Carpeta Web I - Unidad 1

Unidad 1.1: Introducción a HTML y CSS Parte I

Introducción

Para comenzar, se debe saber el concepto de **Frontend**, que está relacionado con la parte visual que ve el usuario o cliente que consume una aplicación (en este caso una página web), y dicha interactúa con él.

Una *web page* suele tener varios lenguajes a la vez que interactúan entre sí para que el navegador interprete la información que manejan y mostrar los resultados. Los lenguajes a estudiar/trabajar serán:

- HTML (Hyper Text Markup Language): lenguaje de marcado diseñado para dar **estructura** y contenido.
- CSS (Cascading Style Sheets): lenguaje de presentación de tipo hojas de estilo cascada que define el **estilo** del contenido.

Ver Buena Práctica N°1.

HTML

Para agregar contenido se generan aperturas y cierres a través de tags, de la forma <”*Nombre_tag*”> [*contenido*] </”*Nombre_tag*”>
Esto se llama **elemento**.

Por ejemplo para agregar un párrafo:

 Hola Mundo

También se le pueden agregar atributos a una etiqueta, que son para agregar ID, Clase, título o insertar una imagen (img) o un vínculo (href). Por ejemplo para una imagen:

```
<img src="panda.jpg" alt="Panda Feliz">
```

Algunos tags con frecuencia usados son:
<html> , <body>, , <p> , <head> , <h1>,<h2>,...<h6> , etc.

Si dentro de un elemento no hay contenido, se puede escribir así:
<”*Nombre_tag*” / >

Ver Buena Práctica N°2

Estructura y Sintaxis

Todos los archivos HTML deben contar con los elementos:
doctype,html,head,body.

- Doctype: sirve para indicar al navegador qué versión de HTML se está usando. Va al comienzo del documento.
  Se escribe como <!DOCTYPE html> , que identifica a la versión HTML5.
  Ver Buena Práctica N°3.

- Html: delimita el inicio y final de todo el código html que se tendrá. Dentro de este ámbito se encuentran head y body.
  - Head: hay información para el navegador que el usuario no ve.
  - Body: acá se escribe toda la estructura de la página web.

Esqueleto principal del HTML:

```
<!DOCTYPE html>
<html>
   <head>
      ...                    Cabeza
   </head>

   <body>
      ...                  Cuerpo
   </body>
</html>
```

Esqueleto por defecto del HTML5:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Agregar Texto: esto se hace a través del elemento tipo párrafo
Ejemplo:
 Hello World!

Agregar Títulos: se agregan con el elementos h1,h2, hasta h6. A medida que aumenta el h, más chico es el título.
Ejemplo combinado con párrafos:

```
    <h1>Welcome to the David's web page</h1>
    <p> Hello World!! </p>

    <h2>This is a second title, smaller than h1</h2>
    <p> Are you okay?</p>
```

Ver Buena Práctica N°4

Metadata: se coloca en la sección de head para la descripción de los datos, como formato de texto (charset) , links a archivos externos como css y javascript, etc.

Agregar un nombre al documento: se añade en el head con el tag <title></title>. Dicho nombre se ve reflejado en la pestaña de donde se abre el documento, pero no tiene efectos en la vista del archivo.

Algunas convenciones en Buena Práctica N°5.

Agregar imagen: esto se realiza a través del tag <img />. Esto significa que se definen atributos como:

- src: el origen de dónde está la imagen. Para no poner rutas, la imagen debe estar en el mismo directorio que el de la carpeta del proyecto.

- alt: una descripción de la imagen. Es importante ponerlo, porque si la imagen no carga se inserta el nombre alternativo. Además sirve para gente no vidente, se escucha el nombre del alternativo.

Ejemplo de cómo insertar:

```
<img src="imagenDavid.jpg" alt="Imagen de David" />
```

CSS

Estructura y Sintaxis

Definición de un estilo: para definir un estilo a un contenido, se debe crear un **selector** con el mismo tag asociado a dicha estructura. Luego, se define una lista de **propiedades** con sus respectivos **valores**.

Para **enlazar** el css al archivo se agrega en la parte del head la etiqueta link, mencionando al archivo css.

```
<link rel="stylesheet" href="holamundo.css">
```

Existen diferentes tipos de selectores, y son:

- **Tipo**: los más básicos, sin necesidad de agregar atributos.
  En HTML: ... , en CSS: p{...}

  Ejemplo:
  HTML:

```
    <p> Hello World!! </p>
```

CSS:

```
p{
    color: #FF0000;
    font-size: 18px ;
}
```

- **Clase**: se define en CSS una clase de estilo, y luego desde el HTML se debe poner explícitamente como atributo a los elementos que deban adoptar ese estilo.
  En HTML: ... ; en CSS: .clase{...}
  Ejemplo:

  HTML:

```
<p class="pregunta"> Are you okay?</p>
```

  CSS:

```
 .pregunta{
   color:darkgreen;
    font-size: 18px;
}
```

- **Identificador**: similar a clase, pero este estilo lo puede usar un único elemento por página. Se usa muy poco porque los estilos de texto pueden cambiar.
  En HTML: ... ; en CSS: #estilounico{...}

Agregar un fondo: esto se define a nivel <body>, y en el CSS se agrega de la siguiente manera:
background-color = <valor del color>

Ejemplo:

```
   background-color: darkkhaki;
```

Agregar una imagen de fondo: también a nivel de body, con: background-image:url(imagen.jpg) . La imagen debe estar en el mismo directorio de la carpeta del proyecto.

Ejemplo:

```
background-image: url(imagenDavid.jpg);
```

Si el formato de la imagen es chica para el contenido, se repetirá en todo el fondo. Entonces se puede configurar para que no se repita con:
background-repeat:no-repeat
Cabe destacar que esto no agrandará la imagen, así que la pondrá una vez y el fondo restante quedará con lo que tenía anteriormente.

Alinear texto: esto sirve para orientar el texto de los títulos, subtítulos y párrafos. Se anexa en el selector del CSS correspondiente a alinear. Este puede ser centrado(center), izquierda(left) , derecha(right) o justificado(justify).
text-align: <center | left | right | justify>

Ejemplo: alinear el título al centro:

```
h1{
    color: chocolate;
    text-align: center;

}
```

Decorar texto: esto agrega a un texto decoraciones como subrayado hacia arriba(overline), hacia abajo(underline), tachar el texto(line-through), etc.
Se escribe como: text-decoration: (tipo válido de decoración).Ver Buena Práctica N°6.

Ejemplo: tachar el contenido de un párrafo:

```
text-decoration: line-through;
```

Transformar un texto: un texto se puede pasar a minúsculas(lowercase), a mayúsculas(uppercase) o capitalizado(capitalize) que todas las palabras comienzan con mayúsculas. Se hace de esta forma:
text-transform: (tipo de transformación)

Ejemplo: pone el título en mayúsculas:

```
  text-transform: uppercase;
```

Complementos adicionales HTML+CSS

Listas

Sirven para enlistar cosas. Lo importante es que estas listas pueden estar desordenadas(unorder list) u ordenadas(order list).

En primer lugar se define el tag <ul> para identificar que ahí dentro se tendrá una lista desordenada (si se quiere una ordenada). Luego por cada elemento de la lista, se encerrará entre <li>....

Ejemplo (desordenada):

```
<ul>
    <li>Esteban</li>
    <li>Adrián</li>
    <li>Tomás</li>
    <li>Lionel</li>
    <li>Matías</li>
</ul>
```

Salida en el navegador:

Esteban
Adrián
Tomás
Lionel
Matías

Ejemplo (ordenada):

```
<ol>
  <li>Pepsi</li>
  <li>Coca-Cola</li>
  <li>Paso de los Toros</li>
  <li>Crush</li>
</ol>
```

Salida en el navegador:
1.Pepsi
2.Coca-Cola
3.Paso de los Toros
4.Crush

En CSS se le puede agregar **viñetas** a las listas desordenadas, o cambiar en las ordenadas. Esto se hace a través de:
list-style-type: (tipo de viñeta)

Ejemplo: agregar viñeta a una lista ordenada con letras mayúsculas.

```
ol{
    list-style-type: upper-alpha;
}
```

También se le puede agregar una imagen como viñeta:
list-style-image: url('imagen.gif')

Hay ciertos **caracteres especiales** como el < y el > que si se quieren escribir en forma de texto puede tener problemas con los signos asociados con los tags. Se agregan de la siguiente forma:

&lt = '<'
&gt = '>'

Es posible contar con una **familia de fuentes**, que posee por nivel de prioridad fuentes aplicable a textos. Si la primera fuente no la soporta el navegador, sigue con el resto.
Un ejemplo:

```
font-family: "Comic Sans",Helvetica,sans-serif;
```

Observaciones:

- Si la fuente va separada por espacios, se debe poner entre comillas.
- La última fuente debe ser genérica, que pertenezca a una familia de fuente.

Además de esto, se puede agregar un **estilo de fuente** para transformar textos en tipos italic(cursiva) , oblique(cursiva menos soportado), o normal.

font_sytle = <normal|italic|oblique>

Ejemplo:

```
font-style: italic;
```

Buenas Prácticas Unidad 1.1

Buena Práctica N°1:

Se debe entender que como todo lenguaje de programación, los archivos que contienen el código fuente poseen una extensión dependiendo el tipo de lenguaje. En el caso de HTML, la extensión es .html , y de CSS es .css .

Ambos lenguajes son independientes uno del otro, por lo tanto se deben escribir cada uno en un archivo diferente.

Es por esto que en principio se tendrán dos archivos para armar una página: <*nombre_archivo_contenido*>.html y <*nombre_archivo_estilo*>.css

Buena Práctica N°2:

Siempre se deben cerrar los tags.

Buena Práctica N°3:

Poner siempre el tag de doctype para que el navegador identifique correctamente el tipo de HTML a usar.

Buena Práctica N°4:

Usar los h1 sólo para títulos. Además usar en orden los headings.

Buena Práctica N°5:

- Escribir todos los tags en minúsculas.
- Escribir un sólo elemento por línea.
- Es más legible usar identación para ver qué ámbitos están dentro de otros. Por ejemplo dentro de un <html> están el <head> y el <body>. A su vez dentro de estos están el <title> y <p> respectivamente.

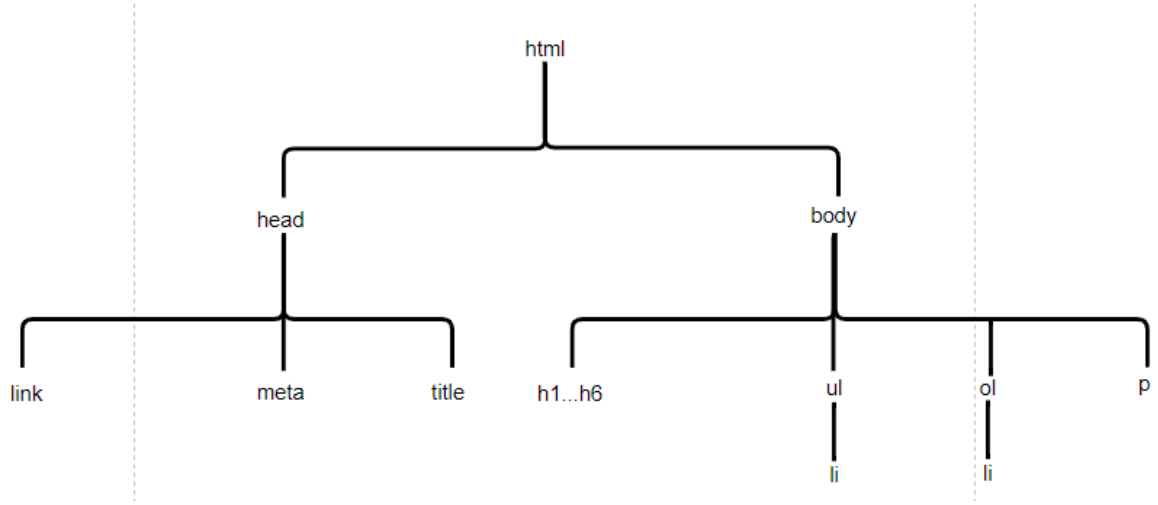
Buena Práctica N°6:

Evitar subrayar texto, ya que se presta a confusión con links que se puedan importar.

Práctico 1- Parte I

1. ¿Cómo es la estructura de un documento HTML? Investigar para qué sirve cada sección del documento.

La estructura de un documento HTML se puede diagramar(hasta lo visto) a través del siguiente árbol:



- html: todo código relacionado con la estructura del documento.
    - head: irá toda la configuración, metadatos y título del documento por fuera del usuario final.
        - link: se encarga de vincular el archivo HTML con el CSS.
        - meta: características de los datos a usar, como la codificación de los caracteres.
        - title: relacionado con el nombre del documento.
    - body: la definición de la estructura. Posee un conjunto de elementos que arman el documento.
        - h1..h6: títulos, en orden de más grande a más chico.
        - ul: delimita a una lista desordenada
            - li: se definen cada fila de la lista.
        - ol: delimita a una lista ordenada.
            - li: se definen cada fila de la lista.
        - p: se definen los párrafos.

2. ¿Para qué sirven los estilos web? ¿Cómo se aplican los estilos al contenido de una página web?

Sirven para manipular los elementos que componen el documento, a través de colores, fondos, tamaños, etc. Para aplicar un estilo se debe crear un archivo CSS y definir los selectores asociados de alguna forma a los elementos definidos en el HTML.

3. ¿Qué tipos de selectores existen en CSS? ¿Cuándo conviene usar cada uno de ellos?

Existen tres tipos de selectores, y estos son:

- De tipo, donde se asocia unívocamente el tag del HTML con el selector. Entonces el estilo adoptado va a repercutir en todos los elementos que estén bajo ese tag. Esto sirve si todos los párrafos definidos se quieren definir de esa manera.
- De clase, que define una clase dentro de un tag seleccionado, y dicha clase se define como un selector aparte con sus estilos. Conviene definir clases cuando se quieren tener dos elementos que son del mismo tipo, pero que tengan diferentes estilos.
- De identificador, que se define como una clase, pero dicho estilo se puede aplicar a un único elemento. En general, se usa si se tiene la certeza que dichos elementos tendrán siempre un único estilo que difiera del resto.

4. Confeccionar la página principal del periódico con las marcas mínimas que debe tener e incluyendo los elementos:
    - <doctype>
    - <html>
    - <head>
    - <body>

En este ejercicio simplemente se debe armar el esqueleto del HTML incluyendo los elementos pedidos, pero introducir contenido.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width,
              initial-scale=1.0">
        <title>Noticias Serranas</title>
     </head>
     <body>
     </body>
</html>
```

5. Colocar **un título de nivel 1** al sitio web que funcione como nombre del periódico.

   Se agrega en la sección del body un tag <h1></h1> que haga de título.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width,
              initial-scale=1.0">
        <title>Noticias Serranas</title>
     </head>
     <body>
         <h1>Noticias Serranas</h1>
     </body>
</html>
```

6. Agregar una hoja de estilos para asignar color y tamaño de la fuente del título.

Para esto ya es necesario agregar un nuevo archivo CSS. Se agrega un color y tamaño definido al título. Recordar que es necesario vincular desde el HTML hacia el archivo CSS.

```
h1{
    color:olive;
    font-size: 45px;
}
```

7. Dado el siguiente código:

```
<bOdy>
   <H1>Bienvenidos a Tudai!</H1>

    <p>Les damos la bienvenida a la Tecnicatura Universitara en Desarrollo de Aplicaciones Informáticas</p>

                  <br>

      <p>Les deseamos un buen inicio y esperamos que la carrera cumpla con vuestras expectativas</p>

          <h3>Catedras del Primer Cuatrimestre

            <h4>Programacion I</h4>
       <p>Se iniciarán y aprenderán los fundamentos y tecnicas básicas de la programación.</p>


     <h4>Web I<h4>
       <p>Aprenderán los conceptos y tecnologías para el desarrollo Web Front-End</p>

     <h4>Taller de Matemática</h5>

       <p>Aprenderán conceptos y ejercitarán la matemática que requiere el contexto de la programación </P>


    </bOdy>
```

Detectar errores y malas prácticas.

Los errores y malas prácticas detectados son:

- No posee la versión de HTML con el que se va a usar.
- No cumple con la estructura de un html. Falta los delimitadores <html> y la parte del <head>.
- La indentación del código es ilegible. Se supone que todos los elementos que poseen una prioridad menor a la del body deben comenzar a la misma altura.
- Los tags <bOdy> , <H1> y </P> no es una buena práctica escribirlos con mayúsculas, aunque HTML sea **case sensitive**.
- Es una mala práctica usar un
 para separar párrafos.
- Después del <h1> se pasa a usar el <h3>. Siempre se deben usar de forma consecutiva.
- No se usa un tag de cierre para el <h3>.
- En el segundo uso de <h4> le falta la / al tag de cierre.
- En el último uso de <h4> se cierra </h5>.

8. Confeccionar **dos títulos de segundo nivel** con los textos *Noticias Políticas* y *Noticias Deportivas*, en cada una de estas secciones definir **dos titulares de tercer nivel** con un párrafo cada una que funcionen como resumen de cada noticia.

