| | |
|---|---|
| | **UNIVERSIDAD DON BOSCO**<br>**FACULTAD DE INGENIERÍA**<br>**ESCUELA DE COMPUTACIÓN** |
| **CICLO: 02/2020** | **GUIA DE LABORATORIO #3**<br>**Nombre de la Práctica:** **Construcción de layout responsive**<br>**Lugar de Ejecución:** **Centro de cómputo**<br>**Tiempo Estimado:** **2 horas**<br>**MATERIA:** **Lenguajes Interpretados en el Cliente** |

## I. OBJETIVOS

Que el estudiante:

- Utilice correctamente la meta-etiqueta viewport.
- Defina correctamente las media queries para aplicar reglas de estilo en función del ancho del viewport del navegador.
- Utilice todos los conocimientos adquiridos durante el transcurso de la materia para la construcción de páginas web responsivas.

## II. INTRODUCCION TEORICA

### DISEÑO WEB RESPONSIVO

A lo largo de los últimos años hemos visto un cambio fundamental en el modo en que los usuarios consumen contenidos web. El auge de los dispositivos móviles ha hecho que el planteamiento de diseñar un sitio web para un tamaño fijo de pantalla haya dejado de ser una buena práctica.

**¿Qué es el diseño web responsivo?**

El diseño Web Responsivo (Responsive Web Design en inglés), es una técnica de diseño y desarrollo web que, mediante el uso de estructuras e imágenes fluidas, así como de media-queries en la hoja de estilo CSS, consigue adaptar el sitio web al entorno del usuario.

Toma las mejores prácticas para aplicarlas en la construcción de sitios, logrando buena calidad en las aplicaciones. La idea es que un solo sitio sea no solo adaptable a las características del recurso, sino que llegue a ser adaptativo.

En resumen, se trata de una estrategia de diseño que nos centrará en crear nuestros contenidos para que respondan de forma correcta independientemente del tamaño de pantalla o tipo de dispositivo en el que se estén visualizando.

**Implementando diseño web responsivo**

Hoy en día, una empresa que desee tener un sitio web "digno de ese nombre" deberá proponer un contenido que se visualice correctamente en todos los medios actuales: en una pantalla de ordenador (¡de 13" a 32"!), en una tableta táctil y en un smartphone.

La dificultad consiste, claro está, en lograr que nuestro contenido sea visible y legible a partir de esos tres tipos de pantalla. El objetivo es proponer una experiencia de uso, una interfaz web, un diseño web, que se adapte automáticamente a las diferentes resoluciones de pantalla que podría utilizar el usuario.

Para lograr nuestro objetivo, vamos a usar tres técnicas:

- Un **grid flexible**, que permita reorganizar la estructura de las páginas, de modo que puedan adaptarse a la resolución de la pantalla.
- **Imágenes flexibles**, para que su tamaño se adapte al tamaño de la pantalla de visualización.
- **Media queries**, que nos permitan saber cuál es el tipo y las dimensiones del dispositivo del visitante.

**Meta-etiqueta viewport**

El viewport en un navegador de una computadora es igual al área disponible para renderizar el documento web (o sea, le restamos toda la interfaz del navegador, como botones, barra de direcciones, barra de menús, barras de desplazamiento, etc.) Dicho de otro modo, es el área útil donde se mostrará la página web.

Para el caso de los dispositivos móviles el viewport no corresponde al tamaño real de la pantalla en píxeles, sino al espacio que la pantalla está emulando que tiene. Por ejemplo, en un iPhone, aunque la pantalla en vertical tiene unas dimensiones de 320 píxeles, en realidad el dispositivo está emulando tener 980 píxeles. Esto hace que ciertas páginas web (optimizadas para navegadores de escritorio) quepan en una pantalla de 320 píxeles, porque en realidad el Safari para iOS está emulando tener un espacio de 980 píxeles.

Lo interesante en este caso es que los desarrolladores somos capaces de alterar el viewport que viene configurado en el navegador, algo que resulta totalmente necesario si queremos que nuestra página se vea correctamente en dispositivos móviles.


Esta imagen tiene la misma foto que se muestra en la pantalla de un iPhone. Supongamos que la foto mide 320 píxeles de ancho. En la parte de la derecha tendríamos la foto a tamaño real, que es como se vería si tuviéramos un viewport configurado a 320 píxeles de ancho. Pero al verla en un iPhone con un viewport configurado a 980 píxeles de ancho, la imagen se verá bastante más pequeña.

La meta-etiqueta viewport fue creada en principio por Apple para su móvil predilecto, pero se ha convertido en todo un estándar que es soportado por la mayoría de los dispositivos móviles (smartphones, tablets y gran parte de móviles de gama media y baja). Como cualquier etiqueta meta, **el viewport se define dentro del head del documento web.**

Un ejemplo de etiqueta viewport sería el siguiente:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

- Con **width=device-width** conseguimos que el viewport sea igual a la anchura real de la pantalla del dispositivo, de modo que no se tratará de emular una pantalla mayor de lo que realmente es y veremos los píxeles reales.

