

## Libros:

- <https://openlibra.com/es/book/download/html5>
- <https://openlibra.com/es/book/download/introduccion-a-css>

## Curso interactivo:

- <https://www.codecademy.com/es/tracks/html-css-traduccion-al-espanol-america-latina-clone#>

¡Recuerden registrarse!

# Ejercicio



# Javascript

- Lenguaje JavaScript: Conceptos Básicos
- ES6



# Lenguaje JavaScript y Conceptos Básicos

## ¿Qué es JavaScript?

- Lenguaje de programación interpretado, dialecto del estándar ECMAScript
- **Se define como orientado a objetos, basado en prototipos, débilmente tipado y dinámico.**
- Se lo utiliza principalmente del lado del cliente (client-side), implementado como parte de un browser permitiendo mejoras en la interfaz de usuario y páginas web.
- Permite interactuar con el **DOM** (***D**ocument **O**bject **M**odel*), que es la representación del documento HTML en una jerarquía de objetos.
  - **Interactuar con el DOM significa que se podrá crear elementos HTML dinámicamente, leer y escribir valores de los elementos, modificar elementos, cambiar estilos aplicando clases o reglas CSS.**
- JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java.

# Lenguaje JavaScript y Conceptos Básicos

## Sentencias

- Comando terminado con punto y coma (;)

```
var totalCost = 3 * 21.15;
```

## Variables

- Una variable es una referencia con nombre de una ubicación en memoria usada para almacenar información.
- En JavaScript se pueden declarar variable usando las palabras reservadas **var**, **let** o **const**
- Nombres de variables deben comenzar con letras, pueden contener números pero no símbolos matemáticos, signos de puntuación salvo \_ y \$, espacios.
- Los nombres son case-sensitive. *miVariable* es otra variable diferente a *MiVariable* o *MIVARIABLE*.

# Lenguaje JavaScript y Conceptos Básicos

## Diferencias entre let – var – const

- Las variables declaradas con const no se pueden reasignar pero no significa que sea inmutable
- Las variables declaradas con let tienen un ámbito a nivel de bloque
- Let previene la sobrescritura de variables

```
var variableVar = 1; {var variableVar = 2; } console.log(variableVar);
```

```
let variableLet = 1; {let variableLet = 2; } console.log(variableLet);
```

```
const variableConst = 1; {const variableConst = 2; } console.log(variableConst);
```

# Lenguaje JavaScript y Conceptos Básicos

## Funciones

- Agrupación de sentencias que son ejecutadas cuando se llama a esa función
- Se usa la palabra reservada **function** seguida de un nombre, luego paréntesis que abren y cierran. Si la función necesita información externa para trabajar, se indican los nombres de los parámetros entre los paréntesis.

```
function Add(x, y) {  
    return x + y;  
}  
var c = Add(5, 10);
```

- El cuerpo de la función debe estar incluido entre llaves { }
- Las funciones pueden almacenarse dentro de variables, y ser ejecutadas llamando a esa variable:

```
var addFunction = function (x, y) {  
    return x + y;  
};  
var c = addFunction(5, 10);
```



# JavaScript - Ámbito de Variables

- Llamado "scope" en inglés, es la zona del programa en la que se define la variable y la variable puede ser accedida y usada.
- JavaScript define dos ámbitos para las variables: *local* y *global*.
- **Variable Local**
  - Accesible solo dentro de la función en la cual la variable se definió

"message" debe usarse  
dentro de la función

```
function createMessage() {  
    var message = "Test message";  
    alert(message);  
}  
  
createMessage();
```

# JavaScript - Ámbito de Variables

## ➤ Variable Global

- Accesible desde cualquier parte del programa
- Se utilizan para compartir variables entre funciones de forma sencilla

Se mostrará "External" y luego "Internal"

```
var message = "External";  
function showMessage() {  
    alert(message);  
}  
function createMessage() {  
    message = "Internal";  
}  
showMessage();  
createMessage();  
alert(message);
```

- Si una variable se declara dentro de una función pero sin poner la palabra **var**, el scope de la variable es global

Se mostrará "Internal" en ambos casos

```
function showMessage() {  
    alert(message);  
}  
function createMessage() {  
    message = "Internal";  
}  
createMessage();  
alert(message);  
showMessage();
```

Siempre declarar las variables que solo se usarán en la función dentro de ésta con **var**.