Ahora se insertarán dos títulos con el tag <h2> para insertar lo que pide el enunciado. Luego, debajo de cada h2 se definirán dos <h3> que contendrán cada uno un párrafo con algo de texto.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Noticias Serranas</title>
        <link rel="stylesheet" href="noticiasserranas.css">
    </head>
    <body>
        <h1>Noti-Sierras</h1>
```

```
        <h2>Noticias Políticas</h2>

        <h3>Presentaron el nuevo espacio Acción Tandilense</h3>

        <p>Al frente de la flamante iniciativa política está el exdirector
Regional de Trabajo y Empleo
           de la Secretaría de Trabajo de la Nación Gonzalo Santamarina, quien
expresó que Acción Tandilense
           nace como un “hijo bien tandilense de Juntos por el Cambio”. Y
sostuvo que buscan que ese espacio
           “se amplíe, y ofrezca a los bonaerenses una alternativa para ganar
las próximas elecciones”.</p>

        <h3>Lunghi anunció que aceptará todas las medidas dispuestas por el
gobierno provincial</h3>

        <p>El mensaje completo del intendente Lunghi señala: “Queridos
tandilenses, me dirijo nuevamente para
            decirles con total claridad que la pandemia no ha terminado, que
ciertamente ha llegado la segunda
            ola a la República Argentina, con mucha más fuerza, con mucha más
virulencia, con nuevas cepas y
            que las cepas en la República Argentina se encuentran todas. Y que
estas cepas nuevas son muchas
            más virulentas, son mucho más letales y contagian a jóvenes entre
25 y 40 años”.</p>

        <h2>Noticias Deportivas</h2>

        <h3>Volvió el fútbol: los resultados del Preparación</h3>

        <p>Tras estar sin actividad por más de un año, por la pandemia, y tras
un par de postergaciones por
           motivos de protocolo y mal tiempo, la Liga Tandilense de Fútbol
finalmente puso en marcha ayer su
           torneo Preparación para Primera, Quinta y Sexta División.</p>

        <h3>El Maxi Básquet tuvo otra jornada en Unión</h3>

        <p>El gimnasio de Unión y Progreso tuvo el desarrollo de la sexta
fecha del torneo Apertura de Pre Maxi
           y Maxi Básquet.</p>
```

```
    </body>
</html>
```

9. Agregar un **titular de cuarto nivel** con el pie de página del sitio que incluya el nombre del periódico y el año actual.

Con la instrucción

```
<h4>Noti-Sierras - 2021</h4>
```

se agrega un h4 con el nombre del periódico y el año actual.

10. Terminar de implementar las noticias del sitio web que viene realizando. Cada noticia completa debe cumplir las siguientes consignas.
    - Hacer una página diferente para cada una de las noticias.
    - Cada noticia debe tener el nombre del periódico y el pie de página al igual que la página principal.
    - Las noticias deben tener imágenes y al menos una debe tener una lista.
    - En alguna de las noticias que tengan varios párrafos, aplicar un estilo particular y diferente a solo uno de ellos.

- Lo primero que se hizo fue crear 4 nuevos archivos HTML para cada noticia, respetando el título y el pie de página con los estilos dados respecto de la página principal.
- En 3 de las 4 noticias se insertaron imágenes.
- En la noticia deportiva 1 se insertó una lista.
- En la noticia política 2 se aplicó una clase particular para diferenciar un párrafo respecto del otro.

11. Aplicar estilos al sitio para darle una imagen corporativa. Cada una de las páginas debe tener una imagen visual unificada. Asigne dentro del sitio colores a las fuentes, diversos tipos de fuentes, tamaños y estilos de texto. Utilice tamaños relativos de fuentes en las diferentes secciones del sitio.

    Así queda el CSS luego de los estilos aplicados:

```
/*Estilos de los tags*/

h1{
    color:olive;
    font-size: 45px;
```

```
    text-align: center;
    text-decoration: overline;
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
}

h2{
    color: red;
    font-style: italic;
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
}

h3{
    color:darkslategrey
}

h4{
    color: black;
    text-align: center;
    font-size: 13px;
    font-family: 'Times New Roman', Times, serif;
}

p{
    color:black;
    font-size: 16px;
    font-family: 'Times New Roman', Times, serif;

}

img{
    display:block;
    margin:auto;
}

hr{
    height: 10px;
    background-color: black;
}

/*Estilos de las clases*/
```

```css
.comentario{
    color:rgb(41, 36, 36);
    font-size: 18px;
    font-style: italic;

}

.tituloNoticia{
    text-align: center;
}

.separadorTitulo{
    color: brown;
    height: 15px;
    background-color: brown;
}
```

12. Agregar a la pagina del periodico el formato de codificación de caracteres UTF-8 (8-bit Unicode Transformation Format).

13. Aplicar un estilo especial al texto que se encuentra en las listas.

```css
li{
    color: darkgoldenrod;
    list-style-type: square;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
```

14. El periódico decide cambiar su imagen corporativa con una renovación del estilo visual del diario. ¿Que sugiere al periódico que haga con su sitio web?. Si las noticias viejas también deben estar con el nuevo estilo que les recomendaría hacer?

Lo único que debería hacerse en el sitio web sería modificar el archivo CSS, y esos efectos van a modificarse a todas las páginas que estén vinculadas con dicho archivo.

15. Reconocer errores y malas prácticas en el siguiente código:

HTML:

```
<head>
    <style>
      .titulo {
        color: red;
      }
    </style>
  </head>


  <h1>WEB I - TUDAI</h1>
  <h2 style="font-family: Arial">Universidad Nacional del Centro</h2>

  <p id="horarios"><b>Horarios Tandil</b></p>

  <h4>Teóricas</h4>
  <ul>
    <li>Lunes de 15 a 18</li>
  </ul>

  <h4>Teórico-Practicas</h4>
  <ul>
    <li>Jueves de 18 a 21</li>
  </ul>

  <h4>Prácticas</h4>
  <ul>
    <li>COM I: Viernes de 13 a 15</li>
    <li>COM II: Viernes de 15 a 17</li>
    <li>COM III: Viernes de 17 a 19</li>
  </ul>

  <br>

  <p id="horarios"><b>Horarios Bolivar</b></p>

  <h4>Teóricas</h4>
  <ul>
    <li>Martes de 16 a 20</li>
  </ul>
```

```
  <h4>Prácticas</h4>
  <ul>
    <li>Miércoles de 14 a 16</li>
  </ul>

  <h4>Teórico-Practicas</h4>
  <ul>
    <li>Miércoles de 16 a 18</li>
  </ul>

  <h5 class="titulo">Proximamente en otras ciudades de la provincia...</h5>

  <p style="text-align: center;"><IMG style="width: 200px; text-align:
center;"
src="https://www.exa.unicen.edu.ar/sites/default/files/fondoexaface.png"
alt=""/></p>
```

- No está la definición del formato del HTML para el navegador.
- Falta el tag <html>.
- No aparece el tag <body> .
- En el HTML hay definición de estilos, lo cual es una muy mala práctica.
- El título está escrito en mayúsculas. Se recomienda escribirlo normalmente y pasarlo a mayúsculas desde CSS.
- Se saltea del h2 al h4.
- Se usa el <b> , es un tag viejo que da estilo negrita a texto.
- Se usa
 , que no se recomienda.
- En la imagen final, se usa alineación del párrafo al centro para alinear la imagen.
- El tag <IMG> está en mayúsculas.
- En la imagen final, no se pone un alternativo a la imagen.

CSS:

```
body {
    background: #234772;
  }

  h1 {
    color: white;
    text-align: center;
```

```
}

h2 {
  color: white;
  text-align: center;
}

h4 {
  color: white;
}

ul {
  color: white;
}

#horarios {
  font-size: 1.5em;
  color: white;
}
```

- Falta la clase definida en el HTML del título.

16. Transcribir el código del ejercicio anterior para que funcione en su navegador local, con las correcciones realizadas y modificaciones que crea necesarias.

HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Horarios TUDAI</title>
        <link rel="stylesheet" href="style.css">
    </head>

    <body>

        <h1>Web I - TUDAI</h1>
```

```
        <h2>Universidad Nacional del Centro</h2>

        <p id="horarios">Horarios Tandil</p>


        <h3>Teóricas</h3>

        <ul>
            <li>Lunes de 15 a 18</li>
        </ul>

        <h3>Teórico-Practicas</h3>
        <ul>
            <li>Jueves de 18 a 21</li>
        </ul>

        <h3>Prácticas</h3>
        <ul>
            <li>COM I: Viernes de 13 a 15</li>
            <li>COM II: Viernes de 15 a 17</li>
            <li>COM III: Viernes de 17 a 19</li>
        </ul>

        <p id="horarios"> Horarios Bolivar </p>

        <h3>Teóricas</h3>

        <ul>
            <li>Martes de 16 a 20</li>
        </ul>

        <h3>Prácticas</h3>

        <ul>
            <li>Miércoles de 14 a 16</li>
        </ul>

        <h3>Teórico-Practicas</h3>

        <ul>
            <li>Miércoles de 16 a 18</li>
        </ul>
```

```
        <h4 class="titulo">Proximamente en otras ciudades de la
provincia...</h4>

        <img
src="https://www.exa.unicen.edu.ar/sites/default/files/fondoexaface.png"
alt="Logo de Exactas"/>
    </body>
</html>
```

CSS:

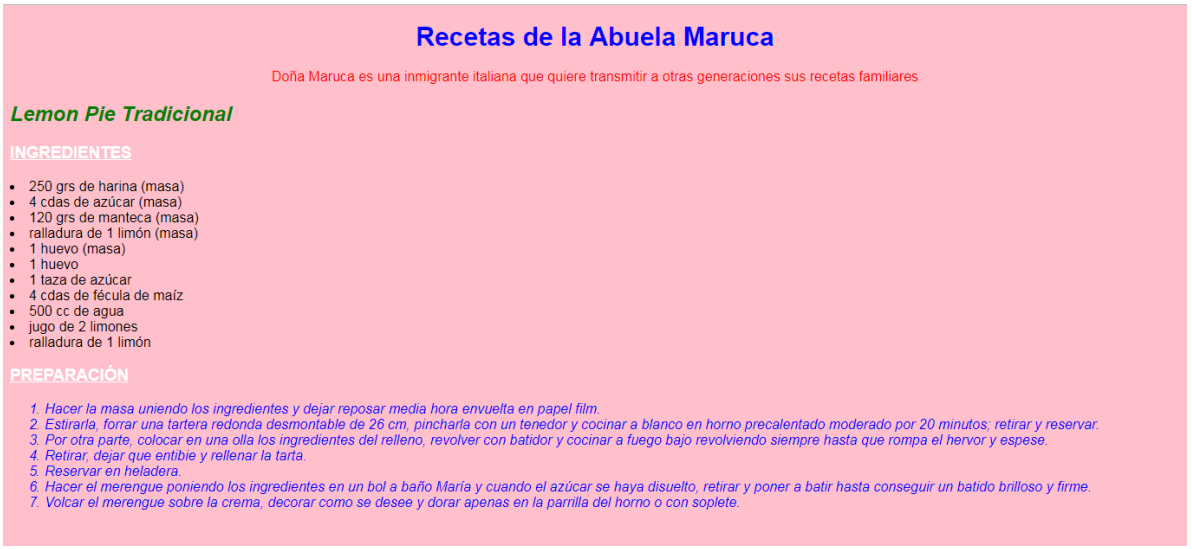
```
body {
    background: #234772;
  }

h1 {
    color: white;
    text-align: center;
}

h2{
    color: white;
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
}

h3 {
    color: white;
  }

img{
    display:block;
    margin:auto;
    width: 200px;
}


.titulo {
    color: red;
  }
```

```
#horarios {
    font-size: 1.5em;
    color: white;
    font-weight: bold;
}
```

17. Dado el siguiente CSS:

```
    body {
        font-family: Arial;
        }

        h1 {
          text-align: center;
            font-size: 30px;
        }

        h2 {
         color: green;
        }

        h3 {
            text-transform: uppercase;
          color: white;
        }

        p {
        color: red;
        }
```

y la siguiente imagen:



Recetas de la Abuela Maruca

Doña Maruca es una inmigrante italiana que quiere transmitir a otras generaciones sus recetas familiares

Lemon Pie Tradicional

INGREDIENTES

- 250 grs de harina (masa)
- 4 cdas de azúcar (masa)
- 120 grs de manteca (masa)
- ralladura de 1 limón (masa)
- 1 huevo (masa)
- 1 huevo
- 1 taza de azúcar
- 4 cdas de fécula de maíz
- 500 cc de agua
- jugo de 2 limones
- ralladura de 1 limón

PREPARACIÓN

1. *Hacer la masa uniendo los ingredientes y dejar reposar media hora envuelta en papel film.*
2. *Estirarla, forrar una tartera redonda desmontable de 26 cm, pincharla con un tenedor y cocinar a blanco en horno precalentado moderado por 20 minutos; retirar y reservar.*
3. *Por otra parte, colocar en una olla los ingredientes del relleno, revolver con batidor y cocinar a fuego bajo revolviendo siempre hasta que rompa el hervor y espese.*
4. *Retirar, dejar que entibie y rellenar la tarta.*
5. *Reservar en heladera.*
6. *Hacer el merengue poniendo los ingredientes en un bol a baño María y cuando el azúcar se haya disuelto, retirar y poner a batir hasta conseguir un batido brilloso y firme.*
7. *Volcar el merengue sobre la crema, decorar como se desee y dorar apenas en la parrilla del horno o con soplete.*

Escribir el HTML correspondiente, y completar el CSS para que el navegador posea el mismo diseño que la imagen.

HTML:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="style.css">
        <title>Recetas de la abuela Maruca</title>
    </head>
    <body>
        <h1>Recetas de la Abuela Maruca</h1>

        <p>Doña Maruca es una inmigrante italiana que quiere transmitir a
otras generaciones sus recetas familiares</p>

        <h2>Lemon Pie Tradicional</h2>

        <h3>Ingredientes</h3>

        <ul>
            <li>250 grs de harina (masa)</li>
            <li>4 cdas de azúcar (masa)</li>
            <li>120 grs de manteca (masa)</li>
```

```
            <li>ralladura de 1 limón (masa)</li>
            <li>1 huevo (masa)</li>
            <li>1 huevo</li>
            <li>1 taza de azúcar</li>
            <li>4 cdas de fécula de maíz</li>
            <li>500 cc de agua</li>
            <li>jugo de 2 limones</li>
            <li>ralladura de 1 limón</li>
        </ul>

        <h3>Preparación</h3>

        <ol>
            <li>Hacer la masa uniendo los ingredientes y dejar reposar media hora envuelta en papel film.</li>
            <li>Estirarla, forrar una tartera redonda desmontable de 26 cm, pincharla con un tenedor y cocinar a blanco en horno precalentado moderado por 20 minutos; retirar y reservar.</li>
            <li>Por otra parte, colocar en una olla los ingredientes del relleno, revolver con batidor y cocinar a fuego bajo revolviendo siempre hasta que rompa el hervor y espese.</li>
            <li>Retirar, dejar que entibie y rellenar la tarta.</li>
            <li>Reservar en heladera.</li>
            <li>Hacer el merengue poniendo los ingredientes en un bol a baño María y cuando el azúcar se haya disuelto, retirar y poner a batir hasta conseguir un batido brilloso y firme.</li>
            <li>Volcar el merengue sobre la crema, decorar como se desee y dorar apenas en la parrilla del horno o con soplete.</li>
        </ol>
    </body>
</html>
```

CSS:

```
body {
    background-color: pink;
    font-family: Arial;
    }
```

```css
h1 {
  color:blue;
  text-align: center;
    font-size: 30px;
}

h2 {
 font-style: italic;
 color: green;
}

h3 {
    text-transform: uppercase;
  color: white;
}

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

ol{
    color: blue;
    font-style: italic;
}
```

Unidad 1.2: Introducción a HTML y CSS Parte II

Links

Los links (vínculos) sirven para transportarse desde una página hasta otra parte de la misma, o a otra página.

Para generar un vínculo se debe agregar con el tag <a>. Poseé dos atributos importantes que son el **href** y el **target**. El primero se pone el hipervínculo de referencia , y en el segundo se pone si se quiere abrir en la misma pestaña, o en una nueva. La instrucción sería, por ejemplo:

```
<a href="https://www.google.com"
target="_blank">Ir a Google</a>
```

Esta instrucción genera un texto "Ir a Google", que al apretarlo, lleva a la página de google, abriendo una nueva pestaña.

Cabe destacar que si se quiere abrir en la misma pestaña se agrega en el target **_self** , o directamente no se agrega el atributo target ya que es por defecto.

Observación:
Cuando se agrega solamente el <a>, se genera un texto con un link, pero esto puede ponerse dentro de otros tags como h1,p,li, etc. para que estos contenedores al apretarlos direccionen a lo que estipule el <a>.

Si se quiere linkear en la misma página, consta de dos partes:

- Definir un <a href= #c> en el lugar que se quiera hacer click para direccionar.
- Poner <a id="c"> a donde se redireccionará la página web. El valor c puede ser cualquier nombre. En general debe ser representativo.

Práctico 1-Parte II - Links

1. Relacionar las dos noticias elaboradas en el **Práctico 1 - Parte I** por medio de hipervínculos sobre el título de las noticias.

Como aclaración se realizaron 4 noticias, lo que se debe hacer es ir a los h3 del archivo index, y hacer la siguiente modificación:

Antes:

```
      <h3>Lunghi anunció que aceptará todas las medidas dispuestas por el gobierno provincial</h3>
```

Después:

```
      <h3> <a href="noticiapolitica2.html" target="_blank">Lunghi anunció que aceptará todas las medidas dispuestas por el gobierno provincial</a> </h3>
```

2. Relacionar una imagen o texto de una de las noticias a cualquier página de la Web utilizando un hipervínculo. El vínculo se debe abrir en una pestaña nueva.

Lo que se hizo fue generar un hipervínculo dentro de la imagen de la 2da noticia política. Al estar la imagen de Lunghi, al hacer click, direcciona al video que muestra la conferencia de prensa. Para esto, se agregaron las siguientes instrucciones:

```
<a href="https://www.facebook.com/Miguel.Lunghi/videos/459729468580577/?t=18" target="_blank"><img src="imagenesNotiSierras/img1notipoli2.jpg"alt="Lunghi en conferencia de prensa"> </a>

<p class="descripcionImagen">La conferencia de prensa de Lunghi</p>
```

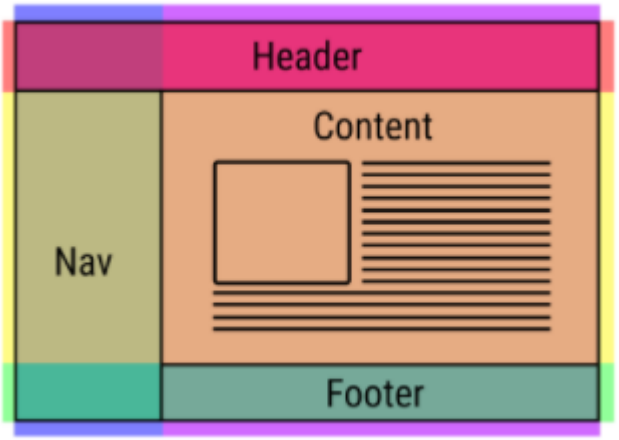
- La primera instrucción pone a la imagen dentro del tag <a>, sin cambios.

- La segunda se agrega un párrafo para aclarar que es una conferencia de prensa. Obsérvese que este párrafo posee una clase especial con un estilo para describir una imagen.

Layouts y Box-Model

Layouts

Definición de **Layout**: define la estructura básica de la *interfaz de usuario* en una aplicación.



Antes de generar el código se recomienda armar un **diagrama de layout** (wireframe) de la página para ver cómo se va a organizar la página web a armar.

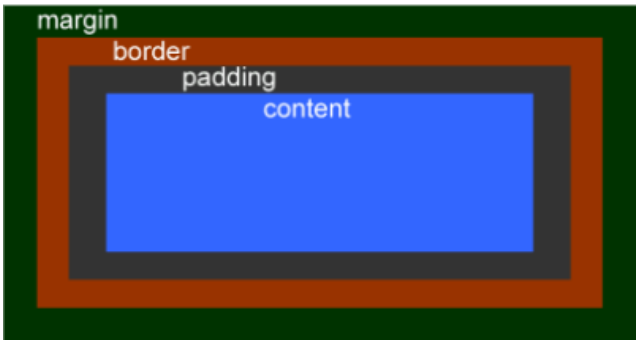
Box-Model

Cada elemento en una página web se representa a través de un box-model, o **caja rectangular**. Cosas a tener en cuenta:

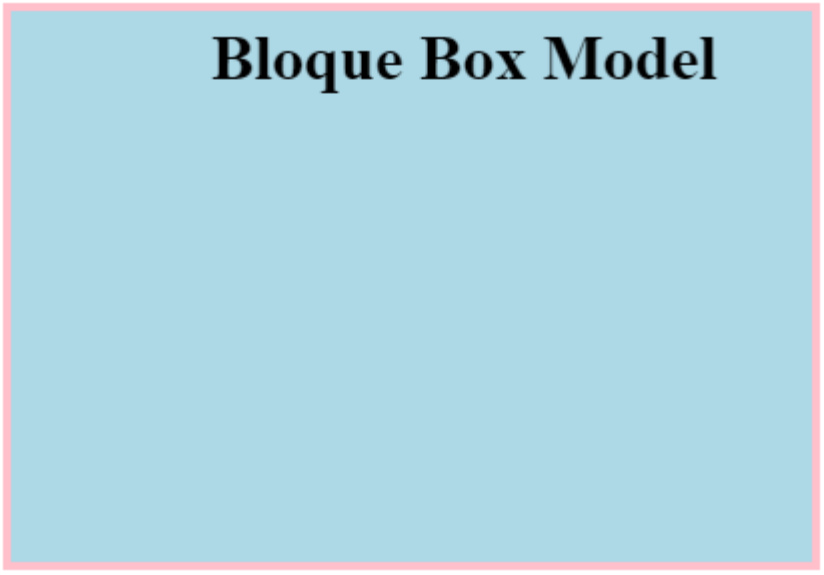
- CSS permite controlar el aspecto y ubicación de las cajas.
- Todos los elementos de HTML se deben representar a través de box-model.

CSS utiliza el modelo de cajas/bloques que consta de 4 partes:

- CONTENT
- PADDING
- BORDER
- MARGIN

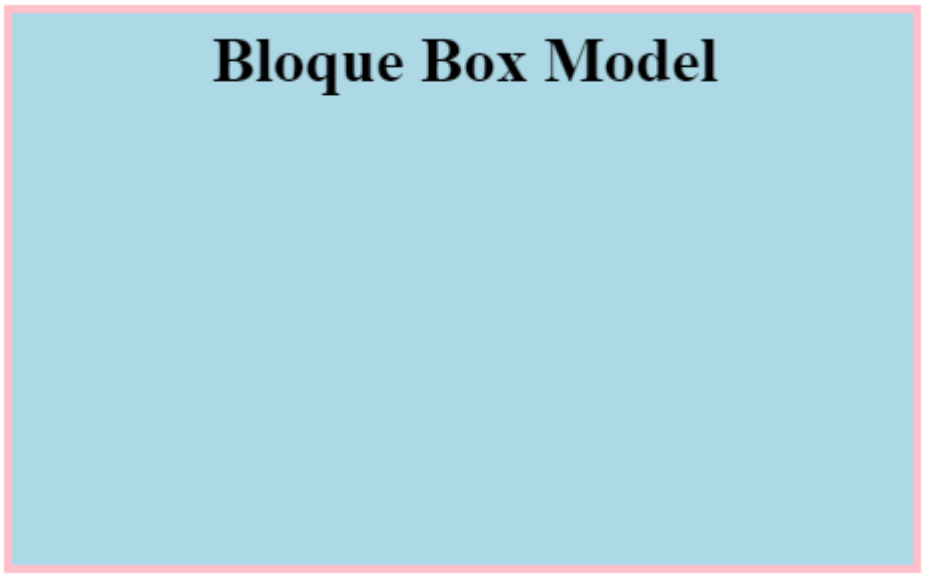


Para explicar los cambios, se usará el siguiente elemento base:

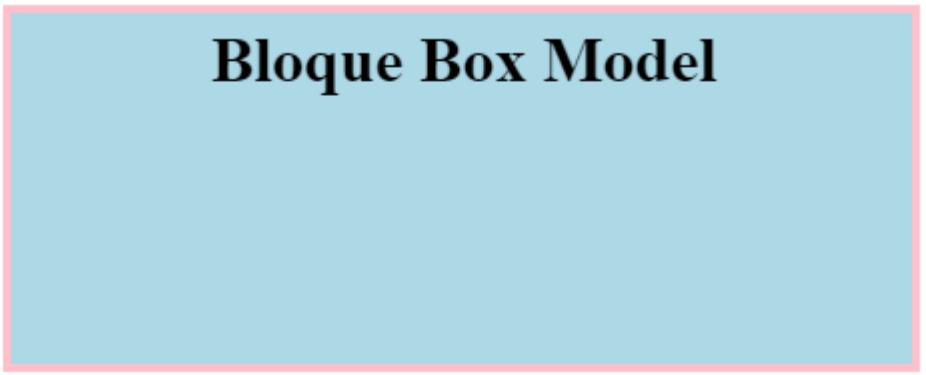


Content: genera el alto(height) y ancho (width) de un elemento.
La imagen original posee widtht:300px y height:250px

Cambiando width a 350px:



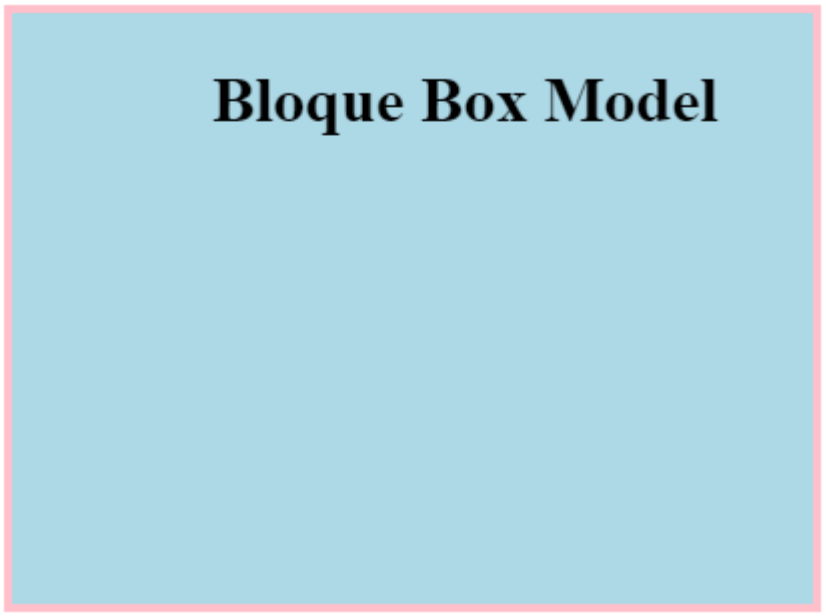
y height a 150px:



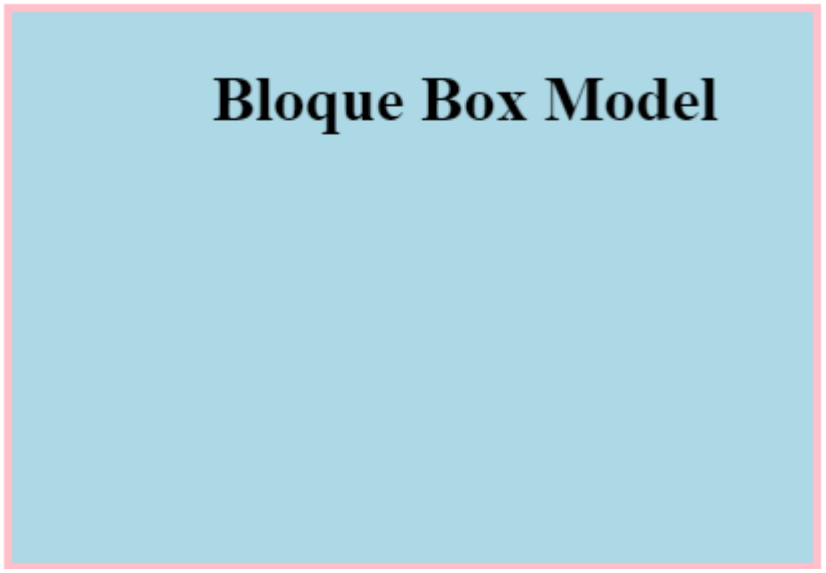
Padding: usado para generar espaciado o margen interior transparente dentro de un elemento. Los márgenes se pueden ajustar a izquierda (**padding-left**),derecha(**padding-right**),arriba(**padding-top**) y abajo(**padding-bottom**). Además se puede ajustar en general con **padding**, que ajusta a todos los lados.

La imagen original posee:
padding-top:5px
padding-bottom:20px
padding-left:100px
padding-right:0

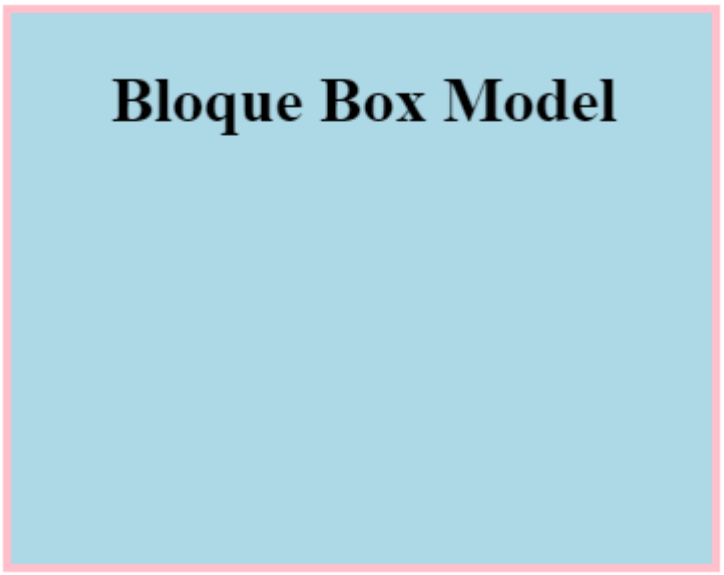
Al cambiar padding-top a 25px:



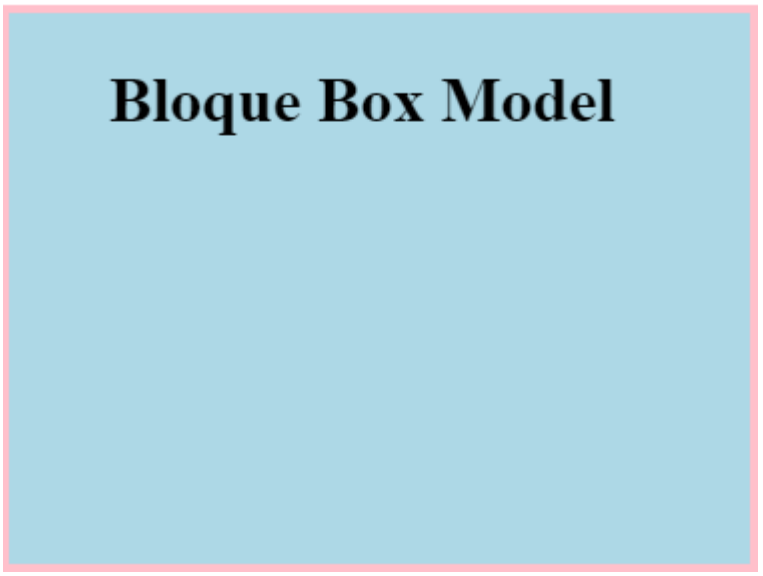
con padding-bottom:0



padding-left:50px



y padding-right:20px



Border: se usa para bordear con una línea a un elemento. Algunos estilos a aplicar son:

border: define el tamaño del borde. También se puede regular como el padding.
border-style: forma del borde.
border-color: color del borde.

Margin: usado para generar margen exterior transparente fuera del elemento. Esto mueve la caja directamente en la misma página. Las operaciones son como las de padding, pero comienzan con margin.

Para calcular el tamaño del box-model es:
Ancho:
width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Alto:
height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

Contenedores

Se usan y , que son contenedores de HTML. Se consideran cajas sin significado semántico. Al ser elementos se pueden poner class o id.

Div

El elemento define un bloque. Se usa generalmente para grandes secciones del sitio y puede incluir varios elementos. Sirve para **construir** el layout y el diseño.

Por defecto el elemento empieza en una **nueva línea** de la página y ocupan todo el ancho disponible. Se puede anidar uno dentro del otro.

Ejemplo de uso del :

```
 <div class="firstBox">
        <h1>Mi primera Caja</h1>
        <p>Es un primer diseño de la caja</p>
    </div>

.firstBox{
    background-color: green;
    font-family:'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
sans-serif;
}
```

Span

Es un elemento **inline**. Es usado para agrupar texto, palabras o frases. Por ejemplo dentro de un párrafo.

El ancho depende del contenido. No se puede anidar con otro elemento, pero si con otro inline.

Un ejemplo para usar el span, sería si se quiere escribir el texto de esta forma:

Texto para probar el **span**

Hasta ahora con Texto para probar el span solo se podía dar estilo a todo ese texto. Pero con el span se puede encerrar el texto para poner en negrita:

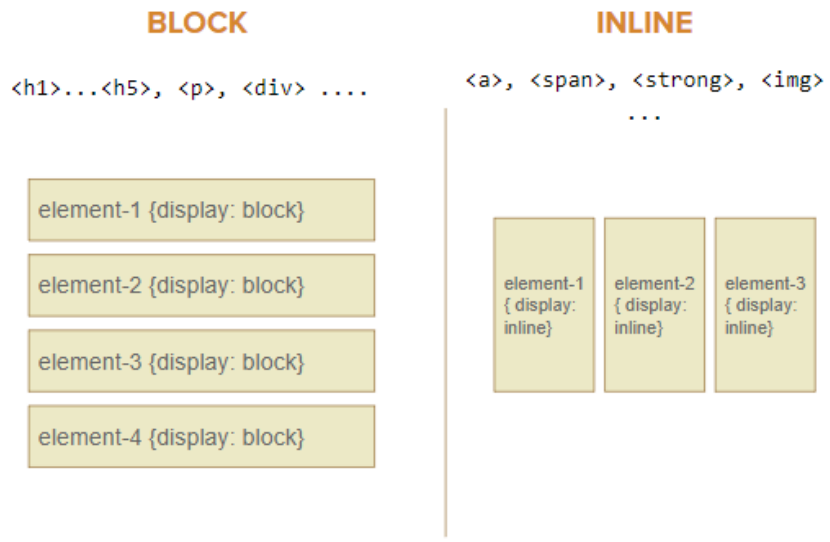
```
<p>Texto para probar el <span class="bold">span</span></p>
```

Flujo de Renderizado(FLEX)

En principio cada elemento, o cajas se vienen agregando uno debajo del otro, pero con lo que se viene explicando, todavía no se pueden agrupar textos de tal manera que estén **inline**. Para esto se hará uso de este concepto para mejorar esto.

En CSS se pueden definir de qué manera los elementos **encajan** con otros, según su propiedad de **display**.

BLOCK: es el que va por defecto, las cajas se colocan una debajo de la otra.
INLINE: las cajas no mueven los elementos alrededor de ellas.



El display que se usará para será el **FLEX**. Esto le da la posibilidad de alterar las dimensiones y orden de los ítems dentro de un contenedor.

Para usar flex, primero se debe crear una caja, y asociarle una clase. Luego en el CSS usar definir el display de esa clase como flex:

```
<div class="container">
    <div class="Izquierda">
       <p>Izquierda </p>
       </div>

       <div class="Derecha">
        <p>Derecha </p>
        </div>
</div>

.container{
    display: flex;
}
```

Ejes del flex: el flex diferencia dos ejes:

- Eje principal: definido por el flex-direction, que puede tomar formas tipo **row**(fila), **column**(columna), entre otros.
- Eje transversal: toma la orientación perpendicular al eje principal.

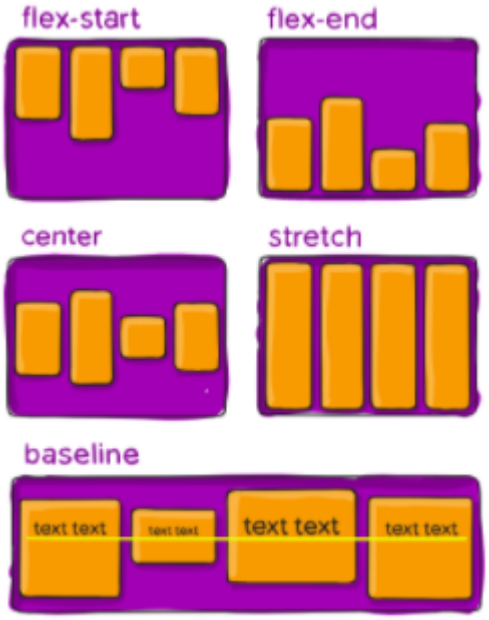
Alineación del flex: se hace con el **justify-content** , que define la alineación de los componentes a lo largo del eje principal. Algunos tipos son:
flex-start , flex-end , center , space-between , space-around, entre otros.

La propiedad de **flex-wrap** permite que, si toma el valor wrap, la caja quede fija, y en caso de no quede lugar en la línea, la caja pasa a estar debajo.

Ejemplo sin flex-wrap:

Ejemplo con flex-wrap:

La **alineación del Flex a nivel perpendicular** se puede hacer a través de **align-items**.



Luego de aplicar estos conceptos de flex, se puede hacer layouts con flex, agregando una caja con header, un contenido dividido en tres secciones (izquierda,centro,derecha) y un pie de página.
Además, usando listas, se puede hacer una botonera como barra de navegación.

Unidades de medida

CSS divide las unidades en dos medidas, las **absolutas** y las **relativas**.

Absolutas: están completamente definidas, ya que son independientes de cualquier referencia. Poseen tamaños fijos en los navegadores/pantallas, y esto genera poca flexibilidad. Algunas medidas son px,pt,mm,cm.

Relativas: no están completamente definidas, porque su valor depende del valor referencia del padre. Es ajustable a cambios de pantalla,generando mayor flexibilidad.

Viendo la imagen se ve que div B y div C están dentro de la misma clase, pero div B posee el 50% de la caja del div A, y div C el 50 por ciento que ofrece el tamaño del navegador.

Posicionamiento

La propiedad **position** sirve para posicionar un elemento dentro de la página. Es muy útil cuando se quieren colocar elementos fuera del flujo normal de la página. Dependiendo del valor que tome position, el elemento tomará dicha referencia u otra para posicionarse. Dichas propiedades son:

- **static**: el valor por defecto. Mantiene el elemento en el flujo normal.
- **relative**: permite usar propiedades como top,bottom,left y right para mover el elemento.
- **absolute**: funciona con las mismas propiedades, pero rompen el flujo normal. Se corresponden con la posición del ancestro(el primero que tiene position que no sea static).
- **fixed**:al igual que el absolute, pero establece directamente una posición fija en la pantalla.
- **scroll**: el elemento se posiciona en base al scroll del usuario.

Práctico 1-Parte II - Layouts y Box-Model

3. ¿Qué es el concepto Box Model y para qué se utiliza? ¿A qué tipo de páginas se aplica?

Box-Model está basado en el armado de cajas a partir de la definición de parámetros como el contenido, relleno, borde y margen. Se supone que todas las páginas web deben seguir el concepto de Box-Model, ya que permite armar los layouts(diseños).

4. Comente las diferencias entre las partes de los contenedores/bloques (content, padding, border, y margin). ¿Cómo se calcula el tamaño total de un contenedor?

El bloque *content* estará el volcado multimedia que tendrá la página. A este se le asigna un alto y ancho. Por su parte el *padding* sería el margen interno del contenedor, determinando la orientación del contenido dentro de la caja usando movimientos hacia arriba,abajo,izquierda y derecha. En el caso del *border* genera un borde a la caja definida que se le puede poner formas,colores y tamaño. Por último el *margin* es el margen externo de la caja. Éste genera una delimitación con otras cajas dentro de la página.

El tamaño total de un contenedor se calcula como:

Ancho:
width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

Alto:
height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

5. En Box Model existen elementos que contienen a otros. ¿Cómo influye en la indentación y escritura clara del código ?

Influye en entender bien cuáles son los niveles de las cajas y qué elementos están dentro de otros. Por ejemplo todos los div que estén a la misma altura serán diferentes cajas que no poseen interrelación. Los elementos internos de otros se los debe indentar para mostrar dicha claridad.

Sin embargo esto no significa que el código no funcione si no se indenta.

El Ejercicio 6 y 7 están repetidos del 1 y 2 respectivamente.

8. Cree una página que incluya diferentes divs. Asigne tamaños de los divs en pixeles (px), porcentaje (%). Experimente que sucede al cambiar el tamaño de la ventana del navegador y obtenga conclusiones.

En principio se crearon unas cajas para probar el funcionamiento del div, y cómo se iban adaptando los fondos, los espaciados entre las cajas y los bordes.

La parte del body queda:

```
<div class="seccionPrincipal">
        <h1>Info-Sistemas</h1>
    </div>

    <div class="seccionPresentacion">
        <p>En esta página se encontrará todo lo necesario para el estudiante
           de Ingeniería de Sistemas de la Facultad de Ciencias Exactas de la
           Universidad Nacional del Centro.
        </p>
```

```
    </div>

    <div class="seccionNoticias">
       <div class="tituloSeccionNoticias">
           <h2>Noticias</h2>
       </div>

        <div class="noticia">
            <div class="tituloNoticia">
                <h3>Es un hecho: el 2do cuatrimestre se cursará de forma
                    virtual
                </h3>
            </div>

            <div class="textoNoticia">
                <p>En el marco por la situación actual de la Pandemia que
                   atraviesa el mundo, el rector Roberto Tassara junto con su
                   equipo de ejecutivos de UNICEN dispusieron continuar la
                   modalidad de clases en forma virtual.Declaraciones del
                   Rector:"Es una decisión difícil de tomar, ya que los
                   estudiantes ansían por regresar a las clases presenciales,
                   pero todavía estamos lejos de superar al Covid-19, y por
                   suerte todas las partes ya se adaptaron a esta nueva forma
                   de aula, por lo que se decidió continuar con la
                   virtualidad".
                </p>
            </div>
        </div>
    </div>
```

Los estilos:

```
h1{
    text-align: center;
    font-style: italic;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
                 sans-serif;
}

h2{
    text-align: center;
```

```css
}

.seccionPrincipal{
    width: 100% ;
    background-color: blue;
}

.seccionPresentacion{
    width: 1200px;
    background-color: crimson;
    border: solid 4px;
    border-color: darkgray;

}

.seccionNoticias{
    width: 50%;
    background-color:darkgreen;
}

.tituloSeccionNoticias{
    width: 100%;
    background-color: darkorange;
    border: solid 3px;
}

.tituloNoticia{
    width: 350px;
    background-color: white;
}

.textoNoticia{
    width: 75%;
    background-color: pink;
}
```

La foto de la página queda de esta manera:

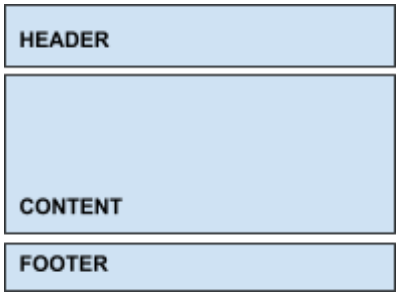
En los análisis pedidos, se ve que si a una caja se le configura que tenga una cantidad fija de píxeles, al cambiar la ventana la caja no variará el tamaño pero se irá cortando a medida que se modifique el tamaño del navegador. Por tanto si se modifica esto con medidas de porcentaje, será más *responsive* , y el tamaño de las cajas se adaptará al tamaño de la ventana.

9. Analice y experimente con los parámetros de los estilos escritos es https://codepen.io/webUnicen/pen/GeLeBZ . Concluya cómo influyen en ejemplo las propiedades margin, padding, border.

Se puede ir variando los valores de margin,padding y border, además de los width. Las conclusiones:

- Margin: al aumentar el margin, no solo se aumenta el tamaño de la caja, sino que genera más separación entre las distintas cajas.
- Border: también aumenta el tamaño de las cajas, pero además se ensancha los bordes de la caja.
- Padding: se corre los elementos a derecha debido a que el margen interno se ensancha.
- El aumento del width hace que el contenido de la caja se alargue y por ende se agranda.

10. Mediante el uso de **Box Layout** diseñe el siguiente layout para una página:



Se usará la misma página, pero se irá modificando el html y el css para que quede con el diseño dado.

Código body modificado:

```
<!-- Header -->
    <div class="header">
        <h1>Info-Sistemas</h1>
    </div>

    <!-- Content -->
    <div class="content">
        <!-- Presentación -->
        <div class="seccionPresentacion">
            <p>En esta página se encontrará todo lo necesario para el
               estudiante de Ingeniería de Sistemas
               de la Facultad de Ciencias Exactas de la Universidad Nacional
               del Centro.
            </p>
        </div>

        <!-- Noticias -->
        <div class="seccionNoticias">
           <div class="tituloSeccionNoticias">
               <h2>Noticias</h2>
           </div>

            <!-- Noticia -->
            <div class="noticia">
                <div class="tituloNoticia">
                    <h3>Es un hecho: el 2do cuatrimestre se cursará de forma
                        virtual
                    </h3>
```

```
                </div>

                <div class="textoNoticia">
                    <p>En el marco por la situación actual de la Pandemia que
                       atraviesa el mundo, el rector Roberto Tassara junto con
                       su equipo de ejecutivos de UNICEN dispusieron continuar
                       la modalidad de clases en forma virtual. Declaraciones
                       del Rector:"Es una decisión dificil de tomar, ya que
                       los estudiantes ansian por regresar a las clases
                       presenciales, pero todavía estamos lejos de superar
                       al Covid-19, y por suerte todas las partes ya se
                       adaptaron a esta nueva forma de aula,por lo que se
                       decidió continuar con la virtualidad".
                    </p>
                </div>
            </div>
        </div>
    </div>

    <!-- Footer -->
    <div class="footer">
        <h4> Info-Sistemas   -    Contacto: 2494-595690   -  email:
        infosis@gmail.com
        </h4>
    </div>
```

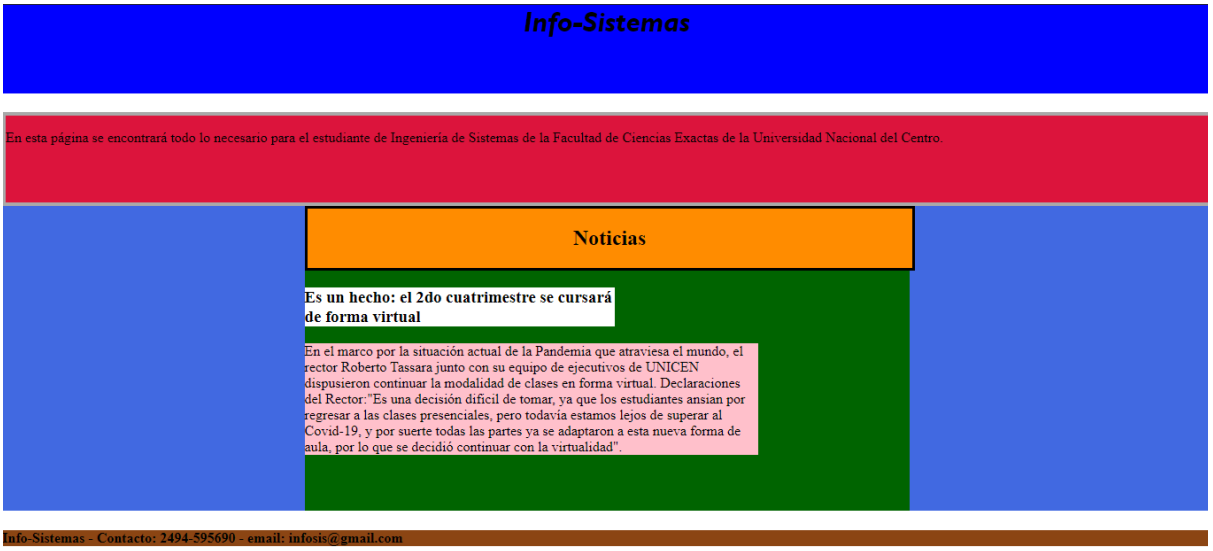
Y el CSS:

```
body{
    margin: 0;
}

h1{
    text-align: center;
    font-style: italic;
    font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
sans-serif;
    margin-top: 0;
    height: 100px;
}
```

```css
h2{
    text-align: center;
}

.header{
    width: 100% ;
    background-color: blue;
    margin: 0;
    padding: 0;
}

.content{
    display: flex;
    justify-content: center;
    flex-flow: row wrap;
    background-color: royalblue;
    width: 100%;
    height: 450px;

}

.seccionPresentacion{
    width: 100%;
    background-color: crimson;
    border: solid 4px;
    border-color: darkgray;

}

.seccionNoticias{
    width: 50%;
    background-color:darkgreen;
}

.tituloSeccionNoticias{
    width: 100%;
    background-color: darkorange;
    border: solid 3px;
}

.tituloNoticia{
    width: 350px;
```

```
    background-color: white;
}

.textoNoticia{
    width: 75%;
    background-color: pink;
}

.footer{
    background-color: saddlebrown;
}
```

Imagen de la página:



11. Mediante el uso de listas, agregue una *barra de navegación horizontal* dentro del header del punto anterior.

A este ejercicio se le agrega al header una barra de tareas. A continuación sólo se muestra la modificación del header:

```
<!-- Header -->
    <div class="header">

        <!-- Titulo -->
        <div class="titulo">
            <h1>Info-Sistemas</h1>
```

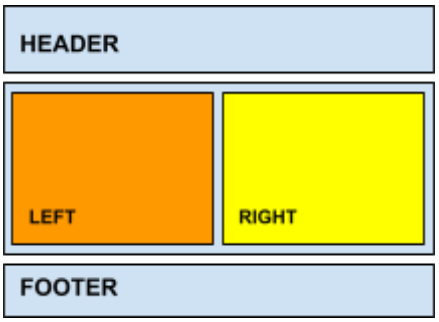
```
        </div>

        <!--Barra de Navegación-->
        <div class="toolBar">
            <ul class="menu">
                <li class="menuItem">Inicio</li>
                <li class="menuItem">Catedras</li>
                <li class="menuItem">Horarios</li>
            </ul>
        </div>

    </div>
```

Las modificaciones que se dieron en el CSS fueron:

```
ul{
    margin: 0;
    list-style-type:none;
}


.header{
    display: flex;
    width: 100% ;
    background-color: blue;
    margin: 0;
    padding: 0;

}

.titulo{
    width: 30%;
    height: 54px;
    border: solid 3px;
}

.toolBar{
    display: flex;
    width: 70%;
    height:60px;
}
```

```css
.list{
    display: flex;
    width: 100%;
    border: solid 3px;
    background-color: yellowgreen;
    justify-content: space-around;