- Con **initial-scale=1** conseguimos que no se haga zoom sobre el documento. El contenido de la web no se transformará, ni se agrandará, ni se hará menor.

- Con **user-scalable=no** conseguimos que el usuario no pueda hacer zoom en la página, con lo que siempre se mantendrán las medidas que nosotros hemos definido al construir la web.

-

La lista completa de parámetros que se pueden utilizar para construir la meta etiqueta viewport es la siguiente (los parámetros se separaran por comas):

- **Width**: ancho de la página (se puede establecer en píxeles o como device-width y usará el ancho del que dispone).
- **Height**: alto de la página, actúa igual que el width.
- **Initial-scale**: escala o zoom inicial de la página (este y los demás tipos de escala se establecen con valores como 1.0 para no tener zoom o 2.5 para tener un zoom del *2,5* de aumento, por ejemplo).
- **Minimum-scale**: zoom mínimo que podemos hacer en la página.
- **Maximum-scale**: zoom máximo que podemos hacer en la página.
- **User-scalable**: establece si está permitido o no hacer zoom (yes/no).

**MEDIA QUERIES**

Las Media Queries son una parte muy importante dentro del responsive design y nos permiten escribir estilos que respondan a los cambios al tamaño de la pantalla, la orientación o el medio en el que se visualiza el sitio web.

Las Media Queries nos permite seleccionar con toda precisión el medio de difusión, mediante criterios específicos o combinaciones de criterios. El resultado obtenido de esta búsqueda será de tipo booleano: el valor será verdadero o falso. De este modo podremos seleccionar una hoja de estilo en función de las respuestas obtenidas en la búsqueda.

Los criterios de búsqueda más importantes para la construcción de media queries son los siguientes:

- El **ancho de la zona de visualización**: width. Podemos examinar el ancho de la zona de visualización del navegador. Se permiten las variantes **min-width** y **max-width**. Es la principal expresión para el responsive design. Ejemplo: width: 780px.
- La **orientación** de la pantalla: orientation. Ejemplo: orientation: portrait o orientation: landscape. Resulta bastante práctico para examinar si el usuario usa su tableta táctil verticalmente (portrait) u horizontalmente (landscape).

Como ya se indicó, los media queries devuelven un resultado booleano: solo pueden ser verdaderos o falsos. Si la condición es verdadera, se aplica el estilo; si es falsa, será ignorada.

```
@media  (max-width:600px) {

      /*reglas a implementar*/

}
```

Este sencillo ejemplo tiene una media query que se interpreta de la siguiente manera: Cuando el ancho de la pantalla tenga un ancho menor a 600px se aplican las reglas de estilo encerradas dentro de las llaves que abren y cierran la media queries.

**Operadores lógicos**

Para la construcción de media queries útiles es necesario utilizar múltiples condiciones relacionadas mediante operadores lógicos.

- **Operador and**

El operador and es usado para colocar juntas múltiples funciones multimedia. Un query básico con el tipo de medio especificado como all puede lucir así:

```
@media (min-width: 700px) { ... }
```

Si se desea quiere aplicar ese query solo si la pantalla esta en formato horizontal, se puede utilizar el operador and y colocar la siguiente cadena:

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

**Operador or**

Al escribir media queries el operador or no existe como tal sin embargo es posible conseguir su comportamiento usando condiciones separados por coma.

Cuando utilizan condiciones separadas por comas y alguno de los queries retorna verdadero, el estilo o la hoja de estilos será aplicada.

Por ejemplo, si se quiere aplicar una serie de estilos para un equipo con un ancho mínimo de 700px o si el dispositivo está colocado en horizontal, debemos escribir la siguiente media query:

```
@media (min-width: 700px), (orientation: landscape) { ... }
```

§ **Operador not**

Es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las media queries.

**Aplicando media queries**

Las media queries se pueden aplicar básicamente en dos posibles puntos de la web:

- Al llamar al archivo css, indicando en cada uno las condiciones para cargarlo.

```
<link rel="stylesheet" media="(max-width: 800px)"
href="ejemplo.css" />
```

- En el propio archivo CSS, formando un apartado donde incluir fragmentos de css a aplicar

```
@media (max-width: 800px) {
  .prueba {
     display: none;
  }
}
```

**Creando puntos de cortes mediante media queries**

Típicamente las medias queries que se construyen para responsive design emplean como criterio de búsqueda el ancho del viewport. En ese sentido encontraremos tres formas de crear media queries:

- Aplicar solo en resoluciones de menos de X píxeles de ancho:

```
@media screen and (max-width:[ANCHO]px) {}
```

- Aplicar solo en resoluciones de más de X pixeles de ancho:

```
@media screen and (min-width:[ANCHO]px) { }
```

- Aplicar solo en resoluciones entre X e Y pixeles de ancho:

```
@media screen and (min-width:[ANCHO X]px) and (max-width:[ANCHO Y]px) { }
```

El proceso de creación de breakpoints (puntos de corte) puede llegar a ser muy completo por la gran diversidad de dispositivos y dimensiones de pantalla que presentan los dispositivos en el mercado de los dispositivos móviles.

