Laboratorio #2

CSS Adaptable

1. Flexbox

Flexbox es un método de diseño de página unidimensional para compaginar elementos en filas o columnas. Los elementos de contenido se ensanchan para rellenar el espacio adicional y se encogen para caber en espacios más pequeños. Se basa en un contenedor flexible (flex container), que a su vez contiene varios elementos flexibles (flex items). El contenedor otorga sus propiedades a los elementos, es decir: los elementos o flexboxes deben su flexibilidad al hecho de estar dentro del contenedor.

Para definir un flex container se emplea *flex* como valor para la propiedad CSS *display*. Esto nos permite maquetar nuestras páginas web de una manera mucho más fácil en comparación con la forma tradicional, en la que utilizamos propiedades como *float* o *position*, entre otras.

Flexbox es una herramienta poderosa y específica para crear layouts flexibles en una dimensión, mientras que Flex hace referencia a la capacidad de los elementos para adaptarse de forma flexible dentro de un contenedor.

Como sistema de disposición de elementos, Flexbox apareció en la esfera del desarrollo web en 2009. Aunque es relativamente nuevo, este sistema ha probado ser igual o más potente que otros sistemas de posicionamiento clásico.

Existen dos tipos de contenedores flexibles: de bloque y en línea.

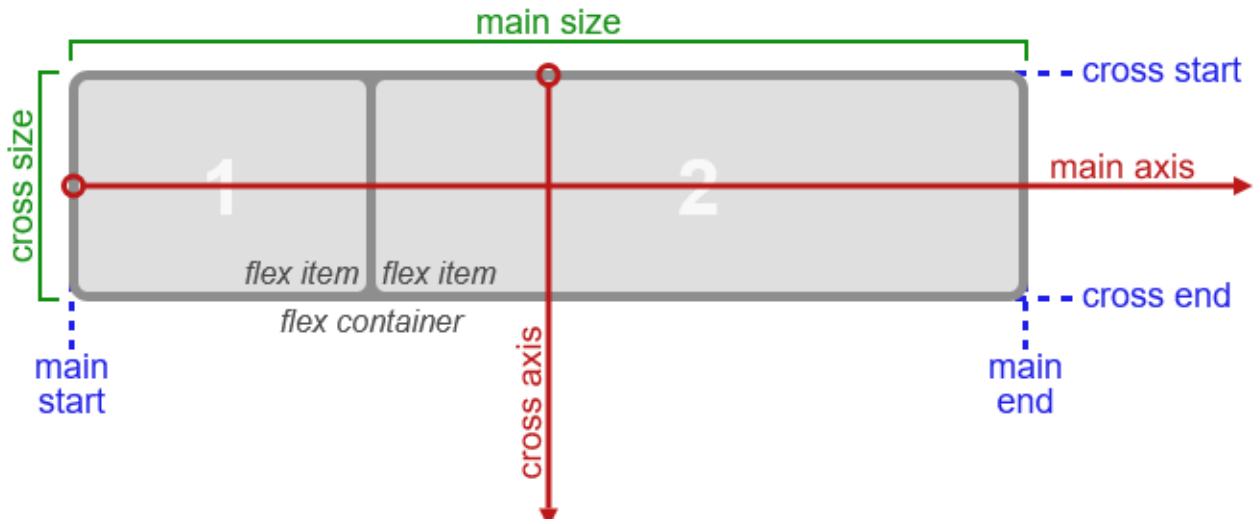


Figura 1: Distintas direcciones y términos de tamaño aplicados a un contenedor flexible en fila.

Flexbox hoy ya es una realidad, puesto que los navegadores modernos lo soportan y lo

interpretan correctamente. Por tanto, está listo para ser usado en cualquier tipo de proyecto. El único navegador que no lo soporta es Internet Explorer en versiones antiguas, como IE8 o IE9. Lo cual puede ser una razón para no visualizar correctamente el uso de flexbox, por no ser compatible con esta propiedad CSS, siendo necesario actualizar la versión del navegador o cambiar a otro.

Figura 2: Compatibilidad flexbox con navegadores.

Características de Flexbox

Algunas de las características clave de Flexbox son:

- La capacidad de ajustar fácilmente el diseño y la posición de los elementos de una página, independientemente del tamaño o la orientación de la pantalla.
- Compatibilidad con sitios web de diseño adaptable y adaptados a dispositivos móviles, con mínimos cambios de código.
- Columnas flexibles que pueden reorganizarse fácilmente arrastrando y soltando.
- Otras funciones, como las opciones de alineación y la compatibilidad con diseños complejos, hacen de Flexbox una potente herramienta para diseñadores y desarrolladores

¿Por qué usar flexbox?

Hay muchas razones por las que los diseñadores y desarrolladores web pueden optar por utilizar Flexbox, entre ellas:

- Sus características de diseño flexible y receptivo lo hacen ideal para crear sitios web

adaptados a dispositivos móviles.

- Puede utilizarse con otros marcos CSS, como Media Queries, para crear diseños de varias capas que respondan a diferentes tamaños de pantalla.
- Como otros marcos CSS, Flexbox es de código abierto y de uso gratuito, lo que lo convierte en una opción atractiva para muchos diseñadores y desarrolladores web.

1.1 Práctica con Flexbox

Para poder probar todas las posibilidades que ofrece Flexbox CSS, vamos a:

a) Crear una capa **class="contenedor",** que hará de padre, y siete capas **class="elemento"** numeradas.