}

.itemList{
    margin: 0 10px;
}
```

12.Modifique el **ejercicio 10** para lograr el siguiente layout:



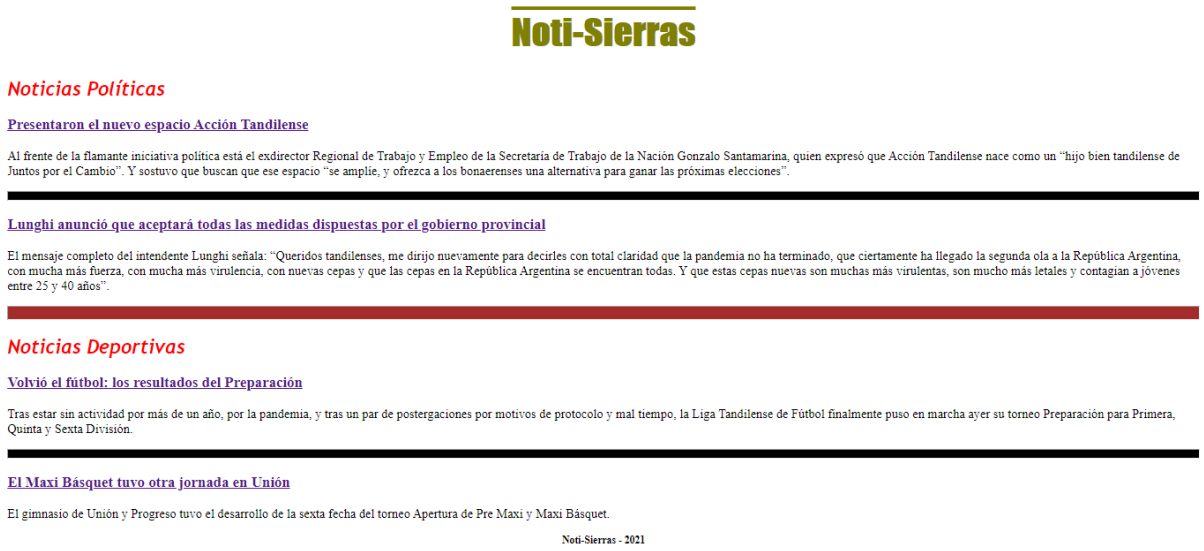
Se dejará la barra de navegación del 11 para no tocar el código.

13. Vuelva a diagramar el sitio de noticias del práctico anterior utilizando Box Model. Deberá tener un encabezado (donde puede ir un logo), una barra de navegación horizontal debajo del encabezado, el contenido donde van las noticias, una barra lateral para publicidades, y un footer con los datos del periódico.

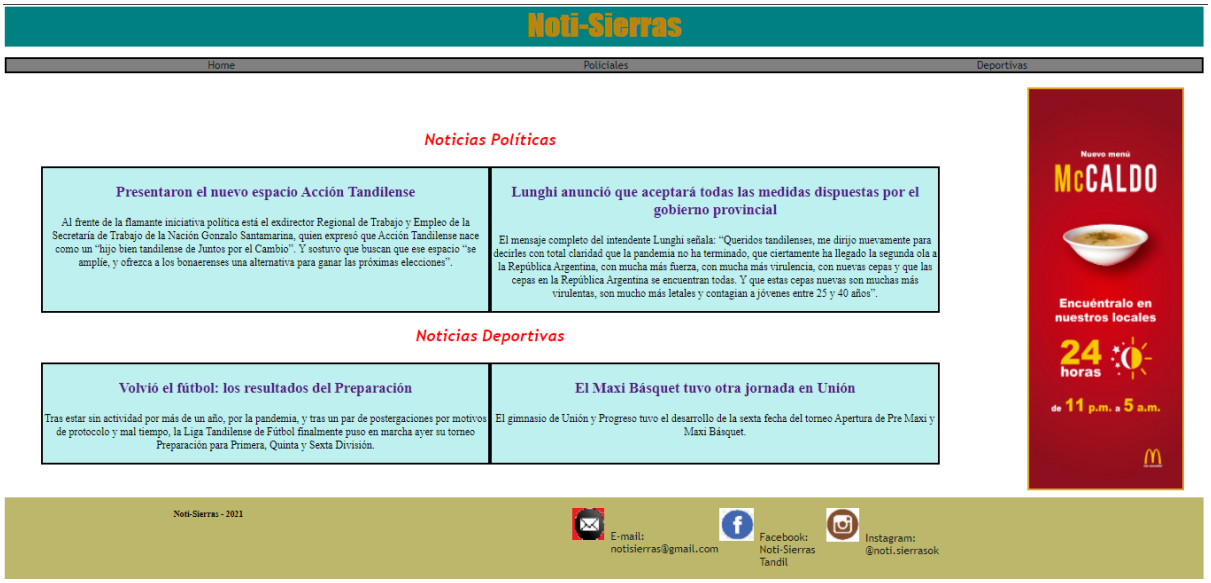
- Las imágenes de las noticias deberán tener un margen de 5 px, y color a elección.
- Las noticias deberán tener un extracto, que usted considere importante de la noticia, resaltado en un rectángulo con un fondo de un color claro, con una fuente diferente y diferente tamaño, centrado, con un borde de color.
- Los links no deberán estar subrayados
- Agregar al periódico una barra de vínculos a redes sociales con sus iconos.

Lo que se hizo fue agarrar el código HTML del home del sitio de noticias, y se modeló en base al concepto de Box-Model, respetando las cosas que pedía el enunciado. Se hará una comparativa para mostrar la diferencia entre la parte I y II.

Sin Box-Model:



Con Box-Model:



Como observación se ve que hay agregados como la barra de navegación, la publicidad y la barra de contactos.

En el siguiente link se puede ver el código HTML de la segunda versión: https://pastebin.com/dDSvsY4n

Y en este el CSS:
https://pastebin.com/DF7dQ8Rx

Luego se hizo una modificación a la primer noticia, también usando box-model y usando los mismos estilos compartidos que el home site.

Como última acotación, en el siguiente texto:

Gonzalo Santamarina expresó su alegría de recibir la visita de los senadores provinciales para el lanzamiento de Acción Tandilense, que **“es un espacio político que reúne a vecinos de Tandil, que lo que queremos es comprometernos con los vecinos de la ciudad. Somos gente de acción y queremos tener este espacio para que otros tandilenses se animen, se sumen y podamos mejorar la calidad de vida de nuestra ciudad”.**

Se observa que parte de él está en negrita. Esto se hizo a través del span. Las aplicaciones fueron las siguientes:

HTML:

```
<p>Gonzalo Santamarina expresó su alegría de recibir la visita de los
   senadores provinciales para el lanzamiento de Acción Tandilense,
   que <span class="negrita">“es un espacio político que reúne a vecinos de
   Tandil, que lo que queremos es comprometernos con los vecinos de la
   ciudad. Somos gente de acción y queremos tener este espacio para que otros
   tandilenses se animen, se sumen y podamos mejorar la calidad de vida de
   nuestra ciudad”.</span>
</p>
```

15. Buscar en Wikipedia un tema de su interés y crear una página de contenido acerca del mismo que contenga un índice con al menos 5 links de referencia a la misma página (anclas, links locales).

a. Definir el elemento title que aparecerá en la barra de título de la página.

b. Definir los metadatos para indicar autor, palabras claves para buscadores, descripción de la página y copyright.

Se hizo una bùsqueda una página tipo Wikipedia que posee la funcionalidad de los enlaces internos, a través del índice que lleva al contenido correspondiente. No se hizo uso de estilos CSS ya que no es el objetivo del enunciado. Respecto de los metadatos sólo se agregaron al autor, y la descripción de la página.

Código de la página: https://pastebin.com/YKwkrjNY

Observaciones:

Se optó por saltear estos ejercicios:

14. En la página principal establecer tamaños de las fuentes con medidas relativas. La primer letra de los párrafos deberá ser el doble de la del párrafo. Los títulos de h1 a h4 deberán ir decreciendo su tamaño en un 20%.

16. Agregar un contenido a un div el cual desborde las dimensiones de altura. Pruebe las distintas alternativas de la propiedad overflow. ¿Qué alternativa usaría para una altura fija? ¿Y para una altura que puede variar?

17.Modificar el siguiente pen para darle un estilo particular al tooltip sin que se solape sobre el texto.
Extras: Considerar que no se solape con una línea inferior que también tenga tooltip.

Sin cursor por encima: Texto con tooltip

Con cursor por encima: Tip Texto con tooltip
https://codepen.io/webUnicen/pen/WpWovg

18. Agregar un menú horizontal de manera que permanezca fijo en la parte superior aun cuando en la página se realice scroll hacia abajo.

19. Agregar un pie de página que siempre quede en la parte inferior del navegador independientemente del largo del contenido. Esto es conocido como **sticky footer**.

Forms y Tablas

Forms

Los forms son formularios que sirven para **pasar información al servidor** (backend). El tag que se usa para esto es el <form>. Dentro de este tag se puede usar el tipo <input>, que éste a su vez tiene diferentes tipos, que se identificarán con el atributo type:

- **text**: se usa para capturar texto.
  Definición: `<input type="text" name="label_name" value="">`

- **password**: se usa para establecer contraseñas
  Definición: `<input type="password" name="label_name" value="">`

- **radio**: es un botón que se usa para seleccionar una única opción de muchas. El sentido de usar radio buttons es definir más de una instrucción, que posean el mismo name para generar exclusividad en la opción.
  Definición: `<input type="radio" name="label_name" value="">`

- **checkbox**: similar en concepto y definición que el radio, pero un checkbox permite seleccionar más de una opción.
  Definición: `<input type="checkbox" name="label_name" value="">`

- **select**: es otra forma de seleccionar opciones, pero se genera un panel para que uno seleccione la opción. Para esto se usa el tag <select>, y la definición de cada una de las opciones.
  Definición: `<option value=""></option>`

- **submit**: es el botón que sirve para enviar los datos del formulario al servidor.
  Definición: `<input type="submit" value="">`

HTML5 agregó otra forma de hacerlo con el tag <button>, que dentro de éste se puede poner texto o una imagen para generar dicho envío.

Luego existen otros tipos de inputs como date, file, textarea, etc.

Algunas definiciones para cerrar:

- El atributo **name** sirve para el control a la hora de enviar el formulario.
- Para los inputs donde el usuario escribe, el **value** pone por defecto tal valor, pero si se modifica por el cliente, cambiará. En general se los pone en vacío, pero en inputs como radio, select o submit se debe poner valor ya que se debe ver la información, que el usuario no tipeará.
- Existe un atributo dentro del input llamado **placeholder** que es un texto dentro de donde se escribirá que recomienda al cliente qué datos poner.
- Todo texto que se quiera escribir dentro de un form para especificar el tipo de campo, se hace a través de la etiqueta **label**. La instrucción sería:

```
<label for="label_name">label_name</label>
```

Tablas

El uso de las tablas es un elemento más que se puede incorporar en una página web. Una de las mayores funcionalidades es que una tabla muestre los datos ingresados en un formulario.

Las tablas se definen con el tag <table>. Esta se divide en filas, definidas con <tr> , y celdas, con <td>. Dentro de una celda se pueden colocar texto,imágenes,etc.

Por motivos de estilos en CSS, y para darle mayor semántica a las tablas, se pueden usar los tags <thead> , <tbody> y <tfooter> para dividir una tabla en parte de arriba, contenido y abajo respectivamente.

Práctico 1-Parte II - Tablas y Formularios

1. ¿ Cuáles son los tags necesarios para generar un tabla ?

   Determine la función de cada uno de los tags:

   - <table>
   - <tr>
   - <td>
   - <thead>
   - <tbody>
   - <tfoot>

Los tags mínimos para crear una tabla sería el <table>, que define al elemento HTML tabla, el <tr>, que crea una fila dentro de la tabla, y el <td>, que forma la celda para ingresar el contenido. Luego los <thead>, <tbody> , <tfoot> son más semánticos, y permiten editarlos más facil en CSS.

Como observación, esto podría solucionarse poniendo al primer tr una clase llamada header, a las intermedios body y a la última footer.

2. Enuncie los tags para construir un formulario y liste los tipos del tag <input> que puede utilizar y cual es la función de cada uno.

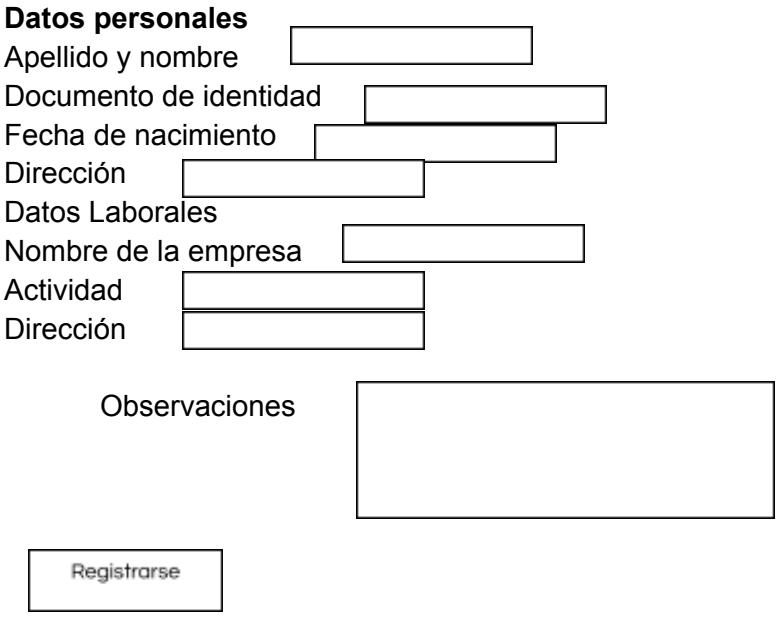
   Los tags necesarios para el armado de un formulario son:

- <form> : es el que genera el elemento HTML formulario.
- <input>: son elementos que permiten interactuar al cliente para realizar una acción dentro del formulario.
- <label>: son las etiquetas de los nombres que tienen los formularios.

   Los tipos de <input> son:

- text: sirven para escribir texto.
- password: escribe texto pero en forma de contraseña.
- radio: sirve para seleccionar una opción dentro de otros elementos radio.
- checkbox: sirve para seleccionar como una de las opciones dentro de otros elementos checkbox.
- select: genera una lista de opciones para elegir.
- submit: genera el botón para enviar el formulario.

3. Crear un formulario de registración a un sitio web que le permita ingresar al usuario los siguientes datos y enviarlos con un botón “Registrarse”.



Para este ejercicio se omitió el uso de Contenedores y estilos en CSS. Se generó el siguiente código para el formulario:

```
<form>

  <label for="apellidoynombre"> Apellido y Nombre: </label> <input type="text" name="apellidoynombre" value="" placeholder="e.g : David D’Ambrosio">


  <label for="dni"> Documento Nacional de Identidad: </label> <input type="text" name="dni" value="" >


  <label for="fechanacimiento"> Fecha de Nacimiento: </label> <input type="date" name="fechanacimiento" value="" placeholder="DD/MM/AAAA">


   <label for="direccion"> Dirección: </label> <input type="text" name="direccion" value="">


   <label for="datoslaborales"> Datos Laborales: </label> <input type="text" name="datoslaborales" value="">


    <label for="nombreempresa"> Nombre Empresa: </label> <input type="text" name="nombreempresa" value="">


     <label for="actividad"> Actividad: </label> <input type="text" name="actividad" value="">


     <label for="direccion_empresa"> Dirección: </label> <input type="text" name="direccion_empresa" value="">


    <label for="observaciones"> Observaciones: </label> <input type="textarea" name="direccion" rows="8" cols="40" value="">
```

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

    </form>
```

4. A la página de noticias incluir una tabla con datos y/o estadísticas, y darle un estilo particular, acorde al diseño.

Se agarró la versión con Layouts de Noti-Sierras y se le agregó por fuera del contenedor de content una tabla de servicio meteorológico de la región, con la fila de cabecera que contuviera los tipos que irán (Ciudad,Min,Max,Condición Meteorológica). Luego otras 4 filas con contenido. Los HTML y CSS son:

HTML:

```
    <table class="weather">
        <thead>
            <td>Ciudad</td>
            <td>Min</td>
            <td>Max</td>
            <td>Condición Meteorológica</td>
        </thead>


        <tr class="fila">
            <td>Tandil</td>
            <td>11º</td>
            <td>24º</td>
            <td>Tormentas</td>
        </tr>

        <tr class="fila">
            <td>Rauch</td>
            <td>12º</td>
            <td>25º</td>
            <td>Nublado</td>
        </tr>

        <tr class="fila">
            <td>Olavarría</td>
```

```
                <td>13º</td>
                <td>25º</td>
                <td>Parcialmente Nublado</td>
            </tr>

            <tr class="fila">
                <td>Balcarce</td>
                <td>9º</td>
                <td>20º</td>
                <td>Lluvioso</td>
            </tr>

        </table>
```

CSS:

```
table{

    table-layout: fixed;
    width: 45%;
    border-collapse: collapse;
    border: red solid 3px;
}

thead,tr{
    border: red solid 3px;
}

thead{
    background-color: seagreen;
}
.fila{
    background-color: steelblue;
}
```

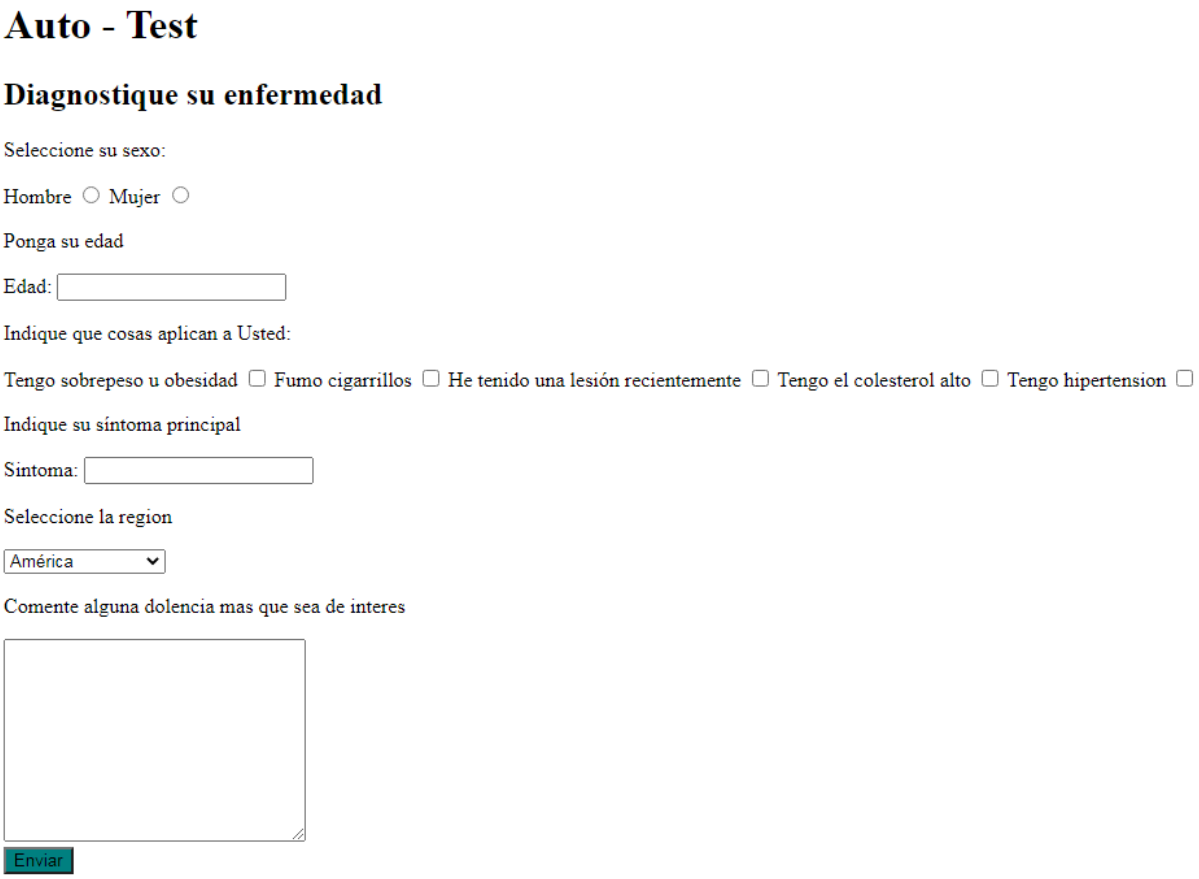
La tabla resultante queda:

Ciudad	Min	Max	Condición Meteorológica
Tandil	11°	24°	Tormentas
Rauch	12°	25°	Nublado
Olavarría	13°	25°	Parcialmente Nublado
Balcarce	9°	20°	Lluvioso

5. Crear una página en HTML que tenga los siguientes elementos de formulario:
   - radio buttons
   - checklists
   - botones comunes
   - text tarea
   - select en sus tres tipos
   - inputs ocultos con valores pre-fijados

   Darle estilo al formulario para:
   - Alinear todos los campos.
   - Cambiar el estilo a los botones.