Por ello, en lugar de trabajar con complejas matrices de dimensiones de dispositivos vamos a escribir nuestras medias queries de la forma más genérica posibles.

Durante el desarrollo de los ejemplos de la práctica estaremos usando tres puntos de corte para smartphones, tablets y PC.

```css
@media screen and (max-width: 480px) {

  /* Estilos para moviles */

}

@media screen and (min-width: 481px) and (max-width: 768px){

  /* Estilos para tablets */

}

@media screen and (min-width: 769px) {

  /* Estilos para PC */

}
```

Utilizando esos puntos de corte también podemos hacer que se aplique una o otra hoja de estilo según el tamaño del viewport.

```html
<link type="text/css" rel="stylesheet" href="css/movil.css" media="screen and (max-width:480px) ">

<link type="text/css" rel="stylesheet" href="css/tablet.css" media="screen and (min-width:481px) and (max-width:768px) ">

<link type="text/css" rel="stylesheet" href="css/pc.css" media="screen and (min-width:769px) ">
```

Usando la etiqueta viewport, las media queries y todos las propiedades y selectores CSS estudiados a la fecha, es posible crear sitios web que respondan al tipo de dispositivo y a los cambios en el ancho del viewport.

Por ejemplo, es posible que nuestro diseño para PC se presente en tres columnas, el diseño para Tablet en dos columnas y el diseño para smartphones se presente en una sola columna.

**Adaptando imágenes al diseño responsivo**

Para crear diseños muy optimizados puede ser importante usar varias imágenes distintas, una con un tamaño superior y otra con uno más ligero, pero por simplicidad nos vamos a centrar en el uso de una misma imagen (que en la medida de lo posible debería de estar optimizada) que se adaptará al tamaño del contenedor.

```
img{
        max-width:100%;
}
```

Con esto conseguiremos que la imagen se muestre con su tamaño original siempre y cuando este tamaño no supere el tamaño del contenedor.

**Probando los diseños responsivos**

Es muy común que para probar un diseño responsivo simplemente se cambien el ancho del navegador. Esta prueba, aunque no es incorrecta, si es muy limitada puesto que haciendo eso no es posible determinar la experiencia de navegación que obtendría el usuario en cada tipo de dispositivo.

Una mejor opción seria entonces probar nuestros diseños en diversos tipos de dispositivos para ver si realmente la experiencia de navegación es cómoda.

La mayoría de navegadores incluyen desde su "inspeccionador de elementos" la posibilidad de probar un sitio web en diferentes tipos de dispositivos. Adicionalmente existen extensiones para algunos navegadores que permiten verificar como se verían nuestros sitios en diversos dispositivos móviles.

## III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

| No. | Requerimiento | Cantidad |
|---|---|---|
| 1 | Guía de práctica #8: Objetos en JavaScript | 1 |
| 2 | Computadora con Sublime Text\|PHP Designer 2007 y navegadores instalados | 1 |
| 3 | Memoria USB o disco flexible | 1 |

## IV. PROCEDIMIENTO

**Ejercicio 1. Entendiendo las media queries**

1. Cree un archivo HTML llamado "Prueba.html".
2. Guardar el archivo en su carpeta de trabajo.
3. Escribir la estructura básica de un documento HTML en el archivo creado en el paso anterior.
4. Colocar como título de la página el texto "Probando las media queries".
5. Escribir la etiqueta meta-etiqueta viewport en el head de su página web.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

6. Coloque el siguiente código HTML dentro del cuerpo de su página web.

```
<div>
    <h1>Probando las media queries</h1>
    <p>El fondo de esta pagina se mostrará gris es versiones de escritorio,
    naranja en tablets y celeste en smartphones.</p>
</div>
```

7. Cree una carpeta denominada css. Dentro de esa carpeta, cree una hoja de estilo llamada prueba.css y vincúlela a su página web.

8. Digite las siguientes reglas de estilo dentro del archivo creado en el paso anterior.

```
*{
    font-family: 'Trebuchet MS', Helvetica, sans-serif;
}

h1{
    font-size: 2rem;
}

p{
    font-size: 1rem;
    text-align: justify;
}

div{
    background-color: white;
    border-radius: 10px;
    margin: 0 auto;
    padding: 2rem;
    box-sizing:border-box;
}
```

Las reglas de estilo anteriores se aplican indistintamente del ancho del viewport del navegador desde el cual se esté visualizando la página web.

9. A continuación procederemos a escribir los estilos que habrán de mostrarse en el diseño para smartphones mediante la siguiente media query:

```
@media screen and (max-width: 480px) {
  /*Estilos para moviles*/
   html{
    font-size: 14px;
  }
```

```
  body{
    background-color: #03A9F4;
  }

  div{
    width: 95%;
  }

}
```

10. A partir de las reglas de estilo anteriores responda: ¿Qué tamaño en pixeles tendrán los h1 y los párrafos de la página en su diseño para Smartphone?