HTML:

```
<div class="contenedor">
   <div class="elemento">1</div>
   <div class="elemento">2</div>
   <div class="elemento">3</div>
   <div class="elemento">4</div>
   <div class="elemento">5</div>
   <div class="elemento">6</div>
   <div class="elemento">7</div>
</div>
```

b) Vamos a colocar de inicio un tamaño de anchura del 25% para los elementos en relación al contenedor padre. Para comenzar a utilizar Flexbox añadimos al contenedor la propiedad **"display:flex"**

CSS EJERCICIO 1:

```
.contenedor{
    display:flex;
}
.elemento{
    width:25%;
}
```

El comportamiento de dirección y tamaño que tendrán los elementos de nuestro contenedor

se adaptan a su padre, aunque hayamos definido una anchura de elementos del 25% éstos ocuparan el 100% de anchura del contenedor entre la suma de todos. Por defecto, tiene ese comportamiento "flexible" como indica su nombre.

c) Vamos a ver la propiedad "**flex-direction**", que puede tomar 4 valores y se aplica al padre (contenedor):

- **flex-direction:row;** -> Los elementos se visualizan **de izquierda a derecha** (valor por defecto, similar al ejemplo 1)
- **flex-direction:row-reverse;** -> Los elementos se visualizan **de derecha a izquierda**.
- **flex-direction:column;** -> Los elementos se visualizan **de arriba hacia abajo**.
- **flex-direction:column-reverse;** -> Los elementos se visualizan **de abajo hacia arriba**.

CSS EJERCICIO 2

```
.contenedor{
  display:flex;
  flex-direction:row-reverse;
}
```

Los elementos invierten su orden de visualización de derecha a izquierda, sin tener en cuenta el orden de la maquetación.

d) Vamos a asignar a la propiedad "**flex-direction**" el valor column.

CSS EJERCICIO 3:

```
.contenedor{
    display:flex;
    flex-direction:column;
}
```

Se visualizan formando una columna de arriba hacia abajo. En este caso, como los elementos miden el 25% del contenedor, su anchura sí que toma el valor indicado. Si eliminamos la anchura, las capas se ajustarán al tamaño del contenedor, ocupando el 100% de anchura.

1.2 Otros atributos de Flexbox

Flex-wrap:

- **flex-wrap:nowrap ->** Los elementos se muestran en línea, en una sola fila, y su tamaño se ajusta al contenedor siempre y cuando la suma de todos ellos sea mayor o igual que el 100% de la anchura del contenedor. Si es inferior, se siguen mostrando en línea pero conservan su tamaño. Este es el valor por defecto, y como veíamos en el *ejemplo 1*, aunque la anchura de los elementos es el 25% del contenedor, todos se muestran en línea modificando su tamaño para que la suma total no sea superior al 100% de su contenedor.
- **flex-wrap:wrap ->** Los elementos se muestran en línea, pero si su anchura supera la del contenedor, se distribuyen en varias filas.
- **flex-wrap:wrap-reverse ->** Los elementos se muestran en línea, pero si su anchura supera la del contenedor, se distribuyen en varias filas, y además lo hacen en orden inverso al de maquetación.

Justify-content: En cuanto a la **alineación horizontal** de los elementos en Flexbox, encontramos la propiedad "**justify-content**", que alinea los elementos a lo largo del eje principal (main axis) de su contenedor, pero a diferencia de la alineación de un texto, en Flexbox hay que tener en cuenta también la dirección de los elementos. Sus valores más utilizados:

- **justify-content:flex-start;** -> Alinea los elementos en horizontal desde el inicio de la dirección del eje principal de su contenedor (partiendo desde el inicio de la línea). Este es el valor por defecto. Es importante destacar que, como veremos más adelante, **la dirección establecida en "flex-direction" afecta a la alineación**.
- **justify-content:flex-end;** -> Alinea los elementos en horizontal desde el final de la dirección del eje principal de su contenedor (partiendo desde el final de la línea)
- **justify-content:center;** -> Alinea los elementos al centro del eje principal de su contenedor. Similar a un texto alineado al centro.
- **justify-content:space-between;** -> Alinea los elementos justificándolos a lo largo del eje principal de su contenedor. Similar a un texto justificado. Los elementos laterales se pegan a los extremos y el resto se distribuyen a lo largo del eje principal dejando el mismo espacio entre ellos.
- **justify-content:space-around;** -> Alinea los elementos distribuyendo sus centros de forma horizontal a lo largo del eje principal de su contenedor, dejando el mismo espacio lateral de separación al comienzo, al final y entre ellos.

align-items | align-self | align-content: alineación vertical, se realiza a través del llamado **"eje transversal" (cross axis)**, y para ello contamos con tres propiedades diferentes, "*align-items*", "*align-self*" y "*align-content*". El W3C indica sobre estas propiedades:

- **align-items** establece la alineación predeterminada para todos los elementos del contenedor, incluidos los elementos independientes.
- **align-self** permite alinear elementos independientes del contenedor.
- **align-content** alinea las líneas/filas de elementos de un contenedor.

En el enlace https://www.w3.org/TR/css-flexbox-1/ podemos visualizar otros atributos con respectivos ejemplos de uso.

2. Grid Css

El grid es una cuadrícula sobre la que se distribuyen los distintos elementos que componen la web.

