Responsive Web

Profesor: Eduardo Palacio

¿Qué es?

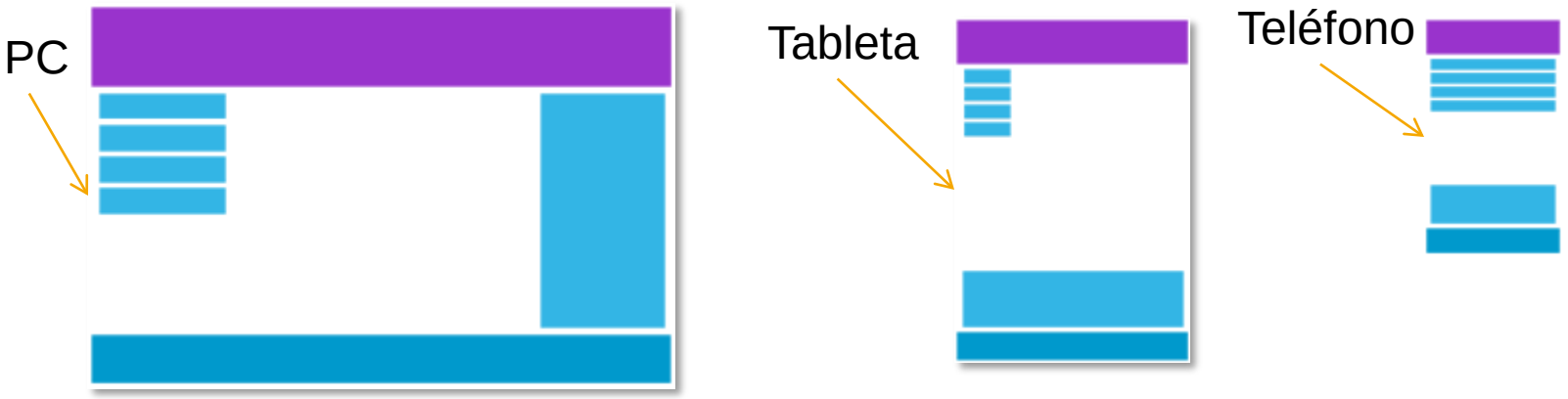
- Responsive web permite que tus paginas Web se vean bien en cualquier dispositivo teléfono, tableta o PC
- Mediante el uso de CSS y HTML se consigue que la pagina Web se vea bien en cualquier pantalla incluso al redimensionarla.
- Los elementos de una pagina Web así desarrollada se pueden reubicar según el dispositivo y el tamaña de pantalla.
- Pretende la satisfacción del usuario al acceder a nuestra información

¿Qué no es?

- No tiene nada que ver con el uso de JavaScript o cualquier otro lenguaje.

¿Cómo?

- Adaptando el tamaño.
- Oculta información innecesaria.
- Redistribuye los contenidos para adaptarlos al dispositivo.



La pantalla

- The Viewport.- es el área visible de la pagina.
  - Cambia dependiendo del dispositivo
- Antes de la aparición de los dispositivos móviles el diseño se realizaba solo para los PC.
- El diseño de la página era estático y desde el punto de vista del tamaño y de los componentes.
- Con la aparición de los dispositivos móviles empezaron los problemas de tamaño para adaptar los contenidos según las diferentes tamaños de pantalla.
- La primera solución de los navegadores fue escalar y mantener el diseño dentro de la pantalla.

Configuración HTML

- HTML5 introduce una etiqueta <meta> para controlar el viewport
- Ahora todas las páginas deben de incorporar
  - <meta name="viewport" content="width=device-width, initial-scale=1.0">
    - width=device-width para ajustar el tamaño al ancho
    - initial-scale=1.0 para definir el zoom en la carga de la página

Configuración HTML: Ejercicio

- Ejercicio 1: Crear una pagina Web con una imagen y un texto “loren ipsum” y probar la diferencia entre ponerle y quitarle el meta viewport