Así queda el formulario que se pidió:



HTML:

```
<div class="Header">
        <h1>Auto - Test</h1>
    </div>

    <div class="formulario">

        <h2>Diagnostique su enfermedad</h2>

        <form>
            <div class="campo">
                <p>Seleccione su sexo:</p>
                <label for="Hombre">Hombre</label> <input type="radio"
                    name="sexo" value="">
                <label for="Mujer">Mujer</label> <input type="radio"
                    name="sexo" value="">
            </div>
```

```
<div class="campo">
    <p>Ponga su edad</p>
    <label for="edad"> Edad:</label> <input type="text"
     name="edad" value="">
</div>

<div class="campo">
    <p>Indique que cosas aplican a Usted:</p>
    <label for="obesidad"> Tengo sobrepeso u obesidad</label>
        <input type="checkbox" name="caracteristica" value="">
    <label for="fumar">    Fumo cigarrillos </label> <input
         type="checkbox" name="caracteristica" value="">
    <label for="lesion"> He tenido una lesión
         recientemente</label> <input type="checkbox"
         name="caracteristica" value="">
    <label for="colesterol"> Tengo el colesterol alto</label>
         <input type="checkbox" name="caracteristica" value="">
    <label for="hipertension"> Tengo hipertension</label> <input
         type="checkbox" name="caracteristica" value="">
</div>
<div class="campo">
    <p>Indique su síntoma principal</p>
    <label for="sintoma">Sintoma:</label> <input type="text"
    name="sintoma" value="">
</div>


<div class="campo">
    <p>Seleccione la region</p>
    <select>
        <option value="america">América</option>
        <option value="america">Europa</option>
        <option value="america">Resto del Mundo</option>
    </select>
</div>

<div class="campo">
    <input type="hidden" name="capIP" value="192.168.4.0">
</div>

<div class="campo">
```

```
                <p>Comente alguna dolencia mas que sea de interes</p>
                <textarea name="comentario" cols="30" rows="10"></textarea>
            </div>

            <div class="campo">
                <input type="submit" value="Enviar">
            </div>
        </form>
    </div>
```

El CSS:

```
.campo{
    display: block;
}

input[type="submit" i] {

    background-color: teal;

}
```

6. ¿Cuáles son las diferencias entre organizar una página usando división por tablas y división mediante capas (etiqueta DIV)?

La principal diferencia es que a las cajas DIV se les puede asignar el flujo de renderización, y se pueden crear muchos bloques dentro que también puedan cumplir la misma propiedad si se les aplica el mismo display.

8. Construya una tabla con las siguientes características:

- Incluya celdas combinadas
- Estilo particular para el encabezado de la tabla
- Asignar un estilo distinto para las filas pares e impares.

HTML:

```
<table>
        <thead>
            <td>Nombre del Libro</td>
            <td>Autor</td>
            <td>Editorial</td>
            <td>Precio</td>
            <td>Codigo del Inventario</td>
            <td>Stock</td>
        </thead>

        <tr class="rowImpar">
            <td class="tcelda1">El nombre de la Rosa</td>
            <td class="tcelda2">Umberto Eco</td>
            <td class="tcelda3">Saligaris </td>
            <td class="tcelda1">$1700</td>
            <td class="tcelda3">4312-12</td>
            <td class="tcelda2">Si</td>
        </tr>

        <tr class="rowPar">
            <td class="tcelda1">Harry Potter y la Piedra Filosofal</td>
            <td class="tcelda2">J.K Rowling</td>
            <td class="tcelda3">Salamandra</td>
            <td class="tcelda1">$3200</td>
            <td class="tcelda3">6600-1</td>
            <td class="tcelda2">Si</td>
        </tr>

        <tr class="rowImpar">
            <td class="tcelda1">You</td>
            <td class="tcelda2">Carolina Kepnes</td>
            <td class="tcelda3">Atlantis</td>
            <td class="tcelda1">$2500</td>
            <td class="tcelda3">7650-2</td>
            <td class="tcelda2">Si</td>
        </tr>

        <tr class="rowPar">
            <td class="tcelda1">Persecución</td>
            <td class="tcelda2">Sidney Sheldon</td>
```

```
            <td class="tcelda3">New Carolina</td>
            <td class="tcelda1">$800</td>
            <td class="tcelda3">430-1</td>
            <td class="tcelda2">No</td>
        </tr>

    </table>
```

CSS:

```
table{
    table-layout:auto;
    width: 45%;
    border-collapse:collapse;
    border: blue solid 6px;
}

thead,.rowImpar,.rowPar{
    border: green solid 4px;
}

thead{
    background-color: grey;
    font-size: large;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}

.rowImpar{
    background-color: hotpink;
}

.rowPar{
    background-color: lavenderblush;
}

.tcelda1{
    border-right: cadetblue solid 3px;
    font-family: fantasy;
}
```

```css
.tcelda2{
    border-right: chocolate solid 3px;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
    Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
}

.tcelda3{
    border-right: gray solid 3px;
    font-family: monospace;
}
```

Tabla resultante:

Nombre del Libro	Autor	Editorial	Precio	Codigo del Inventario	Stock
El nombre de la Rosa	Umberto Eco	Saligaris	**$1700**	4312-12	Si
Harry Potter y la Piedra Filosofal	J.K Rowling	Salamandra	**$3200**	6600-1	Si
You	Carolina Kepnes	Atlantis	**$2500**	7650-2	Si
Persecución	Sidney Sheldon	New Carolina	**$800**	430-1	No

Observaciones:

Se optó por saltear estos ejercicios:

7. Diseñe un formulario con el propósito que alumnos puedan inscribirse de manera organizada en las 3 comisiones de la práctica de la asignatura. Además de los datos que considere necesarios para el formulario debe permitir el ingreso del DNI de cada alumno con la restricción que el número ingresado sea como máximo de 8 caracteres.
    - Asigne estilos a los inputs, por ejemplo bordes, fondo y fuente
    - Asignar estilo diferente cuando un input de texto es seleccionado para completar

9. Crear un formulario para realizar un inventario de libros de una librería . Debe contener Nombre del libro, Autor, Editorial, Precio, Código de Inventario, Si está en stock o no. Debe asignar estilos y el input debe ser acorde al tipo de dato que debe ser ingresado.

10. Realice una tabla para poder listar los libros ingresados por el formulario. (Por ahora los datos de la tabla se escriben en el archivo html, mas adelante se podrá haciendolo con JS). Asigne estilos a dicha tabla para que se vea distinguido el encabezado, las filas pares e impares diferentes. Crear un estilo distinto no_stock (la lógica para que se aplique se verá luego con JS)

Unidad 1.3: JavaScript

Introducción

Javascript es un lenguaje de programación que se encarga de darle comportamiento a una página web, estando del lado del cliente. Algunas cosas que se usarán para hacer en este lenguaje son:

- Validar Formularios.
- Reaccionar a lo que haga el usuario como hacer click o teclear.
- Cambiar algo al pasar el mouse.
- Partes de páginas que se muestran/ocultan.
- Hacer cálculos.
- Carga dinámica de contenido (AJAX)
- Hacer Single Page Application (SPA).

Para incluir un archivo JS dentro del HTML se debe poner lo siguiente:

```
<script type="text/javascript" src="js/main.js">
```

Ver Buena Práctica Nº1

Algunas funcionalidades básicas

- Alert: esta es una función que permite ingresar texto, y mostrarlo con un cartel al recargar la página web.

  Se define con : **alert(“Texto”);**

  Nota: si se pusieran alerts consecutivos, al cerrar un cartel se abriría el otro.

- Escribir por consola: sirve para ver si lo que se está haciendo es correcto. Además como JS es un lenguaje de programación puede tener errores de sintaxis o de algún funcionamiento. Entonces si se escribe por la consola del navegador se puede ir viendo hasta qué punto el programa corre.

  Se define: **console.log("Texto");**

- Uso de variables y constantes:
  - Variables : **let** <nombre_variable>;
  - Constantes: **const** **<**nombre_constante**> = <**valor**>**

- Lectura del cliente: a través de la función prompt que está anexada al mismo tipo de cartel que el alert se le puede pedir ingresar un valor al usuario. Para que tenga validez se recomienda que el valor se lo tome una variable

  Se define : **prompt**("Ingrese algo");

DOM

El **D**ocument **O**bject **M**odel es una API para documentos HTML. Posee una representación estructurada del documento. Es posible la lectura y escritura del contenido, y esto se puede hacer mediante JavaScript.

Para editar el DOM:

1. Seleccionar qué parte del documento se va a modificar. Para esto se hace a través de los selectores. Esto se almacenará en una variable , con la instrucción:

   variable = document.**querySelector** (<tipo_selector>);

   El documento está al alcance de JS para ser llamado.
   querySelector es una función que trae el elemento que posea ese selector.

2. Al elemento capturado con la variable, se le sobreescribe lo que ya tenía: variable.**innerHTML**(“texto nuevo”);

Observación: al momento de hacer un querySelector, si se tiene un selector de tipo *tipo* o *clase* , estos pudieran estar repetidos. En este caso se tomará el primer elemento que contenga ese selector.

Según el selector, serían, por ejemplo:

- tipo: variable = document.querySelector (“h1”);
- clase: variable = document.querySelector (“.clase”);
- id: variable = variable = document.querySelector (“#id”);

Podría no usarse una variable, y se podría modificar de una, haciendo:

document.querySelector (“#id”).innerHTML(“Cambiar texto”);

Existen otros tipos de funciones de consultas que sirven específicamente para clases o para id, que son
getElementByClass(<selector_class>) y
getElementById(<selector_id>) respectivamente.

En estos casos no es necesario poner el . o el # porque ya estarían contenidos dentro de la misma definición.

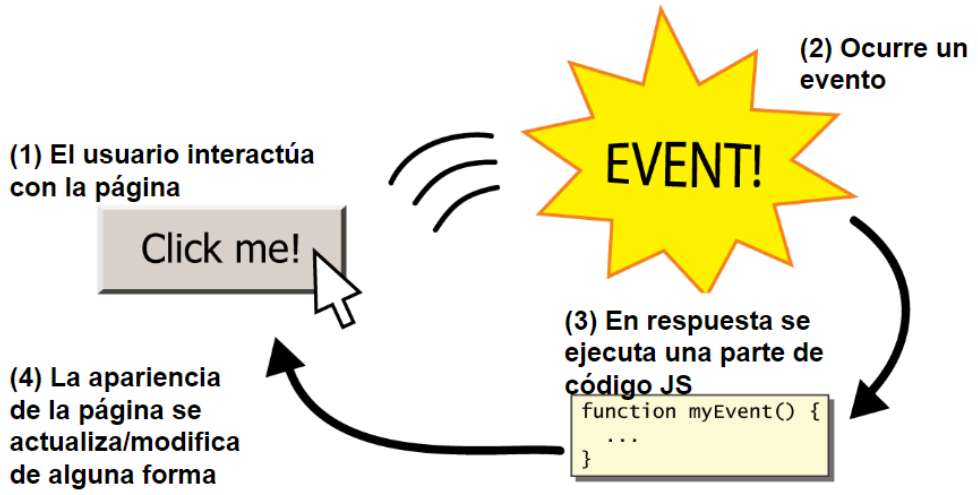
Debe entenderse que innerHTML se encarga de cambiar el contenido del elemento, pero se puede ir modificando otros atributos. Por ejemplo en una imagen se puede cambiar un alternativo o la dirección de origen.

Además del documento, existen otros objetos como **window** (quien tiene el método alert) , **history** (para ver historiales) , **location** (URL de la barra de navegación), etc que se pueden manipular desde JS.

Eventos

Esto ocurre en el sistema, que es originado por el cliente, o por una misma parte del sistema, y se le avisa al sistema para que efectúe una acción. Ejemplos: usuario hace click, al terminar de cargar la página, entre otras cosas.

Diagrama de cómo funciona:



Si se observa el 3er paso, se ve que es necesario el uso de **funciones** en JavaScript.
Para definir una función:
function nombreFuncion(){}

Para ver cómo funciona este mecanismo de eventos, se hará un ejemplo que el usuario escriba su nombre, y se lo salude.

1. En HTML escribir :
   Hola Este será el párrafo en donde se hará el reemplazo debido al evento.

2. Luego definir el elemento input en donde el usuario va a escribir su nombre:
   <input name=”nombre” value=””>

3. Definir el botón con el evento de **onclick** , que al ocurrir el evento se ejecutará el saludar() (esta función se llama handler o callback) :

<button onclick=”saludar();” >Saludar

4. Definir en JS la función saludar:

```
function saludar() {
 let nombre = document.querySelector(input).value; //Obtiene lo escrito en el
                                                    input

document.querySelector(p).innerHTML = “Hola” + “ ” + nombre;  //Se escribe
                                                              en el elemento
                                                              p el saludo.
}
```

En definitiva, lo que hace este evento es que el usuario ingrese su nombre en el elemento del input, y al apretar el botón, se ejecutará la función de saludar, que se encarga de escribir reescribir lo que estaba en párrafo:

OBSERVACIÓN: esta no es la mejor forma de definir eventos , a continuación se definirá otra mejor, para no llamar funciones desde HTML.

Para definir un evento en Javascript, se usará la función addEventListener, que acceden elementos obtenidos desde el HTML. Para el ejemplo anterior, se realiza de esta forma:

HTML:

```
<p>Hola</p>
<input name=”nombre” value=””>
<button id=”btn-saludar”>Saludar
```

JS:

```
let btn = document.querySelector(“#btn-saludar”);
btn.addEventListener( “click” , saludar);

function saludar() {
      let nombre = document.querySelector(input).value;
      document.querySelector(p).innerHTML = “Hola” + “ ” + nombre;
}
```

Ahora se le asigna un id (podría haber sido una clase también) al button, para que desde JS se consulte sobre ese elemento.

En JS el evento que corresponde al hacer click se lo llama "click".

Dentro del addEventListener saludar va sin parámetros. Esto es porque no se está llamando en ese momento, sino que el evento se está definiendo, que "**cuando el usuario haga click en btn, se ejecute la función saludar.**

Ver Buena Práctica Nº2

Estilos desde JavaScript

Este lenguaje no solo permite editar el contenido de la página, sino que también los estilos de la misma. Esto se hará agregando o quitando clases a los elementos seleccionados. Esto se hace con div.classList . Las distintas formas son:

- Agregar una clase: div.classList.add("class");
- Quitar una clase: div.classList.remove("class");
- Agregar una clase si no está, o quitarla si está: div.classList.toggle("class");

Se puede determinar con **contains** sí una clase está dentro de un elemento.

Ver Buena Práctica Nº3

Buenas Prácticas Unidad 1.3

Buena Práctica N°1

Incluirlo al final del body, luego de que ya se cargó el HTML con todos sus elementos.

Buena Práctica N°2

Crear los eventos SIEMPRE desde JavaScript con la función addEventListener.

Buena Práctica N°3

Solamente modificar las clases desde JS, y no los estilos propios, que eso se encarga CSS.

Práctico 1 - Parte III - JavaScript

1. Muestre una alerta al cargar una página
    - Que sea un texto fijo
    - Que sean dos variables *nombre* y *apellido* concatenadas, mostrando en el mensaje: **Nombre:** (valor_nombre) **Apellido:** (valor_apellido)

Este ejercicio es puramente JS porque solamente se pide generar un cartel de alerta al cargar la página. Lo único que se hace desde HTML es adjuntar el script de JS.