Phil Cupp fue el pionero de este trabajo, él presentó la especificación que dio paso a CSS Grid. Sin embargo, no fue hasta 2012 cuando en Microsoft, por primera vez se implementó un módulo como el que hoy tenemos para manejo de filas y columnas en CSS Grid

CSS grid layout o CSS grid es una técnica de las Hojas de Estilo en Cascada que permite a los desarrolladores web crear diseños complejos y adaptables con mayor facilidad en todos los navegadores

La propiedad CSS grid es un shorthand que permite definir todas las propiedades *grid* explícitas (grid-template-rows, grid-template-columns, y grid-template-areas), implícitas (grid-auto-rows, grid-auto-columns, y grid-auto-flow (en-US)), y relativas a *gutter* (grid-column-gap y grid-row-gap (en-US)) en una sola declaración.

2.1 Práctica con Grid

HTML:

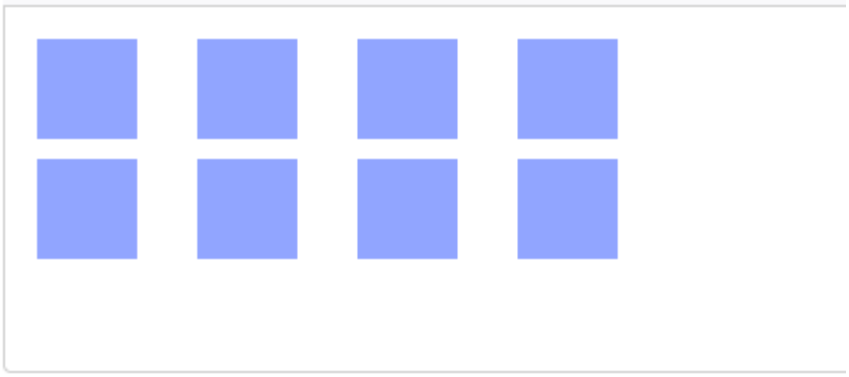
```
<div id="container">
   <div></div>
   <div></div>
   <div></div>
   <div></div>
   <div></div>
   <div></div>
   <div></div>
   <div></div>
</div>
```

b) Vamos a usar grid

CSS EJERCICIO 4:

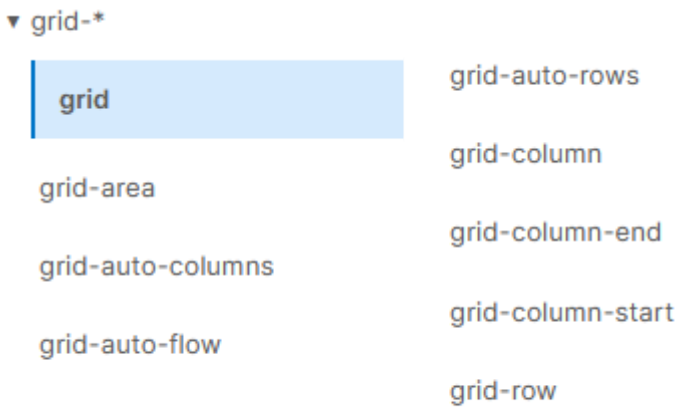
```
#container{
        display: grid;
        grid: repeat(2, 60px) / auto-flow 80px;
  }

  #container > div {
        background-color: #8ca0ff;
        width: 50px;
        height: 50px;
  }
```

Analicemos el resultado:



2.2 Otros atributos de Grid

En el enlace https://developer.mozilla.org/en-US/docs/Web/CSS/grid podemos consultar otras propiedades con su definición, sintaxis y ejemplos. Entre ellos:



No olvidemos chequear la compatibilidad con el navegador:

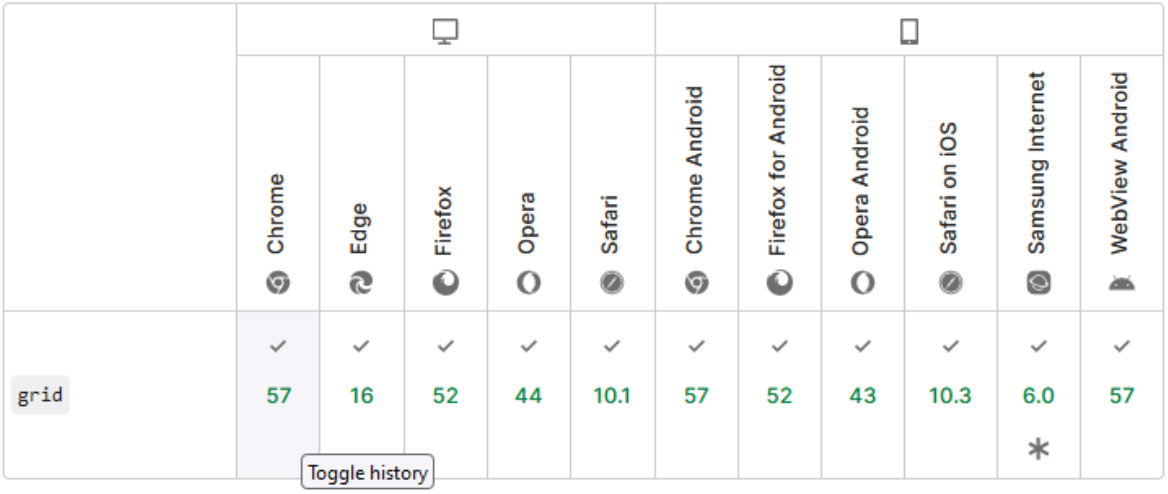


Figura 3: Compatibilidad Grid con navegadores.

