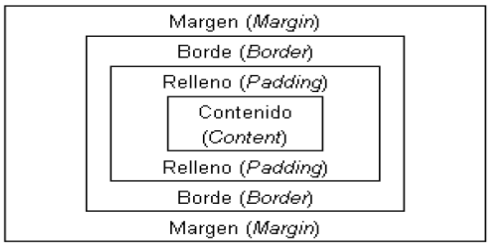
Retomando lo que vimos en el documento anterior, CSS (Cascading Style Sheets) es un lenguaje que describe el estilo de un documento HTML, es decir, describe cómo será mostrado.
CSS3 es una tecnología que ha tenido una evolución en el tiempo, que actualmente se encuentra en su versión3, como su propio nombre indica.
En este tramo, vamos a conocer otros conceptos que nos permitirán ir familiarizándonos con lo que es el diseño. Para ello comenzamos recuperando algunos conceptos y entendiendo lo siguiente:

Modelo de Caja

Es el comportamiento que hace que todos los elementos de un documento HTML se representen mediante cajas rectangulares. De este modo, permite asignarles propiedades a los elementos, y así afectar el alto, el ancho, el margen, etc.
Este modelo condiciona el diseño de todas las páginas web. Las propiedades de modelo de caja solo aplican a etiquetas de bloque

El Modelo de Caja de CSS

- Los **div** en HTML son cajas.
- CSS nos permite cambiar el alto, ancho, posición, márgenes, etc de las cajas..
- Importante: los "div" por defecto se irán **posicionando** hacia abajo del sitio web, independiente del ancho que nosotros le coloquemos.



ELEMENTOS DE BLOQUE

Las etiquetas de bloque intentan ocupar el 100% del ancho del sitio. Visualmente generan un salto de línea. Esto se da porque, al ocupar todo el ancho disponible, no dejan espacio para que entre otro elemento. Las etiquetas son un ejemplo de etiquetas de bloque muy utilizadas ya que permiten generar divisiones en nuestro sitio.

```
<div> Mi Texto </div>
<div>Mi otro Texto </div>
div {
        background-color: green;
}
```

ELEMENTOS EN LÍNEA

Las etiquetas en línea ocupan sólo el ancho de su contenido y no cambian la distribución del sitio. Es decir, no van a generar saltos de línea por defecto, ya que su ancho va a estar determinado por el contenido que lleve dentro.

```
<span> elemento 1 en linea</span>

<span>elemento 2 en linea</span>

span {

    background-color: red;

}
```

TIPOS DE ELEMENTOS

inline

Define un elemento con comportamiento en línea, y no recibe algunas propiedades del modelo de caja.

block

Define un elemento con comportamiento de bloque, y puede recibir propiedades del modelo de caja.

inline-block

Define un elemento con comportamiento de semi-bloque. Puede recibir propiedades del modelo de caja, y también comparte propiedades de elementos de línea.

none

Oculta a un elemento en la visual. No lo elimina de la estructura de HTML, sólo desaparece de la vista.

Mediante la propiedad **display** de css podemos cambiar la disposición del elemento que queramos. Los valores que recibe son block, inline, inline-block y none

```
<span>Elemento span</span>

span {
  background-color: green;
  display: block;
}
```

PROPIEDAD WIDTH

Si un elemento no tiene declarada la propiedad width, el ancho será igual al 100% de su padre contenedor, siempre y cuando ése sea un elemento de bloque.

Para asignarle valor a esta propiedad, lo podemos hacer usando la medida de porcentajes (%) ó píxeles (px).

```
div {
  width: 120px;
}
```

PROPIEDAD HEIGHT

Si un elemento no tiene declarado la propiedad height, el alto será igual a la altura que le proporcione su contenido interno. Sea un elemento de bloque o de línea.