11. Ahora escribiremos los estilos particulares para el diseño en tablets.

```
@media screen and (min-width: 481px) and (max-width: 768px){
  /*Estilos para tablets*/
  html{
    font-size: 15px;
  }

  body{
    background-color: #FF9800;
  }

  div{
    width: 85%;
  }

}
```

12. A partir de las reglas de estilo anteriores responda: ¿Qué tamaño en pixeles tendrá el padding del div en su diseño para tablets?

13. Proceda a escribir las reglas de estilo particulares para su versión desktop. En esta versión el tamaño de la fuente de la página será de 16 pixeles, el color de fondo de la página será #E0E0E0 y el ancho del div será de 75%.

```
@media screen and (min-width: 769px) {
  /*Estilos para PC*/

}
```

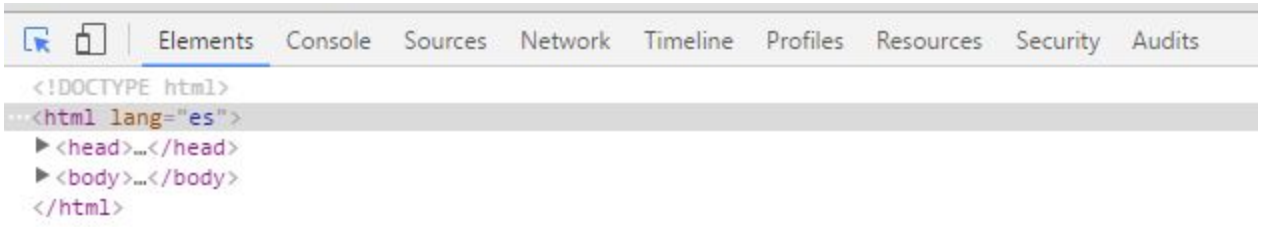
**Ejercicio 2. Probando el diseño responsivo**

1. Una forma sencilla (pero un poco limitada) de probar los diseños responsivos consiste simplemente en reducir el ancho del viewport del navegador. Pruebe de esta manera la página web construida en el ejercicio anterior y note como cambia el color de fondo una vez que se alcanzan los distintos puntos de quiebre.

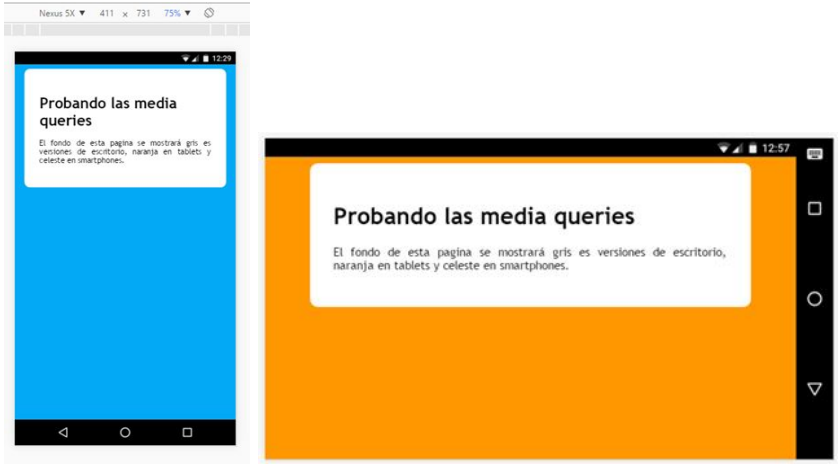


2. Los navegadores en su "inspeccionador de elementos" cuentan con distintas herramientas para probar los diseños responsivos. Mediante estas herramientas es posible verificar como se visualizaría la página en distintos dispositivos.

En **Google Chrome** para observar estas herramientas basta con abrir el inspeccionador de elementos y seleccionar la opción "Toggle device mode".

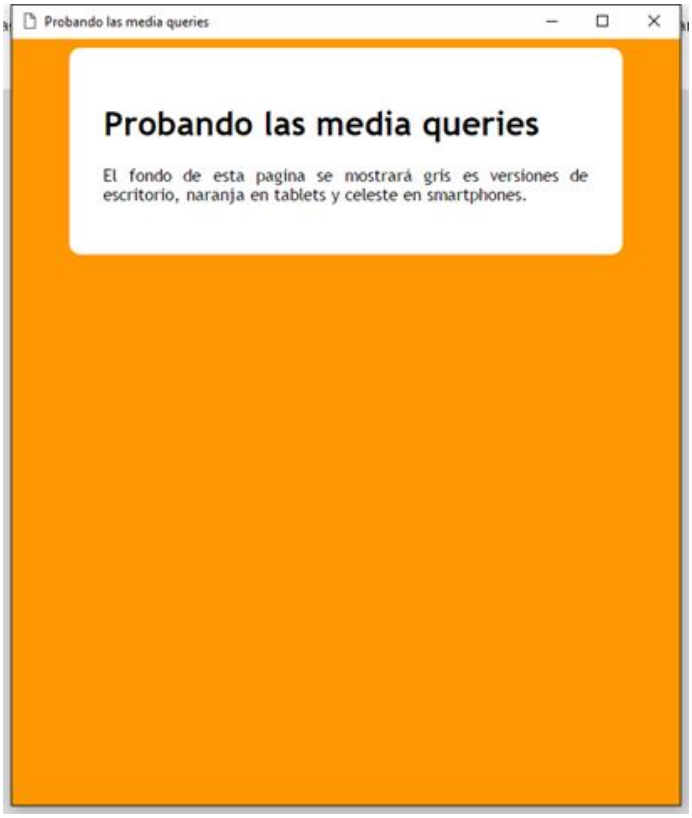



Luego basta con seleccionar alguno de los dispositivos disponibles y observar cómo se visualizaría la página en dicho dispositivo.