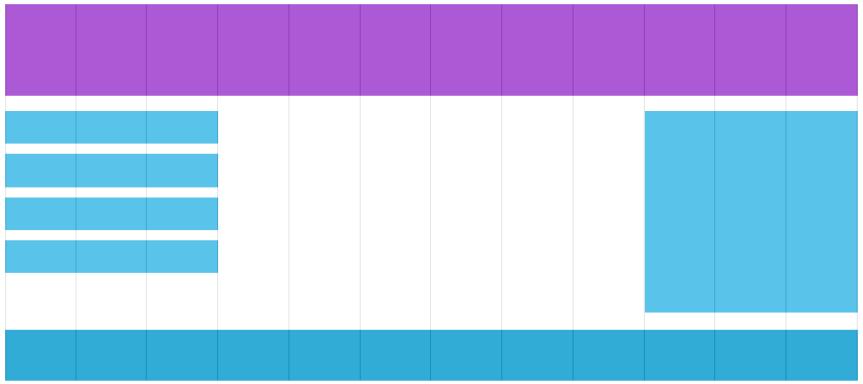
Reglas del ViewPort

- Como la gente suele usar el scroll vertical mas que el horizontal si tu página los obliga a usarlo o a tener que hacer zoom les resultará molesto. Ten en cuenta estas reglas para evitarlo.

1. No uses elementos cuyo tamaño horizontal (ancho) sea fijo y pueda exceder del tamaño del viewport.
2. No hagas depender la visualización de tu pagina de una configuración de pantalla.
3. Usa los medios que te aporta CSS para configurar la pantalla y adaptarla a cada tamaño.
    - No uses tamaños absolutos usa relativos witdh:100%
    - No uses valores grandes para las posiciones de elementos de forma absoluta.

El fundamento de Responsive

- Grid-View.- es la base y consiste en dividir la pagina (ViewPort) en columnas.
- Generalmente se usan 12 columnas con un ancho de 100%
- Las columnas las puedes estirar y encoger cuanto quieras.



Modelo Box-sizing

- Antes de nada configura la pagina para que el padding y el borde estén incluidos en el tamaño total del elemento.
- Antes al ancho que se quería se restaba el padding y el borde.

```
* {
    box-sizing: border-box;
  }
```

Modelo Box-sizing: Ejercicios

- Ejercicio 2: Prueba esta pagina con y sin el modelo Box-sizing.

```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
    width: 300px;
    height: 100px;
    border: 1px solid blue;
    box-sizing: border-box;
}

.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 1px solid red;
    box-sizing: border-box;
}
</style>
</head>

<body>

<div class="div1">Mismo tamaño</div>
<br>
<div class="div2“>Iguales</div>

</body>
</html>
```

Modelo Box-sizing: Ejercicios

- Ejercicio 3: Ahora trata de crear tal y como vimos el otro día una pagina que conste de cabecera, pie, aside y article.
  - Dale tamaños width en porcentaje y no olvides el float:left|right;

```
.aside {
        width: 25%;
        float: left;
    }
.article{
        width: 75%;
        float: left;
    }
.cabecera pie{
    border: 1px solid red;
    padding: 15px;
}
```

- Como ves ya empieza a funcionar algo… pero todavía queda

Modelo Box-sizing: Ejercicios

- En el ejercicio anterior las cosas no están mal para dos columnas pero aún nos permite poco control de los elementos.
- Para tener mas control debemos de usar un grid mas preciso (con mas columnas) como veremos en el siguiente ejercicio.

Modelo Box-sizing: Ejercicios

- Ejercicio4: Establecer un grid de 12 columnas:
  - como el ancho es 100% /12 columnas = 8,33% el ancho de cada columna.
  - Ahora crea una clase class="col-x" para cada una de ellas y establece el ancho (width).
    - Ej:

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Modelo Box-sizing: Ejercicios

- Ejercicio 4.1: Ahora haz que las columnas floten a la izquierda y tengan un padding de 15px

```
[class*="col-"] {
        float: left;
        padding: 15px;
        border: 1px solid red;
    }
```

Modelo Box-sizing: Ejercicios

- Ejercicio 4.2: Ahora establecemos las columnas por las que se extiende cada fila sabiendo que el máximo es 12. el calculo es el siguiente:

  Como tengo 12 columnas entonces 4grupos x 3columnas =12 columnas

  - Cada tres columnas ocupa un ancho de 25% de la pagina.
  - Los bloques los creo con nuestro amigo
    - Por ejemplo:

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

Modelo Box-sizing: Ejercicios

- Ejercicio 4.2: Cuidado porque como las columnas tienen la propiedad float:left, el resto de elementos que estén en la pagina a continuación de las columnas se salen de esta (saliéndose 100% de vista que establecen las columnas) y no harán caso de las columnas. Para controlar esto se establece este estilo que limpia e inicializa el flujo de la pagina.

```
.row::after {
    content: "";
    clear: both;
    display: table;
}
```

- Con esto ya tienes todo lo necesario para crear tu pagina responsive ahora añade colores y estilos para que sea bonita.