Luego , el código para generar un texto fijo con variables nombres y apellido:

```
//Se definen las variables//

let nombre = "David";

let apellido = "D'Ambrosio";

//Se genera el cartel de Alert//

alert(nombre + " " + apellido);
```

2. Escribir un HTML con tres divs vacios (sin contenido) y darle un texto desde Javascript (al cargar la pagina). Reflexione entre las diferencias en hacerlo en HTML o en Javascript.
    a. Es posible desde JS insertar diferentes parrafos adentro del div?

Desde el HTML se crearon 3 divs sin contenido , asignándole a cada uno un tipo particular de id para luego desde JS obtener el elemento a través de esa id, y agregarle el contenido.

HTML:

```
    <div id = primer-div></div>

    <div id = segundo-div></div>

    <div id = tercer-div></div>



    <script type="text/javascript" src="js/main.js"></script>
```

JS:

```
//Se insertan (reemplazar por vacío) estos contenidos a los divs con las clases correspondientes

document.querySelector("#primer-div").innerHTML = "Primer contenido";

document.querySelector("#segundo-div").innerHTML = "Segundo contenido";

document.querySelector("#tercer-div").innerHTML = "Tercer contenido";
```

Es posible incluir párrafos dentro de un DIV. Esto se logra creando el elemento párrafo con createElement , y para agregarlo al DIV se usa el appendChild, en donde se pone a qué contenido padre se debe agregar.

3. Mostrar el mismo mensaje del punto uno, pero haciendo click desde un botón

Para realizar esto, primero se debe crear un botón desde el HTML. Luego en JS se debe generar un evento que al hacer click en el botón, se genere un cartel que diga el nombre y apellido que está almacenado variables.

En la creación del evento se debe asignar una función que se va a ejecutar al hacer click. Como las variables solo sirven para almacenar los nombres y apellidos, van a vivir y morir dentro de la misma función, y se crearán dentro de esta.

HTML:

```
<button id="btn-mostrar-nya">Mostrar Nombre y Apellido</button>

    <script type="text/javascript" src="js/main.js"></script>
```

JS:

```
//Generar evento asincrónico

document.querySelector("#btn-mostrar-nya").addEventListener("click",mostrarNyA
);



//Funcion para Mostrar Nombre y Apellido

function mostrarNyA(){

    let nombre = "David";

    let apellido = "D'Ambrosio";

    alert( nombre + " " + apellido);

}
```

4. Crear 3 botones de distinto color. Agregar la funcionalidad para que se muestre en un párrafo un mensaje que avise cual de los botones fue el último cliqueado.

La idea de este ejercicio es crear 3 botones con distintos id, configurar sus estilos en CSS, y agregarlos a eventos de click en JSS, y definir funciones para que escriban el párrafo definido en el HTML que contendrá el mensaje de quién fue el último botón que se apretó.

HTML:

```
<div class="content-btn">

        <button id="btn-izquierda">Apretame</button>

        <button id="btn-centro">Apretame</button>

        <button id="btn-derecha">Apretame</button>

</div>

<script type="text/javascript" src="js/main.js"></script>

<p></p>
```

CSS:

```
p{

    text-align: center;

    font-size: 45px;

    font-weight: bolder;

       font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
sans-serif;

}

.content-btn{

    display: flex;

    justify-content: space-between;

    align-items: center;

    margin: 4px;

    padding: 150px;

    border: solid darkred 10px;

    background-color: rosybrown;

}

#btn-izquierda{
```

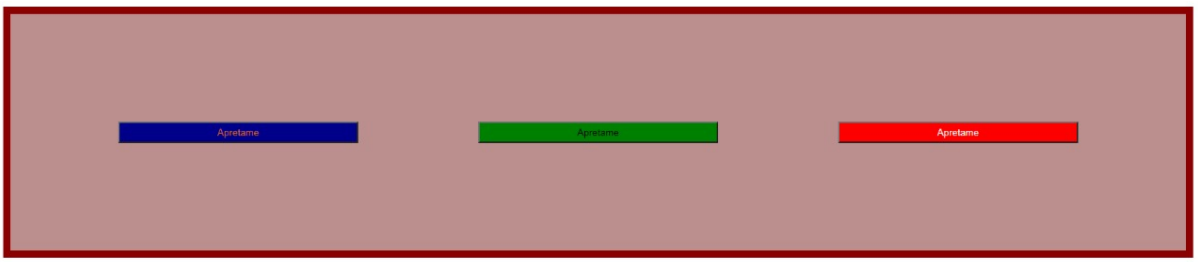
```
    color: chocolate;
    background-color: darkblue;
    width: 25%;
    height: 30px;
}
#btn-centro{
    color: black;
    background-color: green;
    width: 25%;
    height: 30px;
}
#btn-derecha{
    color: white;
    background-color: red;
    width: 25%;
    height: 30px;
}
```

JS:

```
//Generar los eventos de los botones
document.querySelector("#btn-izquierda").addEventListener("click",mostrarMensa
jeIzquierda);
document.querySelector("#btn-centro").addEventListener("click",mostrarMensajeC
entro);
document.querySelector("#btn-derecha").addEventListener("click",mostrarMensaje
Derecha);
```

```
//Funciones

function mostrarMensajeIzquierda() {

     document.querySelector("p").innerHTML = "Se acaba de apretar el botón de
la izquierda"

}

function mostrarMensajeCentro() {

    document.querySelector("p").innerHTML = "Se acaba de apretar el botón del
centro"

}

function mostrarMensajeDerecha() {

     document.querySelector("p").innerHTML = "Se acaba de apretar el botón de
la derecha"

}
```

La página queda:



Se acaba de apretar el botón del centro

5. Realizar una validación tipo Captcha. Para esto la página debe mostrar un texto o número (o algo similar) y pedir introducirlo en el input. Mostrar mensaje en caso de que coincida o no.

Para esto se define un párrafo que tenga el texto (puede ser incluso una imagen) a escribir. Luego se define un input donde el cliente ingresará el texto que se supone que sea el indicado para validar el captcha. Y después el botón para apretar.

Respecto del funcionamiento se genera el evento de clickear el botón, y la función se encarga de capturar el texto de valicación, y el ingresado por el usuario. En caso de que coincidan, se emite un mensaje que fue correcto. Caso contrario un mensaje que no se escribió bien.

HTML:

```
<p class="texto-captcha">A560</p>

<label for="texto-validacion"> Ingrese el contenido de arriba para verificar
que eres humano </label> <input type="text" name=validacion value="">

<button id="btn-validacion-captcha">Validar</button>

<div></div>

<script type="text/javascript" src="js/main.js"></script>
```

JS:

```
document.querySelector("#btn-validacion-captcha").addEventListener("click",val
idarCaptcha);

function validarCaptcha() {

    let texto_captcha = document.querySelector(".texto-captcha").innerHTML;

    let texto_validacion = document.querySelector("input").value;

    if (texto_captcha === texto_validacion)

        document.querySelector("div").innerHTML = "El texto ingresado es

        válido";

    else

        document.querySelector("div").innerHTML = "El texto ingresado es

        inválido, intente de nuevo";

}
```

6. Cree una “Lista de Tareas” donde cada tarea se agrega desde un input de texto.

Este ejercicio se pensó para hacer con layouts, donde en el primero se tiene un input tipo textarea donde el cliente irá anotando la tarea a agregar, y un botón para agregarla. En la segunda sección se irá agregando cada una de las tareas.

Antes de pasar a JS, por cuestiones de comodidad en el HTML es crear la lista ul vacía. Respecto de JS se crea como siempre el evento del click, y lo novedoso definir una variable que tenga apuntada a la lista ul, porque luego se referenciará para ir agregando las nuevas tareas.

Respecto de las nuevas tareas, se irán creando items de la lista para que se agreguen. Estos se hacen con createElement(“li”), que genera un elemento de tipo li. Luego a este elemento se le asigna el contenido escrito desde el textarea.

Una vez terminada esta función, se usa la referencia de la ul, y se le agrega el elemento.

HTML:

```
<div class="content">

        <div class="add-tareas">

            <p>Tarea a agregar</p>

            <textarea name="tarea" id="contenido-tarea" cols="30" rows="10"

            placeholder="Escriba una tarea" ></textarea>

            <button id="btn-add-tarea">Agregar Tarea</button>

        </div>

        <div class="list-tareas">

            <ul></ul>

        </div>

    </div>

    <script type="text/javascript" src="js/main.js"></script>
```

JS:

```
document.querySelector("#btn-add-tarea").addEventListener("click",addTarea);

//Esta variable tiene apuntada a la lista

let listU = document.querySelector("ul");


function addTarea(){

    //Se crea el elemento de la lista

    let elemento_lista = document.createElement("li");

    //Se obtiene el valor de lo escrito en el input

    let contenido_tarea = document.querySelector("#contenido-tarea").value;

    //Se inserta en el elemento el valor obtenido

    elemento_lista.innerHTML = contenido_tarea;

    //Se agrega el elemento creado en el div de la lista de tareas

    listU.appendChild(elemento_lista);




}
```

Captura antes de agregar “Tarea 1”:

Captura después de agregar “Tarea 1”:

7. Crear un formulario con Nombre, Apellido y un botón Enviar. Al presionar Enviar debe mostrar el nombre y apellido como título dentro de una página.

Desde el HTML se crea un formulario con campos de entrada para el nombre y el apellido, además del botón para el evento a crear en el JS. Para la parte del comportamiento, se tiene como objetivo que al apretar el botón de Enviar, se escriba el nombre y apellido como h1 en el home. En vez de crear desde HTML un div o párrafo vacío, se genera desde JS el nuevo elemento de tipo título, tomando la referencia del padre (body) e insertándose ahí. No obstante, se propone insertar el título arriba del formulario, y con appendChild lo agregaría abajo. Para solucionar esto, se hace uso de la función insertBefore(elementoNuevo , elementoActual), que la debe llamar en este caso el body. **elementoNuevo** sería el h1 creado, y **elementoActual** sería el formulario (que también se lo debe traer desde HTML).

HTML:

```
<form>
  <label for="nombre">Nombre</label> <input type="text" name="nombre" value=""
  id="nombre">
 <label for="apellido">Apellido</label> <input type="text" name="apellido"
  value="" id="apellido">
 <button id="btn-enviar"> Enviar </button>
</form>
<script type="text/javascript" src="js/main.js"></script>
```

JS:

```
document.querySelector("#btn-enviar").addEventListener("click", addTitle);

let boxBody = document.querySelector("body");

let boxForm = document.querySelector("form");

function addTitle(event){

    event.preventDefault();

    let contenido = document.querySelector("#nombre").value + " " +

    document.querySelector("#apellido").value;

    let titulo = document.createElement("h1");

    titulo.innerHTML = contenido;

    boxBody.insertBefore(titulo,boxForm);

}
```

Observación: el tag button dentro de un form espera que al ser insertado efectúe alguna acción en otra página, entonces no se reflejan los cambios en la página. Entonces con el preventDefault se pueden deshacer esas operaciones automáticas.

8. Crear un botón que al hacer click cambie el color de fondo de un div y agregue borde de 3px rojo..

Para este ejercicio desde el HTML solo se creo un div con un botón dentro.

A nivel de CSS, se definieron (para lo que importa del ejercicio) dos clases, una actual que posee un par de estilos de la caja original y otra que será la que tendrá después del evento.

Para JS, se crea el evento que al hacer click se llame a la función, donde esta eliminará la clase del div, para ponerle la nueva.

HTML:

```
<div class="box-original">

   <button id="btn-cambiar-estilo"> Cambiar Estilo </button>

</div>

<script type="text/javascript" src="js/main.js"></script>
```

CSS:

```
.box-original{

    background-color: burlywood;

    height: 150px;

}

#btn-cambiar-estilo{

    width: 70%;

    height: 50%;

    margin-top: 2%;

    margin-left: 10%;

}

.box-cambiado{

    background-color: gray;

    border: solid red 3px;

    height: 150px;

}
```

JS:

```
document.querySelector("#btn-cambiar-estilo").addEventListener("click",changeS
tyle);

function changeStyle(){

    let boxOrginal = document.querySelector(".box-original");

    boxOrginal.classList.remove("box-original");

    boxOrginal.classList.add("box-cambiado");

}
```

9. Crear una página que tenga un contenedor (div) con información y un botón. Cree una función Javascript que oculte o muestre el div que contiene la información.

Este ejercicio se basa en crear un contenedor que por defecto tenga una clase con contenido y un estilo bien definido. Además se cuenta por fuera de la caja dos botones, que permitirán aparecer o desaparecer dicho contenido. En CSS se definen las clases para cuando la caja está **presente** y para cuando está **oculta**, donde a esta se le debe poner que no haya display. Luego en JS se definirán los eventos de desaparecer y aparecer la caja, donde el primero se encarga de eliminar la clase presente y agregar la oculta, y el segundo viceversa.

HTML:

```
<div class="presente">
        Lorem ipsum dolor sit amet consectetur, adipisicing elit. Dolor
recusandae inventore impedit ex, dolores
        delectus alias exercitationem tenetur ullam, laboriosam aut
consequatur sint dignissimos nostrum repellat
        culpa voluptatum reprehenderit consectetur?
</div>

<div class="botones">
```

```
    <button id="btn-desaparecer"> Desaparecer </button>
    <button id="btn-aparecer"> Aparecer </button>
</div>

<script type="text/javascript" src="js/main.js" ></script>
```

CSS:

```
.presente{
    background-color: khaki;
    border: solid 5px black;
    text-align: center;
    margin: 10px;
}

.oculto{
    display: none;
}

.botones{
    display: flex;
    background-color: blanchedalmond;
    justify-content: space-around;
    align-items: center;
    margin: 300px;
    width: 60%;
}
```

JS:

```
document.querySelector("#btn-desaparecer").addEventListener("click",hideBox);
document.querySelector("#btn-aparecer").addEventListener("click",showBox);

function hideBox(){
    let box = document.querySelector("div");
    box.classList.remove("presente");
    box.classList.add("oculto");
}
```

```
function showBox(){
    let box = document.querySelector("div");
    box.classList.remove("oculto");
    box.classList.add("presente");
}
```

10. Implementar una calculadora que permita ingresar dos operandos mediante dos inputs y permita realizar las operaciones básicas (suma, resta, división, multiplicación) elegidas desde una lista desplegable (select).

Para este ejercicio se definieron 2 inputs para ingresar los valores, y una lista desplegable para seleccionar el operador para la cuenta. Además el botón de calcular para que genere el evento de obtener el resultado.

La generación del resultado se hace desde el javascript, cuando se clickea el botón de calcular. Esta función obtiene los inputs escritos, los parsea a tipos enteros (porque los toma como String) y se obtiene además el signo seleccionado. Se llama a otra función que de acuerdo a qué tipo de operación, devuelve el resultado entre el primer operando y el segundo. El resultado se escribirá como un nuevo párrafo debajo de la calculadora. Aquí algunas aclaraciones:

- En caso de una división por cero, el elemento contiene un mensaje diciendo que la cuenta no se puede realizar.
- Para que se puedan hacer muchas cuentas sin necesidad de refrescar la página, al calcular el nuevo resultado, se elimina el elemento anterior (si existía) y se inserta el nuevo.

HTML:

```
<div>
   <label for="primer-operando"> Ingrese el primer numero </label>
   <input type="text" name="primer-operando" id="primer-operando" value=""
    placeholder="e.g 27">
</div>

<div>
    <select>
       <option value="suma"> + </option>
```

```
        <option value="resta"> - </option>
        <option value="producto"> * </option>
        <option value="cociente"> / </option>
    </select>
</div>

<div>
    <label for="segundo-operando"> Ingrese el segundo numero </label>
    <input type="text" name="segundo-operando" id="segundo-operando"
     value="" placeholder="e.g 72">
    </div>

<div> <button id="btn-calcular"> Calcular </button>  </div>

<script type="text/javascript" src="js/main.js"></script>
```

JS:

```
document.querySelector("#btn-calcular").addEventListener("click",generarResult
ado);

let boxBody = document.querySelector("body");

function generarResultado(){

    let signo = document.querySelector("select").value;

    let primerOperando =  document.querySelector("#primer-operando").value;

    let segundoOperando = document.querySelector("#segundo-operando").value;

    primerOperando = parseInt(primerOperando);

    segundoOperando = parseInt(segundoOperando);

    let resultado = getResultado(primerOperando, signo ,segundoOperando);

    let nodo = document.createElement("p");

    if(resultado=== Infinity)

        nodo.innerHTML = "No se puede operar";

    else

        nodo.innerHTML = "El resultado es: " + resultado;

    if( boxBody.querySelector("p")){

        let elemBorrar = boxBody.querySelector("p");

        boxBody.removeChild(elemBorrar);

    }

    boxBody.appendChild(nodo);


function getResultado(primerOperando, signo ,segundoOperando){

    switch(signo){

        case "suma":

             return primerOperando + segundoOperando;

        case "resta":
```

```
            return primerOperando - segundoOperando;
        case "producto":
            return primerOperando * segundoOperando;
        case "cociente":
            return primerOperando / segundoOperando;
    }
}
```

12. Extender el ejercicio 7 para que podamos generar una tarjeta personal vía web. Debemos poder agregar Nombre, Apellido, Profesión, Email, Teléfono, Dirección. Una vez enviados los datos, estos deben aparecer en la página con apariencia y alineación de una tarjeta personal.