3. Adicional a estas herramientas, es posible añadir extensiones a los navegadores para probar los diseños responsivos. En esta práctica utilizaremos una extensión llamada **Responsive Web Design Tester** para Google Chrome. Dicha extensión puede agregarse mediante el siguiente vinculo: https://chrome.google.com/webstore/detail/responsive-web-design-tes/objclahbaimlfnbjdeobicmmlnbhamkg

4. Una vez instalado, el uso de esta extension es muy sencillo, simplemente se debe dar click en el icono de la extensión y seleccionar el dispositivo que se desea probar.




**Ejercicio 3. Construyendo un layout responsivo**

1. Para este ejercicio retomaremos el ejemplo de la agencia de viajes de la guía anterior. Los archivos HTML a ocupar son exactamente iguales a los empleados en la guía anterior, sino dispone de dichos archivos puede ocupar los que se le proporcionan como recursos.

2. Agregue la meta-etiqueta viewport en las tres páginas del sitio (index.html, planes.html y galería.html)

3. A continuación debe agregar en la cabecera de **sus tres paginas** las llamadas a las hojas de estilo en función del ancho del viewport del navegador.

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

<link type="text/css" rel="stylesheet" href="css/movil.css" media="screen
and (max-width:480px) ">

<link type="text/css" rel="stylesheet" href="css/tablet.css" media="screen
and (min-width:481px) and (max-width:768px) ">

<link type="text/css" rel="stylesheet" href="css/pc.css" media="screen and
(min-width:769px) ">
```

Note que la hoja de estilo "base.css" se aplica indistintamente del ancho del viewport del navegador o de cualquier otro criterio. Las hojas de estilo móvil.css, tablet.css y pc.css se aplicaran según se cumpla o no la condición de las media queries (Note que se están usando los mismos puntos de ruptura del ejercicio 1).

4. Cree dentro de su carpeta css la hoja de estilo llamada "**base.css**". Las reglas de estilo que escribamos en este archivo se van a aplicar a la página independientemente del dispositivo desde el cual se visualice.

5. Incluir las siguientes reglas de estilo dentro del archivo creado en el paso anterior.

```
/* FUENTES */

@import url(http://fonts.googleapis.com/css?family=Didact+Gothic);

*{

	font-family: 'Didact Gothic', sans-serif;

	margin: 0;

	padding: 0;

}


body{

	background-color: #BBB;

	color: #444;

}


h1{

	font-size: 1.3em;

	text-align: center;

	color: #464d52;

}



/* PARTE SUPERIOR */
```

```
#superior{

	width: 100%;

	padding: 2% 0px;

	background-color: #464d52;


}


/* PIE DE LA PAGINA*/

footer{

	text-align: center;

	margin: 0;

	font-size: .9em;

	line-height: 2;

	clear: both;

	background-color: #464d52;

	color: white;

	width: 100%;

}



/*BOTON DE LA PAGINA INDEX */


.boton{

	background-color:#464d52;

	padding: 7px;

	margin-top: 10px;
```

```
        border-radius: 5px;
        color: white;
        font-size: 0.9em;
        text-align: center;
        vertical-align: middle;
        cursor: pointer;
        transition: background 3s, color 1s;

}

.boton:hover{
        background: #5077c7;

}

.boton a{
        text-decoration: none;
        color: white;
        width: 100%;
        margin: 0px;
        display: block;
}
```

**Construyendo el diseño para desktop**

Comenzaremos construyendo el diseño para escritorio, este diseño **será prácticamente el mismo diseño fluido que realizamos en la práctica anterio**r.

6. Cree una hoja de estilo llamada **pc.css** y ubique en este archivo las siguientes reglas de estilo:

```
/*    MENU */

nav{
        width: 100%;    /* 960/960*=100%*/
        font-size: 1.4em;

}

nav ul{
        list-style: none;
        margin: 0px 0px 65px 0px;
}

nav ul li{
        float: left;
        margin: 0px;
        padding: 10px 0px 0px 0px;
        border-top: solid 2px white;
}

nav ul li a {
        text-decoration: none;
        padding: .3em .5em;
        margin: 0em .5em;
```

```
        display: block;
        color: white;
        transition: color 1s, transform 3s;
}




nav ul li a:hover{
        color: #AAA;
        transform:scale(1.5);
}

nav ul li a#inicio{
        background: url(../img/turismo/home.png) no-repeat 0px 3px;
        padding-left: 40px;
}
nav ul li a#planes{
        background: url(../img/turismo/planes.png) no-repeat 0px 3px;
        padding-left: 40px;
}



nav ul li a#galeria{
        background: url(../img/turismo/galeria.png) no-repeat 0px 3px;
        padding-left: 40px;
}
```