3. Comparativo entre Flexbox y Grid

Es importante revisar las fortalezas de Grid y flexbox, pero particularmente, la diferencia básica entre CSS Grid Layout y CSS Flexbox Layout es que Flexbox se creó para diseños de una dimensión, en una fila o una columna. En cambio CSS Grid Layout se pensó para el diseño bidimensional, en varias filas y columnas al mismo tiempo. Grid es una opción ideal para diseños complejos, mientras que Flexbox es más adecuado para diseños simples y adaptables.

4. Media Queries

Las Media Queries se utilizan para obtener las características técnicas del dispositivo que solicita la visualización de la página web. Las obtiene solicitando al dispositivo las llamadas Media Features. De esta manera, es posible definir reglas de estilo para diferentes dimensiones de pantalla.

En desarrollo web, las media queries son un módulo CSS3 que permite adaptar la representación del contenido a características del dispositivo como la resolución de pantalla o la presencia de características de accesibilidad como el braille. Las media queries actúan como contenedor de las reglas y selectores a aplicar, por lo tanto, todo lo que engloba empieza con la apertura de una llave y termina con el cierre de la misma.

En cuanto a la sintaxis, una media query siempre comienza con el indicador de @media, seguido de un **mediatype**. Este mediatype es el encargado de seleccionar qué tipo de formato será objeto

de estas reglas.

El segundo parámetro de una media query en CSS es el media feature, donde indicamos al navegador qué **condición debe cumplir el dispositivo de salida** especificado anteriormente para que la condición sea verdadera y se aplique este código contenido en nuestra media query.

```
@media not|only mediatype and|not|only (media feature){
    . my-code { … }
}
```

Los media features más comunes son los referidos a las **dimensiones de pantalla del dispositivo**, pudiendo establecer el alto y ancho en el que se aplicarán (con height y width), o lo que es más interesante, a partir de que ancho o alto se aplicarán (con min/max-width y min/max-height)

Aunque también las hay más complejas, como por ejemplo aquellas combinadas que la orientación del dispositivo (landscape o portrait) o incluso las que definen el ratio de píxeles de la pantalla, como device-*pixel*-ratio, muy útil para definir estilos y CSS para las pantallas retinas, por ejemplo algunos iPads o iPhones.

El navegador **aplicará el código que esté incluido en una media query sobrescribiendo el heredado**. Con esto es como si estableciéramos una capa sobre otra, con cierta transparencia, ya que no eliminamos del todo el código general de la hoja de estilos CSS

4.1.1 Consulta de medios

Una consulta de medios es un componente de CSS3 que hace que una página web adapte su diseño a diferentes tamaños de pantalla y tipos de medios.

```
@media media type and (condition: breakpoint) {
                    // reglas CSS
}
```

Podemos establecer diferentes tipos de medios bajo una variedad de condiciones. Si la condición y/o el tipo de medio se cumplen, la regla dentro de la consulta de medios se aplicará. Comenzamos definiendo la consulta de medios con la regla @media y después incluimos las reglas CSS entre los paréntesis. La regla @media también se usa para especificar los tipos de medios.

Dentro de los paréntesis establecemos una condición. Por ejemplo, quiero aplicar un tamaño de fuente mayor en los teléfonos móviles. Para hacer esto, necesitamos establecer un ancho máximo que compruebe el ancho de un dispositivo:

```
.texto {
  font-size: 14px;
}

@media (max-width: 480px) {
  .texto {
    font-size: 16px;
  }
}
```

El tamaño del texto es de 14px. Sin embargo, como hemos aplicado una consulta de medios, el tamaño del texto será de 16px cuando el ancho máximo del dispositivo sea de 480px o menos.

Importante: añade tus consultas de medios siempre al final de tu documento CSS.

4.1.2 Tipo de medios

Si no establecemos ningún tipo de medio, la regla @media selecciona todos los dispositivos por defecto. Los tipos de medios se aplican justo a continuación de la regla @media. Existen muchos tipos de dispositivos, pero podemos agruparlos en 4 categorías:

- all — para todo los tipos de medios
- print — para impresoras
- screen — para pantallas de ordenador, tablets y smartphones
- speech — para lectores de pantalla que "leen" la página en voz alta

Por ejemplo, si quiero seleccionar solo pantallas, usaré la palabra screen justo después de la regla @media. También tengo que unir ambas reglas con la palabra "and":

```
@media screen and (max-width: 480px) {
  .texto {
    font-size: 16px;
  }
}
```

4.1.3 Breakpoints

El breakpoint es quizás uno de los conceptos que escuches y uses más a menudo. Un breakpoint es esencial a la hora de determinar cuándo cambiar el diseño y adaptar las reglas dentro de las consultas de medios. Volvamos a nuestro ejemplo del principio:

```
@media (max-width: 480px) {
  .texto {
    font-size: 16px;
  }
}
```

Aquí el breakpoint es 480px. La consulta de medios sabe cuándo establecer o sobreescribir la nueva clase. Básicamente, si el ancho de un dispositivo es menos de 480px, la clase texto se aplicará, si no, no. Existen una gran cantidad de dispositivos en el mercado, con lo que no podemos ni debemos definir breakpoints fijos para cada uno de ellos.