Se agarró el modelo del ejercicio 7, y se agregaron los campos que se pedía. Para el armado de la tarjeta personal, desde HTML solamente se generaron solamente los contenedores para que luego insertar la información extraída del formulario.

Para CSS, se puede destacar las clases tarjeta y oculto, donde la primera posee todo el estilo, y la 2da oculta la tarjeta hasta que no se aprete el botón para generar el evento.

En JS, el trabajo fue más laborioso, ya que para cada campo del formulario se tuvo que traer la información desde el HTML, crear un elemento para insertar su respectiva información, e insertar estos elementos dentro de la tarjeta personal. Fue necesario traer las secciones de la tarjeta, para incluir los 3 primeros campos en la sección de arriba, y el resto en la de abajo.

HTML:

```
<form>
        <label for="nombre">Nombre</label> <input type="text" name="nombre"
value="" id="nombre">
        <label for="apellido">Apellido</label> <input type="text"
name="apellido" value="" id="apellido">
        <label for="profesion">Profesión</label> <input type="text"
name="profesion" value="" id="profesion">
```

```
        <label for="email">E-Mail</label> <input type="text" name="e-mail"
value="" id="e-mail">
        <label for="telefono">Telefono</label> <input type="text"
name="telefono" value="" id="telefono">
        <label for="direccion">Direccion</label> <input type="text"
name="direccion" value="" id="direccion">
        <button id="btn-enviar"> Generar Tarjeta Personal </button>
    </form>

      <div class="oculto">

         <div class="seccion-arriba">

         </div>

         <hr>

         <div class="seccion-abajo"></div>

      </div>
```

CSS:

```
.tarjeta{
    text-align: center;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    width: 60%;
    margin: 15%;
    border: solid black 3px;
    background-color: cornflowerblue;
    height: 350px;

}

.oculto{
    display: none;
}

.seccion-arriba,.seccion-abajo{
```

```
    margin:10px;

}

.seccion-arriba{
    background-color: gray;
    height: 70%;
    font-size: 25px;
}

.seccion-abajo{
   display: flex;
   justify-content: space-around;
   height: 20%;
}
```

JS:

```
document.querySelector("#btn-enviar").addEventListener("click", crearTarjeta);

let boxTarjeta = document.querySelector(".oculto");
let boxArriba = document.querySelector(".seccion-arriba");
let boxAbajo = document.querySelector(".seccion-abajo");

function crearTarjeta(event){
    event.preventDefault();

    //Seccion para traer los contenidos
    let nombre = document.querySelector("#nombre").value;
    let apellido = document.querySelector("#apellido").value;
    let profesion = document.querySelector("#profesion").value;
    let email = document.querySelector("#e-mail").value;
    let telefono = document.querySelector("#telefono").value;
    let direccion = document.querySelector("#direccion").value;

    //Seccion para crear los elementos
    let nodoNombre = document.createElement("p");
    let nodoApellido = document.createElement("p");
    let nodoProfesion = document.createElement("p");
    let nodoEmail = document.createElement("p");
```

```
    let nodoTelefono = document.createElement("p");
    let nodoDireccion = document.createElement("p");


    //Seccion para insertar contenido en los nuevos elementos
    nodoNombre.innerHTML = nombre;
    nodoApellido.innerHTML = apellido;
    nodoProfesion.innerHTML = profesion;
    nodoEmail.innerHTML = "Email - " + email;
    nodoTelefono.innerHTML = "Tel. - " + telefono;
    nodoDireccion.innerHTML = "Direccion - " + direccion;

    //Seccion para insertar en HTML
    boxArriba.appendChild(nodoNombre);
    boxArriba.appendChild(nodoApellido);
    boxArriba.appendChild(nodoProfesion);
    boxAbajo.appendChild(nodoEmail);
    boxAbajo.appendChild(nodoTelefono);
    boxAbajo.appendChild(nodoDireccion);

    //Para cambiar la clase de la tarjeta
    boxTarjeta.classList.remove("oculto");
    boxTarjeta.classList.add("tarjeta");

}
```

14. Si no utilizó Event Listener, reescriba los ejercicios utilizando buenas prácticas (.addEventListener). ¿Qué ventajas tiene desacoplar o quitar todo el código JS del código HTML?

Ya se hizo uso de Event Listener. La gran ventaja es descentralizar y no generar comportamiento en el enmaquetado de HTML. Además el proyecto es más mantenible teniendo las definiciones en un archivo afuera.

Observaciones:

Se optó por saltear estos ejercicios:

11. Crear una página en blanco, donde el usuario pueda clickear en cualquier lado y automáticamente se dibuje un DIV de color en esa posición exacta. Además, si el usuario hace click en uno de los divs creados, se debe mostrar un mensaje de alerta diciendo que “en esa posición ya existe un elemento”.

13. Hacer una ruleta básica donde salga un número al azar entre 0 y 36. El usuario debe apostar por un único número mediante un input. Luego de pulsar “Jugar!” debe mostrar el número que salió, el número apostado y si ganó o perdió.

15. Generar un script reutilizable que permita agrandar el tamaño de una imagen cuando posamos el mouse encima.

16. Extender el ejercicio 10 para que se puedan elegir con distintos botones distintos estilos de diseño de nuestra tarjeta personal.

17. Hacer un juego donde se pueda mostrar en una página una imagen de manera aleatoria y se debe adivinar a que se refiere la imagen. Pueden ser 5 imágenes de animales distintos y mediante un input el usuario debe ingresar el nombre del animal. Se debe mostrar si acertó o no.

Unidad 1.4: CSS Avanzado

Introducción

El objetivo de esta unidad es explicar algunas utilidades extra de CSS, para la mantenibilidad, usabilidad y reducción del código ante un proyecto grande.

Por ejemplo, se tiene al **selector universal**, que se encarga de seleccionar todo el archivo HTML. Se especifica con:

```
*{
  <lista estilos>
}
```

Herencia

En CSS, la herencia es el mecanismo por el cual las propiedades que posee una clase padre, se las delega a sus hijos, siendo recursivo. Se debe tener en cuenta que no todas las propiedades de CSS se pueden heredar. Por ejemplo, relacionadas con los tamaños de los elementos.

Por ejemplo en CSS, cada elemento heredable de un selector que no sea el HTML(ya que no tiene padre) posee el tipo **inherit** que significa que dicha propiedad se trae desde el padre.

Ver Buena Práctica N°1

Selectores Combinados

Se pueden combinar selectores para generar mayor especificidad. Se definen de la siguiente manera:

elemento.<class_elemento> . Por ejemplo, si se quiere que todos los párrafos comentados tengan el color azul, y el tipo de letra itálica, se debe definir lo siguiente:

```
p.comentario{

  color:blue;
  font-style: italic;

}
```

Se pueden definir selectores combinados con **elemento**#<id_elemento> pero no tiene mucho sentido porque el id se va a aplicar a un único elemento.

Selectores Anidados

Permite seleccionar elementos contenidos dentro de otros elementos. Esto permite aumentar el nivel de detalle. Se escribe desde el elemento menos específico al más específico, separados por **espacio**.

Por ejemplo si se quiere seleccionar los inputs que se encuentran dentro de un formulario, para que tengan un ancho de 60%:

```
form input {
   witdh:60%;
}
```

Grupo de Selectores

Se pueden agrupar varios selectores a la vez para asignarles propiedades en común. Las características distintas se seleccionan aparte. Se definen con una **coma**.

Ver Buena Práctica N°2

Ejemplo:

Definir los párrafos y los h3 de con un tamaño de 17px.

```
p,h3{
  font-size:17px;
}
```

Cascada

Este concepto es el corazón de este lenguaje de estilos. La cascada es el mecanismo por el cual se define el estilo final que tiene una propiedad en un elemento cuando tiene más de uno. Hay tres conceptos dentro de la cascada, que son: **importancia**, **especificidad** y orden en el código fuente.

Importancia

El orden en que se aplica la cascada es:

1. 1. Hoja de estilo default del navegador.
2. 2. Declaraciones normales en hojas de estilo de **usuario**.
3. 3. Declaraciones normales en hojas de estilo.
4. 4. Declaraciones importantes en hojas de estilo.
5. 5. Declaraciones importantes en hojas de estilo de **usuario**.

La palabra reservada **!important** sobreescribe toda cascada.
Ver Buena Práctica N°3.

Especificidad

Mientras más específico sea el selector, mayor prioridad tiene.

Orden

Si dos declaraciones afectan al mismo elemento, poseen igual importancia y especificidad. En este caso, el que esté declarado después termina ganando.

Pseudo Clases y Pseudo Elementos

Pseudo Clases

Se utilizan para definir un estado o comportamiento especial de un elemento.
Sintaxis:
elemento:pseudo-clase

Algunas pseudo clases son:

- link
- focus
- hover

Pseudo Elementos

Es usado para definir un estado o comportamiento a una parte del elemento.
Sintaxis:
elemento::pseudo-elemento

Algunos pseudo elementos son:

- ::after
- ::before
- ::first-child
- ::last-child
- ::first-letter
- ::first-line

Buenas Prácticas Unidad 1.4

Buena Práctica N°1

Es buena práctica usar herencia ya que se escribe menos código, y mejora la mantenibilidad y menos redundante.

Buena Práctica N°2

Agrupar permite nuevamente evitar la duplicación de código.

Buena Práctica N°3

Se recomienda no usar el important.

Práctico 1 - Parte IV - CSS Avanzado

1. ¿Qué significa herencia y cascada en el contexto de CSS?

Herencia es el mecanismo por el cual todas las propiedades asignadas a un elemento **(padre) se propagan hacia los elementos que contiene** (**hijos**).

Cascada es el mecanismo que decide con qué propiedad se queda un elemento frente a más de una configuración.

2. ¿Cómo se vincula el DOM con los conceptos de herencia y cascada?

En Herencia el DOM se relaciona ya que este modelo muestra en un árbol qué elementos contienen a otros, y estos a otros.

Respecto de la Cascada se puede decir que mata el concepto de herencia cuando el elemento es más específico.

3. ¿Qué es una colisión? ¿Cuando se produce y cómo podemos distinguirla?

Una colisión se produce cuando a una propiedad de un elemento le alcanza más de un estilo aplicado. Esto suele suceder cuando por cuestiones de herencia, que hay duplicación de estilos de la misma propiedad entre padre e hijo, o si se define un elemento genérico, y luego se lo especifica más. También si para el mismo elemento se define más de una vez la misma propiedad.

4. ¿Cómo se controla el orden con el que se aplican las declaraciones de estilo CSS?

Los 3 conceptos que controlar este orden son la **importancia** , **especificidad** y **orden**.

6. Dado el siguiente fragmento de HTML y CSS conteste:

a. ¿Cuál es el color de fondo del texto?
b. ¿De qué color se va a ver el texto?

```html
<div class="container">
  <div id="middle">
    <div class="child1">
      Texto 1
    </div>
    <div class="child2">
      Texto 2
    </div>
    <div class="child3">
      Texto 3
    </div>
  </div>
</div>
```

```css
.container .child1 {
  background-color: inherit;
}
.container {
  background-color: black;
  color: white;
}
#middle {
  color: blue;
}
.child1 {
  background-color: red;
  color: yellow;
}
```

a) Se realiza el siguiente análisis:

En principio el container aplicaría por herencia el fondo negro a sus hijos. Como child2 y child3 no tiene aplicado el background-color ellos quedan con fondo negro. Respecto de child 1, está definido dos veces, y por especificidad termina ganando el primero, ya que usa selectores anidados, y como el color de fondo es con valor heredado, termina tomando el del middle, y este al no tener la propiedad, se va hacia el padre container, que posee fondo negro. Por lo tanto, el fondo es completamente negro.

b) El container posee color blanco para el texto, y en principio todas tendrán ese color. Luego el middle (que contiene a los 3 childs) sobreescribe el blanco del container por el azul. Cómo child 2 y 3 no tienen seteado estas características, quedan con color azul. El child 1 tiene el color amarillo, entonces al azul por su color.

7. Se desea mejorar la asignación de estilo y la hoja css del sitio del periódico. Aplicar buenas prácticas, empleando el concepto de herencia, combinando y anidando selectores, tratando de cumplir los siguientes requisitos:

- La fuente del diario debe ser la misma para todo el sitio (Si desea puede utilizar una Google Font (https://fonts.google.com/)
- El título h1 debe ser de un color específico y en mayúscula
- El título h2 debe tener un tamaño equivalente para todos pero su color debería ser distinto de acuerdo a la sección. (Colores distintos para Política, Deporte, Economía, etc)
- Los links dentros de los párrafos deben ser rojos y sin subrayar

- Los links dentro de las listas deben estar en cursiva y de color verde.

Observación: solo se han realizado los cambios de los elementos que hubiera presente en la página anterior.

Lo que se hizo en la hojas de estilo fue agregar aquellas clases que poseían atributos en común con los mismos valores. Hubo un ahorro de unas 30 líneas de código.

Así quedaron agrupadas las clases:

```
h1,h2,h4,p,.tituloNoticiaDep,.tituloNoticiaPol,.descripcionImagen{
    text-align: center;
}

h4,p{
    font-family: 'Times New Roman', Times, serif;

}

.content,.seccionNoticias,.tipoNoticias,.menu,.footer,.media{
    display: flex;
}

.header,.toolBar{
    width: 100%;
}

h2,.barItem,.barItemE,.barItemF,.barItemI{
    font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
}

h2,.comentario{
    font-style: italic;
}

.menu,.footer,.media{
    justify-content: space-around;
}

.media,.seccionNoticias{
```

```
    width: 75%;
}

.noticia,.noticiaInterna{
    background-color: rgb(190, 240, 240);
    border: solid 3px;
    border-color: black;
    margin-top: 7px;
}

table,thead,tr{
    border: red solid 3px;
}

.barItem,.barItemE,.barItemF,.barItemI{
    font-size: 100%;
    color:rgb(23, 26, 23);
    list-style-type: none;
    width:20%;
}
```

8. ¿Es buena práctica emplear !important en varias propiedades de la hoja de estilo?

Es mala práctica usar el important en propiedades del CSS. Esto es porque si se necesita darle prioridad a un elemento antes que otro solo se necesita darle mayor grado de especificación.

10. En cada ejemplo, explique por cual de los concepto de cascada (importancia, especificidad y orden) se resuelven las colisiones:

a)

CSS	HTML	RESULTADO

`p { color: blue;}`
`.destacada { color:green;}`
`p.destacada { color:red;}`
	`<p class='destacada' id='unico'> Este es un párrafo</p>`	Este es un párrafo

b)

CSS	HTML	RESULTADO
`.titular { color: red;}` `.verde {color: green}`	`<h1 class="titular verde">Nuevo título</h2>`	**Nuevo título**

c)

CSS	HTML	RESULTADO

`h1.titular { color: red;}`
`.verde {color: green}`
`h1 { color: blue !important;}`
	`<h1 class="titular verde">Nuevo título</h2>`	**Nuevo título**

a) En el primer caso se termina resolviendo por **especificidad**, p.destacada posee mayor descripción que p y .destacada.
b) Acá se debe tener en cuenta que a h1 se le asignaron dos clases (titular y verde). Como ambas clases están al mismo nivel de especificidad, termina ganando el **orden**, por eso el resultado es el verde el que predomina.
c) Acà es sencillo, no hace falta ver la especificidad ni orden porque la clase h1 posee la propiedad con el !important, entonces predomina la **importancia**.

11. A partir de **un único div** sin contenido (totalmente vacío) haga que se vea como la bandera de Francia usando solo CSS.

```css
div{
    height: 300px;
    width: 30%;
    background-image:linear-gradient(90deg,red 33%, white 33%, white 66.66%,
    blue 66.66%);
  }
```

12. Investigue si es posible cambiar el texto de un elemento mediante CSS. Le parece una buena práctica? En qué caso lo usaría?

No se puede cambiar directamente. Lo que se puede es usar una pseudo-clase tipo hover que al interactuar con el elemento, se elimine el texto o se modifique alguna propiedad.

Observaciones:

Se optó por saltear estos ejercicios:

5. Crear una página con body, div's, y contenido dentro de los divs, títulos párrafos, span. listas. Asignar propiedades de estilo y luego verificar que propiedades se heredan y cuáles no. ¿Cómo haría para modificar alguna propiedad heredada?

9. Modifique la barra de navegación realizada en los prácticos anteriores utilizando selectores anidados. ¿Cual de las dos soluciones considera conveniente?

13. Implemente el estilo completo para que un sitio web pueda mostrar un texto escrito como un libro. Los capítulos serán archivos html separados pero con un indice principal que nos lleve a cada uno de los capítulos. Debe respetarse la alineación, y formato de un libro.

- El índice puede hacerse con una tabla donde los links tengan un estilo especial.
- El título h1 del capítulo debe estar en cursiva y debe ser 20% más grande que el h2, 40% más grande que el h3 y 80% mayor al tamaño de la letra de un párrafo.
- Los párrafos deben estar separados como suelen mostrar los libros y la primer letra del primer párrafo debe ser el triple de tamaño que el tamaño del resto del párrafo.
- Los links dentro de los párrafos que hagan referencias externas al libro deben tener un formato especial.
- Los links que hagan referencias internas al mismo libro deben tener un formato en cursiva (distinto al anterior) y cuando se posa el mouse deben cambiar sus propiedades. Cuando fue visitado también deber dejarse un estilo que asi fue.
- Agregar un estilo especial para citas textuales.

Los dialogos pueden pueden hacerse con listas con un estilo de viñeta especial y en caso que en el dialogo se quiera agregar el nombre del personaje que habla, deberá tener un estilo resaltado.