```
/* PARTE SUPERIOR */

#encabezado{

	margin: 0px auto;

	width: 80%;

	min-width: 650px;/* YA NO REACCIONARA SI LA PANTALLA ES MAS PEQUEÑA QUE
650px*/

	max-width: 960px;/*YA NO REACCIONARA SI LA PANTALLA ES MAS GRANDE QUE 960
px*/



}



#superior #encabezado #logo{

	margin: 2% 0px; /*23/960=2%*/

	width: 21%; /*200/960= 21%*/

	height: auto;

}



#imgDesc img{

	width: 62.5%; /* 600/960= 62.5% */

	height: auto;

	float: left;

}



#imgDesc{

	overflow: hidden;/*obliga a cubrir a los elementos flotantes*/
```

```
}



#descripcion{
	width: 37.5%; /* 360/960=0.375*/
	background-color: #888;
	float: left;
}

#descripcion p{
	margin: 5%; /* 20/360 =5% */
	color: white;
	font-weight: bold;
	line-height: 170%;
	text-align:justify;
	font-size: 0.95em;
}



/*Contenido principal de las paginas */
#contenido{
	width: 80%;
	margin: 0px auto;
	clear: left;
	overflow: auto;
	min-width: 650px;/* YA NO REACCIONARA SI LA CAJA ES MAS PEQUEÑA DE 650px*/
```

```
        max-width: 960px;/*YA NO REACCIONARA SI LA CAJA ES MAS GRANDE QUE 960 px*/


}



/*   CONTENIDO DE LA PAGINA INDEX */


.plan{
        width: 26.7%; /* 258/960= 27%*/
        padding: 2%; /* 20/960=2%*/
        margin: 30px 1.5%;
        background-color:white;
        border: solid 1px black;
        border-radius: 10px;
        float: left;
        text-align: justify;


}


.plan img{
        width: 100%;
}


/* Primer div, cuarto div,... */
#contenido div:nth-child(3n+1){
        margin-left: 0px;
```

```
}

/* Tercer div, sexto div, .... */
#contenido div:nth-child(3n){
	margin-right: 0px;

}

/*   CONTENIDO DE LA PAGINA PLANES */

.planes{
	width: 94%;
	background: white;
	padding: 1% 3%;
}

.planes img{
	width: 100%;
	border-radius: 15px;
}

.planes h1{
	text-align: left;
	font-size: 1.5em;
```

```
}


.planes .precio{
	font-size: 1.1em;
	margin: 20px 0px 5px;
	padding-bottom: 15px;
	border-bottom: solid 2px grey;
	font-weight: bold;
}




/*   CONTENIDO DE LA PAGINA GALERIA*/


#galeria .plan{
	font-size: 1em;
	padding-bottom: 10px;
	margin: 30px 1.4% 10px;
	display: inline-block;
	float: none;
	vertical-align: top;
	text-align: center;
	font-style: italic;
}
```

7. Visualice los resultados en su navegador y notara que cuando el ancho del viewport es mayor o igual a 769 pixeles el sitio web funciona correctamente

**Construyendo el diseño para tablets**

A continuación procederemos a realizar el diseño para tabletas. Salvo un par de cambios en la cantidad de columnas en la que se presenta la información, este diseño será bastante parecido al diseño de escritorio.

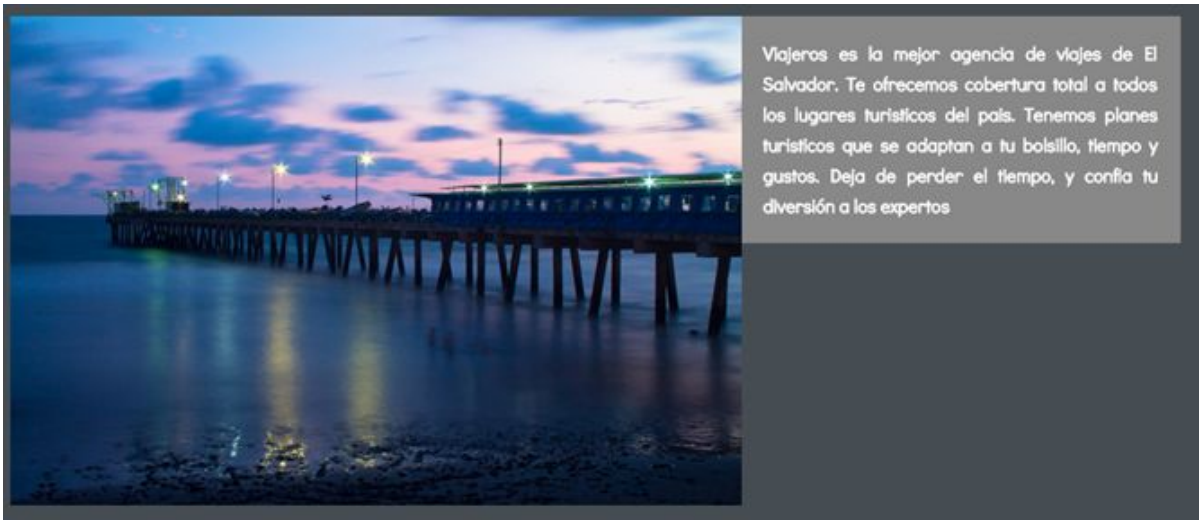
8. Cree una copia de su archivo pc.css y nómbrela "**tablet.css**". A continuación se explica los cambios que deben hacerse en las reglas de estilo de ese archivo. Cualquier elemento del que no se presente un cambio, debe asumirse que se mantiene exactamente igual.

9. Dado que el espacio es un poco más limitado en las tablets que en las computadoras de escritorio, debemos replantear el aprovechamiento que se hace de dicho espacio. El primer cambio que realizaremos es que la caja con id "encabezado" se presentará con un ancho del 90% (en la versión desktop era 80%). Además eliminaremos el min-width y el max-width. Esa regla de estilo debería quedar de la siguiente manera:

```
/* PARTE SUPERIOR */

#encabezado{
    margin: 0px auto;
    width: 90%;
}
```

10. A continuación modificaremos la parte superior de la página. En la versión desktop presentaamos la imagen y la descripción flotando a la derecha de la imagen. En la versión para tablets no tenemos suficiente espacio para mostrar los elementos a la par, por lo que mostraremos la imagen a ancho completo y la descripción por debajo de la imagen.




Versión desktop

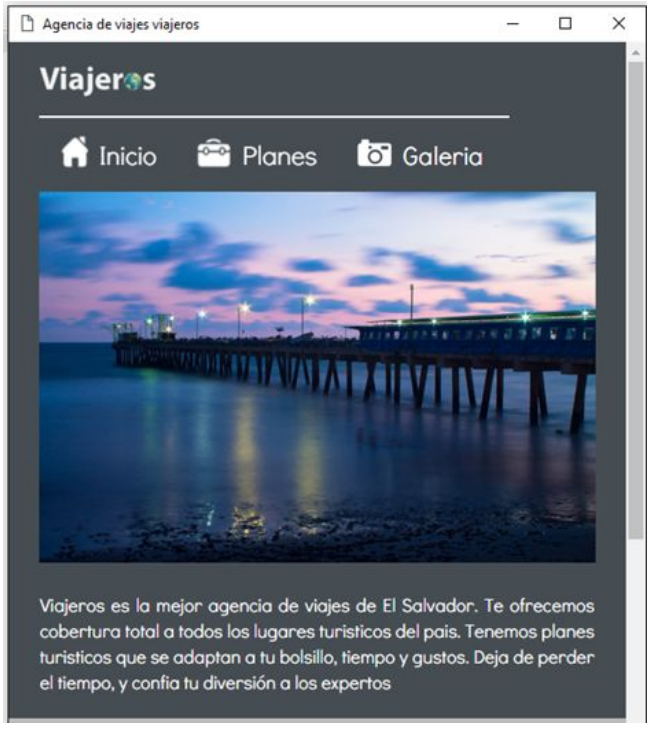
11. Para conseguir el resultado anterior debe modificar las reglas de estilo de los selectores #imgDesc img, #descripcion y #descripcion p de tal forma que luzcan de la siguiente manera:

```css
#imgDesc img{
    width: 100%;
    height: auto;
    float: none;
}

#descripcion{
    width: 100%;
}

#descripcion p{
    margin: 20px 0px 10px 0px;
    color: white;
    line-height: 140%;
    text-align:justify;
    font-size: 1em;
}
```

Ahora el resultado obtenido debería ser similar al siguiente:




Versión para tablets

12. Así como aumentamos el ancho de la caja con id “encabezado” también debemos aumentar el ancho de la caja con id “contenido”. Además se deben eliminar los anchos mínimos y máximos que definimos para la versión desktop. La regla de estilo debe lucir de la siguiente manera:

```
#contenido{
    width: 90%;
    margin: 0px auto;
    overflow: auto;
}
```

13. A continuación cambiaremos el diseño de la página index.html. En su versión para desktop esta página mostraba su contenido mediante 3 columnas.


14. En el diseño para tablets el contenido se mostrará a una sola columna con la imagen flotando a la izquierda y con el texto a la par de la imagen. Las reglas de estilo de este punto deberían lucir de la siguiente manera:

```css
/*   CONTENIDO DE LA PAGINA INDEX */

.plan{
    width: 94%;
    padding: 1.5%;
    margin: 20px 1.5%;
    background-color:white;
    border: solid 1px black;
    border-radius: 10px;
    float: left;
    text-align: justify;
}

.plan img{
    width: 50%;
    height: auto;
    float: left;
    margin-right: 20px;
}