Por ello, no podemos decir que existe una resolución estándar para todos los dispositivos, sino breakpoints de uso común en el día a día de la programación. Los breakpoints comunes según el ancho de los dispositivos:

- 320px — 480px: Dispositivos móviles
- 481px — 768px: iPads, tablets
- 769px — 1024px: pantallas pequeñas, ordenadores portátiles
- 1025px — 1200px: pantallas grandes, ordenadores de escritorio
- 1201px y más — pantallas extra-grandes, televisores

Como se mencionó anteriormente, estos breakpoints son flexibles, ya que no existe una definición estándar, sino algunos de uso común.

4.1 Práctica con Media Queries

HTML:

```
<h1>Ejemplo de Media Query #1</h1>
<p>Reduce el ancho de la ventana a 600 px o menos.</p>
<p>Observe cómo el color de fondo cambió de blanco a azul.</p>
```

CSS EJERCICIO 5:

a) El color cambiará a azul para dispositivos más pequeños

```
body {
  text-align: center;
}

p {
  font-size: 1.5rem;
}

@media (max-width: 600px) {
  body {
    background-color: #87ceeb;
  }
}
```

CSS EJERCICIO 6:

b) color de fondo de azul a rojo si el dispositivo tiene un ancho entre 600 y 768 px. Podemos usar el operador and para lograr esto.

```
body {
  text-align: center;
  background-color: #87ceeb;
}

p {
  font-size: 1.5rem;
}

@media (min-width: 600px) and (max-width: 768px) {
  body {
    background-color: #de3163;
  }
}
```

4.2 Otros atributos

Más detalles sobre el modo de uso en https://developer.mozilla.org/es/docs/Web/CSS/CSS_media_queries/Using_media_queries .

5. Comparativo entre Flexbox y Media Query.

Flexbox ofrece potentes funciones como la fácil reorganización de columnas y opciones de alineación que lo hacen ideal para el diseño responsivo y los sitios web adaptados a dispositivos móviles. Mientras Media Queries es una herramienta más precisa que puede utilizarse para crear diseños personalizados para distintos tamaños y orientaciones de pantalla. Sin embargo, qué framework es mejor depende en última instancia de sus necesidades y requisitos específicos, por lo que es importante considerar los pros y los contras de cada framework antes de tomar una decisión.

Por ejemplo, si necesita un control preciso sobre cómo se muestran los elementos en diferentes tamaños de pantalla, entonces Media Queries puede ser una mejor opción, mientras que si necesita formas más flexibles de organizar elementos o desea crear rápidamente un sitio web adaptable y apto para móviles, entonces Flexbox puede ser la mejor opción.

6. Multimedia flexible: imágenes adaptables.

Media query no solo permite la adaptación de elementos de texto, sino también switchear imágenes de diversos formatos y tamaños al hacer response.

a) Realice un print screen, guardarla con formato bmp, imagen1.bmp.,

b) Use cualquier editor de imagen para cambiar el tamaño y el formato, salvar como imagen2.gif.

c) Crear el html

HTML:

```
<h1>Probando picture</h1>
    <picture>
          <source media="(min-width:650px)"
               srcset="imagen1.bmp">
          <source media="(min-width:465px)"
              srcset="imagen2.gif">
 </picture>
```

d) Usando el browser abra el html y cambie el tamaño de la ventana y hágala más pequeñas

e) Chequee usando botón derecho el nombre de la imagen.

f) Haga doble click en el borde superior del browser y vuelva a revisar el

nombre de la imagen.

g) ¿cuáles son sus conclusiones?.

7. Fuentes tipográficas flexibles

Recordemos según lo visto en teoría sobre las unidades de medidas relativas:

Relativo Tamaño Letra	Relativo x minúscula	Relativo Herencia (contenedor padre)	Relativo Resolución Pantalla (pixel)
em	ex	%	px
`div { font-size:2.65em; }`	`div { font-size:2.65ex; }`	`div { width:50%; }`	`div { font-size:4px; }`

Las unidades em y ex en la propiedad font-size son relativas al tamaño de letra del elemento padre (al contrario que todas las demás propiedades, en las que estas unidades son relativas al tamaño de letra del elemento). Esto quiere decir que las unidades em y los porcentajes se comportan de igual forma cuando hablamos de font-size. Si estás haciendo un estilo que pretendes utilizar en muchos elementos distintos, por ejemplo una class de CSS que pretendes usar en varias partes, es mejor usar rem. Si lo que quieres es que el tamaño sea variable en función de dónde estés usando el estilo, usa em. Tiene como referencia la medida de font-size del navegador, que casi siempre es de 16px, si hemos definido nosotros otra, pues será esa otra medida. Es decir, que 1rem será igual a 16px. Em: En cambio, em hace referencia al font-size del elemento actual.

- **HTML:**

```
<h1>Probando tamaños de fuentes en tag p</h1>
      <p class="p1">Soy rem</p>
      <p class="p2">Soy em</p>
      <p class="p3">Soy px </p>
      <p class="p4">Soy porcentaje</p>
      <p class="p5">Soy px </p>
```

- **CSS EJERCICIO 8:**

a) Configuremos en el html clases con valores para el atributo font-size de diversos tipos .

```
body {
  text-align: center;
  background-color: #87ceeb;
}

p1 {
  font-size: 1.5rem;
}

p2 {
  font-size: 2.65em;
}

p3 {
  font-size: 2.65ex;
}

p4 {
  font-size: 25%;
}

p5 {
  font-size: 4px;
}
```

Identifique las diferencias entre estas definiciones. En la práctica podrían parecer equivalente pero:

- EM es relativo al tamaño de fuente del elemento padre, por lo que si deseas escalar el tamaño del elemento en función del tamaño de su padre utiliza EM.
- REM es relativo al tamaño de fuente de la raíz (HTML) y si este no está establecido se usará el que tenga configurado el navegador, por lo que si deseas escalar el tamaño del

elemento en función del tamaño de la raíz, sin importar cuál sea el tamaño principal, usa REM. Si has utilizado EM y estás encontrando problemas de tamaño debido a muchos elementos anidados, REM probablemente sea la mejor opción.