Modelo Box-sizing: Ejercicios

- Ejercicio propuesto.
  - Ejercicio 5: intenta que cada vez que se pincha en el menú se muestre solo lo que enlaza de modo que no tengamos el resto de texto visible.

    (pista: recuerda los estilos z-index, display y visibility)

https://www.w3schools.com/cssref/default.asp

Modelo flexbox

- Es un modelo también nuevo CSS3.
- Permite predecir como se comportarán los elementos de la página con los diferentes tamaños de pantalla en los diferentes dispositivos.
- Es una importante mejora con respecto al modelo de bloques.
- No utiliza float.
- Los márgenes del contenedor no se colapsan con los márgenes del contenido.
- Consiste en contenedores y contenido flexible.

Modelo flexbox

- Para usar este modelo creamos un contenedor
- Y usamos el atributo de css display con los dos valores siguientes:
  - “flex” para un contenedor de modelo bloque
  - “inline-flex” para un contenedor de modelo en línea
  - Luego tenemos que poner dentro los contenidos (elementos)

Cuidado flexbox solo indica como los contenidos son renderizados, todo lo que este fuera usa el modelo de pagina normal.

Modelo flexbox

- Los elementos dentro del contenedor flexible ocupan toda la línea
- Flexbox solo indica como los elementos se colocan en el contenedor
- EjemploFlex 1. Crea una pagina web con tres elementos class="elemento-flex" dentro de un contenedor class="contenedor-flex"

Recuerda el modelo en bloque y en línea y como crear contenedores con div.

Usa el modelo de caja para ver el espacio que ocupa cada caja.

Modelo flexbox

- EjemploFlex 2. Ahora cambia como aparecen los bloques respecto de la vertical usa direction=“rtl”
- EjemploFlex 3.Ahora cambia la dirección en la que aparecen los elementos en el contenedor por defecto es “row” (linea) usa flex-direction=“row | row-reverse | column | column-reverse”.
- EjemploFlex 4. Ahora cambia como se justifican los contenidos justify-content=“flex-start | flex-end | center | space-between | space-around” dentro del contenedor siempre que tengas espacio.

Modelo flexbox

- EjemploFlex 5. Ahora cambia la dirección en la que aparecen los bloques usa align-items=“stretch | flex-start | flex-end | center | baseline”
- EjemploFlex 6. Describe como se colocan en la linea del contenedor dependiendo de esto pueden o no fluir a otras lineas flex-wrap=“nowrap | wrap | wrap-reverse”
- EjemploFlex 7. align-content modifica el comportamiento de la anterior (necesita que esté definida flex-wrap), es como align-items , pero en lugar e alinear elementos alinea lineas.
  - hora cambia como se justifican los contenidos align-content=“stretch | flex-start | flex-end | center | space-between | space-around” dentro del contenedor siempre que tengas espacio

Modelo flexbox

- Estas propiedades se definen para los elementos:
  - EjemploFlex 8. order permite especificar un nuevo orden relativo de los elementos, order=-1 | 0 | 1;
  - EjemploFlex 9. la propiedad margin[- (left | right | top | down)] : auto quita el espacio extra y permite colocar a los elementos en diferentes posiciones.
  - Ejemplo 10. Centra un elemento con respecto al alto y ancho con margin :auto;
  - Ejemplo11. Reinicia el valor de align-items para un elemento con los valores de align-self=“stretch | flex-start | flex-end | center | baseline”
  - Ejemplo12. Para poner el tamaño con relación a los otros elementos usa flex:valor;

Modelo flexbox

- Este es un modelo complicado únicamente porque tienes que tener claro quien es el contenedor y quien es el elemento.
- Ya que para que funcione las propiedades se deben aplicar de forma clara.
- Las propiedades de contenedores para contenedores.
- Las propiedades de elementos para elementos de forma individualizada.
- Recuerda la estructura del modelo:
  - contenedorFlexible contiene elementosFlexibles y cada elementosFlexibles luego se trata como un elemento individual.

Recordatorio

- CSS selector
- Animaciones
- Tipos de letra seguros
- Caracteres especiales por css
- CSS