.boton{
    width: 40%;
    float: right;
}
```

Nótese que se han eliminado las reglas de estilo que tienen los selectores **#contenido div:nth-child(3n+1)** y **#contenido div:nth-child(3n)** puesto que como el diseño es a una sola columna no debemos controlar esos márgenes.

El resultado obtenido debería ser similar al siguiente:

15. La página planes.html presenta su contenido en una sola columna, por tanto no tendremos que realizar ninguna modificación en esta sección.

16. La página **galería.html** presenta su contenido en tres columnas (par la versión desktop). En la versión para tablets conseguiremos que el contenido se muestre en dos columnas. Las reglas de estilo para esta página quedarían de la siguiente manera:

```css
/*   CONTENIDO DE LA PAGINA GALERIA*/
#galeria .plan{
    font-size: 0.8em;
    padding-bottom: 10px;
    margin: 10px 1%;
    display: inline-block;
    float: none;
    vertical-align: top;
    width: 44%;
    text-align: center;
    font-style: italic;
}
```

```css
#galeria .plan p{
    clear: both;
}

#galeria .plan img{
    width: 90%;
    height: auto;
    margin: 5px 5%;
}
```

Hasta este punto su diseño para tablets debería de funcionar correctamente. Pruebe todas las paginas del sitio y corrija cualquier error detectado.

**Construyendo el diseño para smartphones**

A continuación procederemos a realizar el diseño para smartphones. En este diseño se debe ser mucho más cauto en cuanto al aprovechamiento del espacio puesto que este es mucho más limitado en este tipo de dispositivos.

17. Cree una copia de su archivo tablet.css y nómbrela "**movil.css**". A continuación se explica los cambios que deben hacerse en las reglas de estilo de ese archivo. Cualquier elemento del que no se presente un cambio, debe asumirse que se mantiene exactamente igual.

18. Lo primero que haremos es modificar el menú para aprovechar mejor los espacios. **En la versión para móviles preferimos no mostrar los iconos de los elementos del menú.** Los estilos para el menú deberían lucir de la siguiente manera:

```css
/*   MENU */

nav{
    width: 100%;
    font-size: 1.2em;
}

nav ul{
    list-style: none;
    margin: 0px 0px 5px 0px;
}

nav ul li{
    float: left;
    margin: 0px -15px;
    padding: 10px 0px 0px 0px;
    border-top: solid 2px white;
    border-bottom: solid 2px white;
}
```

```css
nav ul li a {
    text-decoration: none;
    padding: .3em .5em;
    margin: 0em .5em;
    display: block;
    color: white;
    transition: color 1s, transform 3s;
}


nav ul li a:hover{
    color: #AAA;
    transform:scale(1.3);
}
```

**Nota: En el diseño para móviles no se mostraran los iconos en los elementos del menú.**

19. En la parte superior de la página se ha decidido no mostrar la imagen para dejar más espacio para el contenido de las páginas. Las reglas de estilo para la parte superior deberían lucir de la siguiente manera:

```css
/* PARTE SUPERIOR */


#encabezado{
    margin: 0px auto;
    width: 90%;
}

#superior #encabezado #logo{
    margin: 10px 0px;
    width: 120px;
    height: auto;
}


#imgDesc{
    overflow: auto;
    width: 100%;
}
```

```css
#imgDesc img{
    width: 100%;
    height: auto;
    display: none;
}

#descripcion{
    width: 100%;

}

#descripcion p{
    margin: 20px 0px 10px 0px;
    color: white;
    line-height: 140%;
    text-align:justify;
    font-size: 0.8em;
}
```

El resultado obtenido seria el siguiente:


20. Para el caso de la página index.html el diseño se presentará a una columna como en el diseño para tablets pero con la imagen a ancho completo y el texto debajo de la imagen. Las reglas de estilo de esta página deberían lucir de la siguiente manera:

```css
/*   CONTENIDO DE LA PAGINA INDEX */

.plan{
    width: 92%;
    padding: 10px 2% ;
    margin: 20px 1%;
    background-color:white;
    border: solid 1px black;
    border-radius: 10px;
    text-align: justify;
}

.plan img{
    width: 100%;
    height: auto;
}

.boton{
    width: 92%;
    margin: 5px auto;
}

.plan p{
    font-size: 0.8em;
}

.plan h1{
    font-size: 1em;
}
```

21. Finalmente el contenido de la página galería.html se presentará a una sola columna. Las reglas de estilo para esta página serían las siguientes:

```css
/*   CONTENIDO DE LA PAGINA GALERIA*/

#galeria .plan{
    font-size: 0.8em;
    padding-bottom: 10px;
    margin: 10px  auto;
    float: none;
    width: 90%;
    text-align: center;
    font-style: italic;
}


#galeria .plan p{
    clear: both;
}

#galeria .plan img{
    width: 90%;
    height: auto;
    margin: 5px 5%;
}
```

Pruebe el diseño para móviles y corrija cualquier error detectado.

## V. DISCUSIÓN DE RESULTADOS

1. realizar un sitio web sobre la liga española de Futbol, de tal forma que el sitio web se visualice de forma correcta en computadoras, tablets y smartphones.

## VI. BIBLIOGRAFÍA


- Deitel, Paul / Deitel, Harvey / Deitel, Abbey. Internet & World Wide Web. Cómo programar. 5a. Edición. Editorial Pearson. 2014. México D.F..