- % es un porcentaje del ancho o alto del elemento padre. Las unidades de porcentaje suelen ser útiles para establecer el ancho de los márgenes.

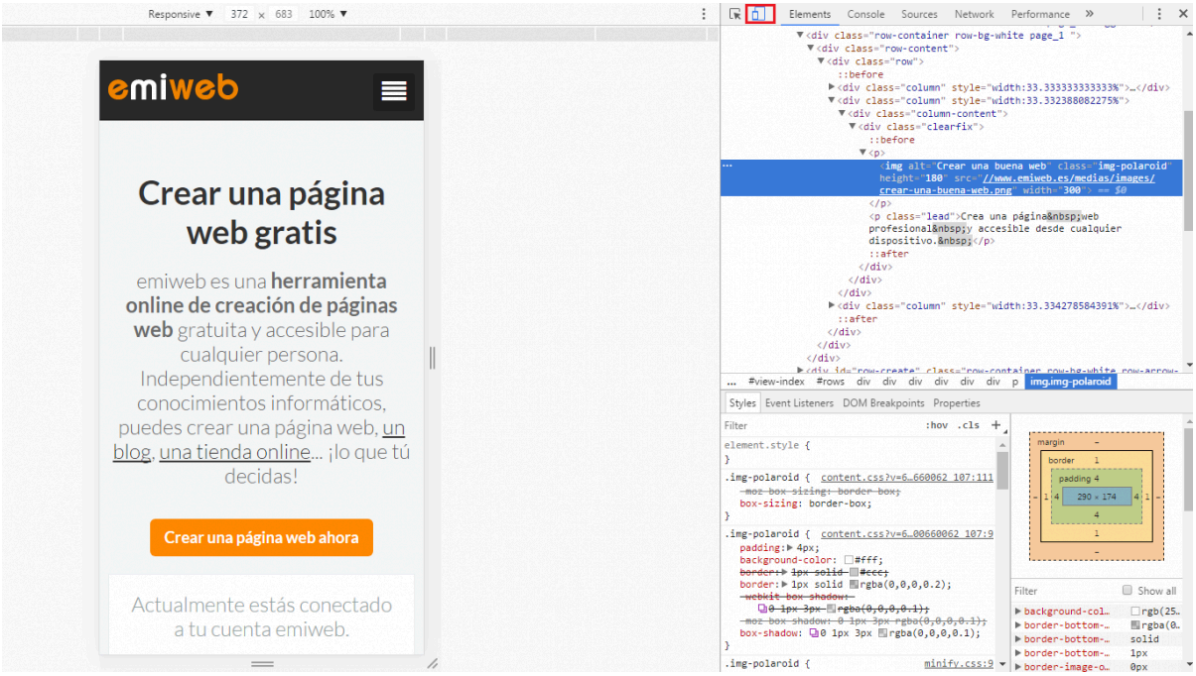
8. Herramienta para desarrolladores

Ver desde el inspector para pantalla desde el browser es posible, según el navegador se puede activar de maneras similares.

8.1 Google Chrome

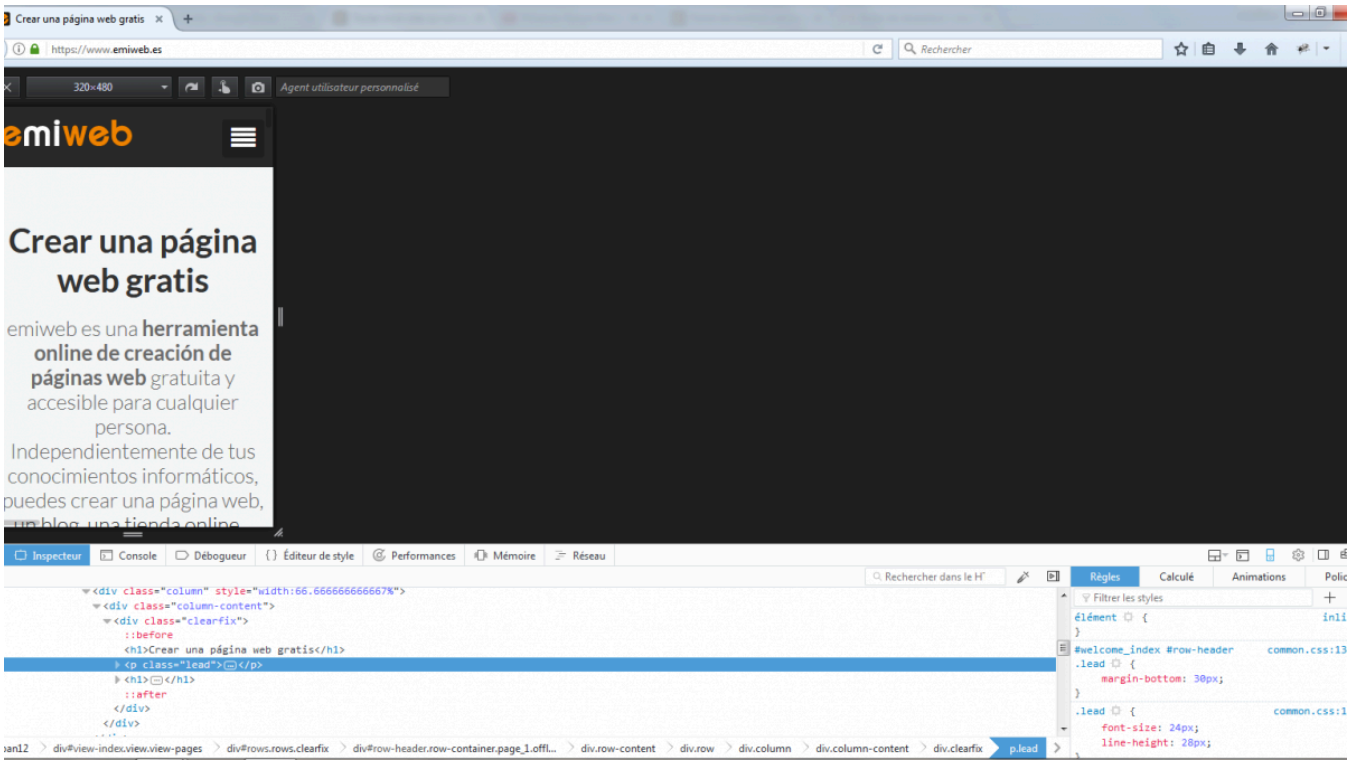
- Dirígete a tu sitio
- Oprime la tecla F12
- Aparecerá una ventana a la derecha o en la parte inferior de tu pantalla. En la parte superior a la izquierda de esta ventana verás dos iconos pequeños.
- Da click en el segundo icono que tiene forma de un smartphone o tablet.
- Puedes elegir el tamaño de la pantalla en la barra horizontal de la parte superior. Esta barra te permitirá probar en diversas resoluciones, como el Nexus 5X, iPhone 6 o iPad Pro.