# JavaScript – funciones útiles

- `document.querySelector()`
- `document.querySelectorAll()`
- `document.getElementById()`
- `document.getElementsByClassName()`

# JavaScript – Objetos

## Object Literal

- La forma más sencilla de crear un objeto en JavaScript es creando un *object literal*.
- Consiste básicamente en una lista separada por comas de conjuntos *clave:valor* y encerrada entre llaves.

```
var person1 = {  
  firstName: "Tom",  
  lastName: "Smith",  
  age: 54,  
  getFullName: function () {  
    return 'Person = ' + this.firstName + ' ' + this.lastName;  
  }  
}
```

# JavaScript – Clases

- A partir de ES6 se pueden declarar clases usando la palabra reservada `class`
- Se puede declarar una clase de forma anónima o nombrada
- El método **constructor** es un método especial para crear e inicializar un objeto creado con una clase
- Un constructor puede usar la palabra reservada `super` para llamar al constructor de una superclase

```
class Rectangulo {  
  var _base;  
  var _altura;  
  constructor(base, altura) {  
    _base = base;  
    _altura = altura;  
  }  
  
  calcArea() {  
    return this.base * this.altura;  
  }  
}
```

```
var r = new Rectangulo(5, 10);  
console.log(r.calcArea()); // 50
```

# JavaScript – Herencia

La herencia se aplica usando la palabra reservada `extends`

```
class Animal {  
  constructor(nombre) {  
    this.nombre = nombre;  
  }  
  hablar() {  
    console.log(this.nombre + ' hace un ruido.');  }  
}
```

```
class Perro extends Animal {  
  hablar() {  
    console.log(this.nombre + ' ladra.');  }  
}
```

```
var p = new Perro('Bingo');  
p.hablar();
```

# JavaScript – Métodos estáticos

Existen métodos estáticos usando la palabra reservada `static`

```
class Sumatorio {  
  static sumarArray(arrayValores) {  
    let suma = 0;  
    for (let valor of arrayValores){  
      suma += valor;  
    }  
    return suma;  
  }  
}
```

```
let suma = Sumatoria.sumarArray([3,4,5]);
```

# JavaScript – Accessors

```
class Persona {  
  get nombre(){  
    return this._nombre;  
  }  
  
  set nombre(nom){  
    this._nombre = nom;  
  }  
  
  let ramon = new Persona();  
  ramon.nombre = "Ramón";  
  console.log(ramon.nombre);
```



# JavaScript – Funciones Arrow

La expresión de función flecha tiene una sintaxis más corta que una expresión de función convencional. Las funciones flecha siempre son anónimas.

Estas funciones son funciones no relacionadas con métodos y no pueden ser usadas como constructores.

```
function sumar(a,b){  
    return a + b;  
}
```

```
var sumar = (a,b) => {return a + b};
```

# JavaScript – parámetros por defecto

Los parámetros por defecto de una función permiten que los parámetros formales de la función sean inicializados con valores por defecto si no se pasan valores o los valores pasados son undefined.

```
function multiplicar(a, b = 1){  
    return a * b;  
}
```

```
multiplicar(5);
```

# JavaScript – Template String

Los template string son literales de texto que habilitan el uso de expresiones incrustadas. Es posible utilizar cadenas de texto de más de una línea, y funcionalidades de interpolación de cadenas de texto con ellas

```
`cadena de texto` 'cadena de texto' "cadena de texto"
```

```
`línea 1 de la cadena de texto  
línea 2 de la cadena de texto`
```

```
`cadena de texto ${variable} texto`
```

# JavaScript – Storage

Son estructuras clave-valor que guardan información del lado del cliente.

La diferencia entre Local Storage y Session Storage es que la primera no tiene fecha de expiración y la segunda solo será válida para la ventana actual en la que estamos navegando y solo son accesibles para el dominio actual. Ambas pueden ser eliminadas si se limpia la información guardada en el navegador.

```
// Set  
localStorage.setItem('algo', 'Guarde algo');
```

```
// Get  
console.log(localStorage['algo']);
```

```
// Delete  
localStorage.removeItem('algo');
```

# Libros y cursos

## **Libros:**

- <https://openlibra.com/es/book/download/introduccion-a-javascript>

## **Curso interactivo:**

- <https://www.codecademy.com/es/tracks/javascript-traduccion-al-espanol-america-latina-clone>

¡Recuerden registrarse!

# Ejercicio



¿Preguntas?



¡Muchas Gracias!