  También existe la opción de descargar extensiones (webmobilefirst.com), diseñadas para dispositivos específicos (e.g. Galaxy S22, iPhone 15 , dispositivos especiales como el Apple Watch, Algunos dispositivos solo están disponibles en la versión premium).



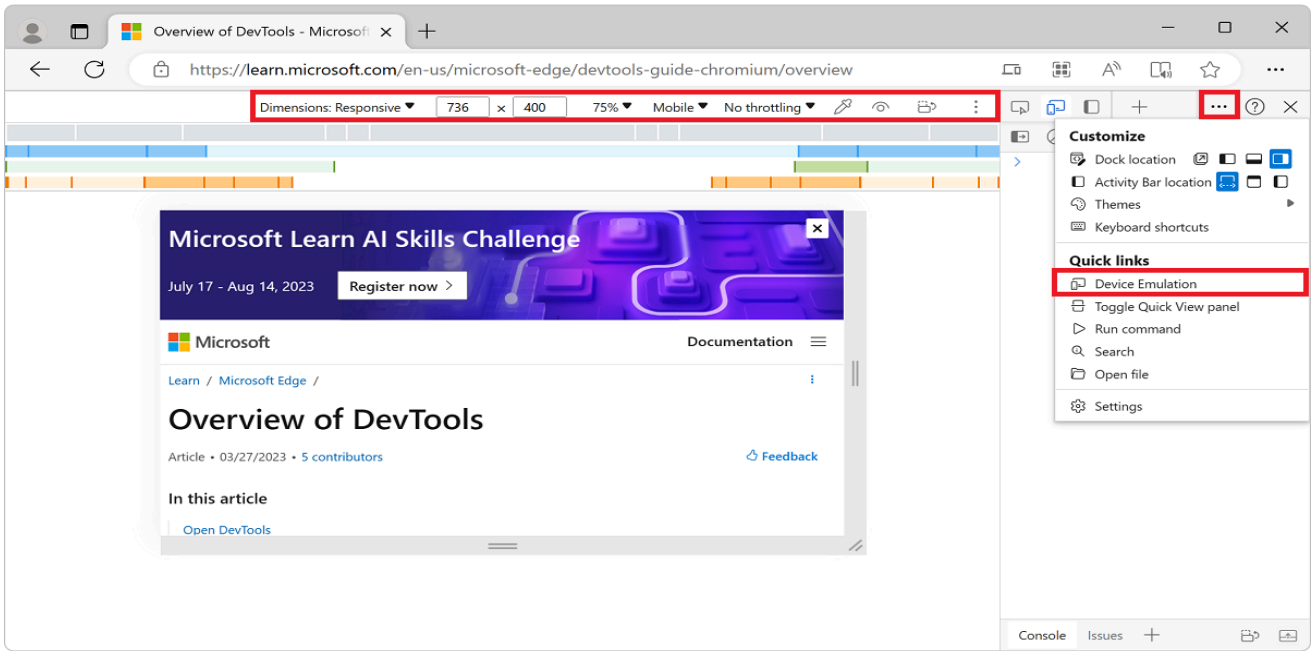
8.2 Mozilla Firefox

- Dirígete a tu sitio
- Oprime la tecla F12
- Aparecerá una ventana a la derecha de tu pantalla. En la parte superior a la derecha de esta ventana verás seis iconos pequeños.
- Da click en el tercer icono que representa un smartphone.
- Puedes elegir el tamaño de la pantalla en la barra horizontal de la parte superior.
- Al contrario que Chrome, Firefox no ofrece la opción de los nombres de smartphones o tablets, pero sí resoluciones equivalentes.



8.3 Microsoft Edge

- Abra DevTools. Por ejemplo, haga clic con el botón derecho en una página web y, a continuación, seleccione **Inspeccionar**.
- En la **barra de actividad**, haga clic en el botón **Alternar emulación de dispositivo** (). O bien, en DevTools, seleccione Personalizar y controlar La **emulación de dispositivode DevTools** (...): >



La página web se representa en el panel Emulación de dispositivo. La barra de herramientas del dispositivo se abre en modo de ventanilla dinámica.

9. Visual Code y CSS

Al igual que con html, Visual Code como entorno de trabajo tiene soporte para la actualización de CSS. Veamos un ejemplo del https://www.youtube.com/watch?v=rkNxOgH29rE. Más información en: https://code.visualstudio.com/docs/languages/html

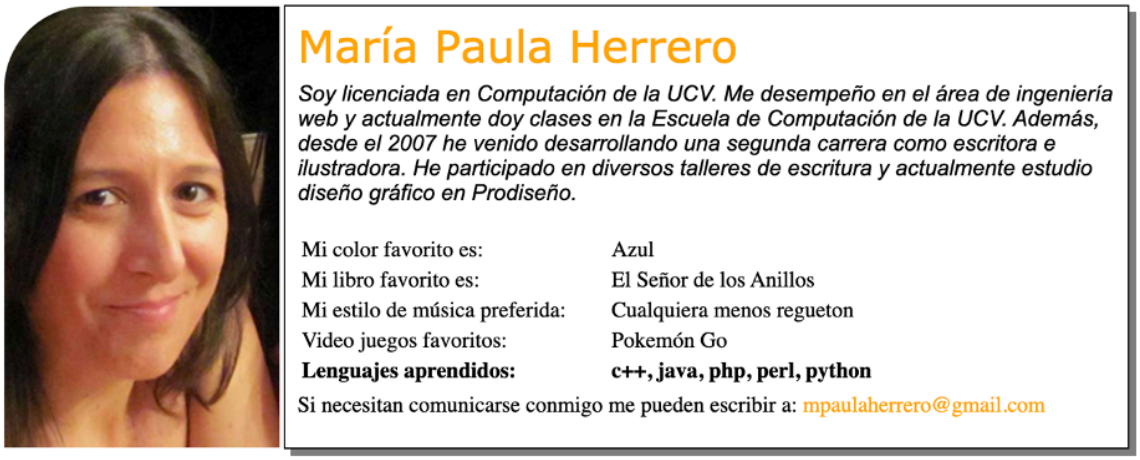
10. Ejercicio

Reto 02 (At home)

Antes de empezar, cree una rama nueva con el nombre “reto-2” a partir de lo que usted realizó en el reto anterior. Trabaje este nuevo reto sobre esa nueva rama.

a) Crear CSS para páginas web creadas en el laboratorio previo.

1. Para el estilo tome en cuenta lo siguiente:
   - Vamos a chequear que no exista estilo incrustado en el HTML y todos se encuentran en un archivo aparte llamado style.css. La página de perfil se debe ver como la imagen que conocemos del reto anterior:



   - Asegurarse de usar los selectores correctos: selectores de etiqueta, creando un class o creando un id. El manejo de id es importante para la automatización de pruebas.
2. Haga commit y suba a GitHub todo lo que ha hecho hasta ahora con un mensaje que diga “CCS en file externo chequeado”
3. Este archivo debe ser compartido tanto por perfil.html como por index.html. Al terminar haga commit del proyecto con el mensaje “hojas de estilo compartidas”
4. Actualice el CSS configurando las características de las columnas y filas de la página perfil usando flexbox y grid.
5. Chequee el llamado desde los html a las nuevas referencias, verifique que todo lo asociado al diseño de las páginas se está manejando desde el archivo CSS.
6. Actualice el CSS configurando las características de cuatro tipos de pantallas usando media con breakpoint:
   - 320px — 480px: Dispositivos móviles
   - 481px — 768px: iPads, tablets
   - 769px — 1024px: pantallas pequeñas, ordenadores portátiles
   - 1025px — 1200px: pantallas grandes, ordenadores de escritorio
7. Renombre la imagen de la foto de perfil, a {CEDULA}Grande.jpg (ej: 29765263Grande.jpg), realice una copia con el nombre {CEDULA}Pequena.jpg y actualice la imagen reduciendo su tamaño.
8. Establezca en el CCS, usando media query, que la aplicación al realizar response use la imagen pequeña para teléfonos móviles y tablets, mientras que en la versión para ordenadores se cambie a la imagen grande.
9. Incorpore el atributo font-size a las etiquetas usadas en el html para definir las unidades de medida de las fuentes tipográficas de forma flexible, en base al análisis y conclusiones del ejercicio 8.
10. Realice el llamado desde los html a la nueva referencias a estos estilos para soportar

responsive. Ejecutar estas pruebas usando lo aprendido para impersonar los estilos de pantalla desde el browser de su preferencia.

11. Tome un capture por cada tipo de pantalla ubicado de tal forma que sea visible el nombre de la imagen que usa la página en cada tamaño de pantalla.
12. También ejecute los pasos desde el 4 al 11 para el listado y cada una de las páginas html de su rama git.
13. Al tener todo listo, haga commit del proyecto con el mensaje “versión reponsive ”
14. Coloque el url de GIT en el aula virtual como entregable del reto junto a las imágenes de las pruebas del paso 11.

¡Exitos!