Para asignarle valor a esta propiedad, lo podemos hacer usando la medida píxeles (px).

```
div {
  height: 130px;
}
```

```
div {
  background-color: blue;
  width: 120px;
  height: 130px;
}
```

PROPIEDAD PADDING

Hace referencia al margen interior del elemento. Para asignarle valor a esta propiedad, lo podemos hacer usando la medida píxeles (px), indicando 1 valor para los 4 lados de la caja. También podemos hacerlo con 2 valores, el primero va a indicar el padding de arriba y abajo, y el segundo el de la izquierda y la derecha.

```
div {
  padding: 12px;
}
div {
  padding: 22px 30px;
}
Div{
  background-color: blue;
  width: 120px;
  height: 130px;
  padding: 12px;
}
```

PROPIEDAD BORDER

Hace referencia al borde del elemento. Para asignarle valor a esta propiedad, lo hacemos definiendo el estilo de línea, su tamaño y su color. El estilo de línea puede ser solid, dotted, dashed o double.

```
div {
  border: solid 3px green;
}
div {
  background-color: blue;
  width: 120px;
```

```
  height: 130px;
  padding: 12px;
  border: solid 3px green;
}
```

PROPIEDAD MARGIN

Hace referencia al margen del elemento. Sirve para separar una caja de la otra. Para asignarle valor a esta propiedad, lo podemos hacer usando la medida píxeles (px), indicando 1 valor para los 4 lados de la caja. También podemos hacerlo con 2 valores, el primero va a indicar el padding de arriba y abajo, y el segundo el de la izquierda y la derecha.

```
div {
  margin: 30px;
}
div {
  background-color: blue;
  width: 150px;
  height: 130px;
  padding: 12px;
  border: dotted 3px green;
  margin: 30px;
}
```

El ancho total de la caja es igual a la suma de su width, padding y border. Además, cuenta con 60px de margin

PROPIEDAD BOX-SIZING

Esta propiedad permite que el modelo de caja sea más fácil de usar, porque descuenta automáticamente del ancho y alto lo que agregamos en relleno y borde. El único valor que se sigue sumando es el del margin.

```
div {
  background-color: blue;
  width: 150px;
  height: 130px;
  padding: 12px;
  border: solid 3px green;
  margin: 30px;
  box-sizing: border-box;
}
```

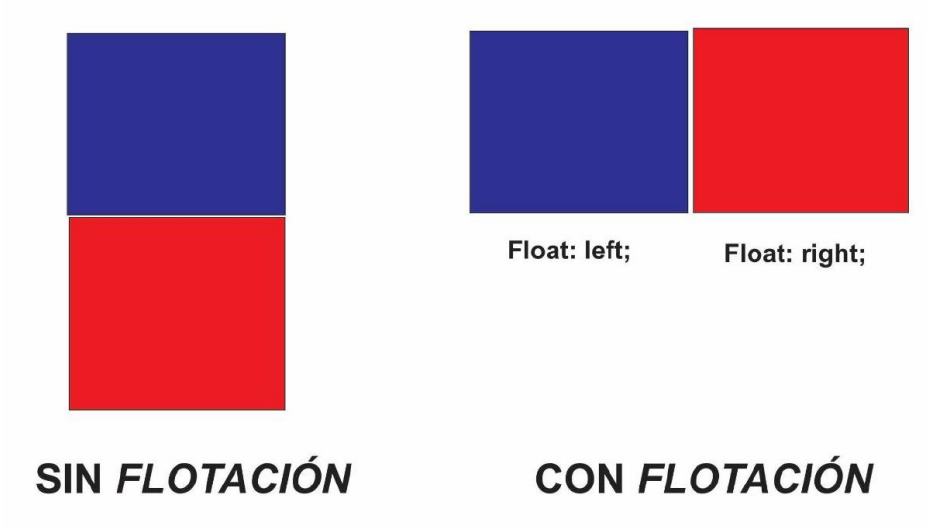
En este caso el bloque va a tener un ancho total de 150px incluyendo su border, padding y contenido. Va a tener 6px de border, 24px de padding y automáticamente su contenido va a tomar 120px de ancho haciendo un total de 150px.

Flotación

Los sitios web tienen un flujo natural de disposición. Por defecto, los elementos se van posicionando uno por sobre el otro, como si fueran cajas.

Mediante la propiedad **float**, le estamos otorgando a un elemento la habilidad de flotar. Es decir, salirse de ese flujo natural del sitio, y decidir hacia dónde queremos que ese elemento flote.

Los valores que recibe son: left, right, none y inherit.



CÓMO FLOTAR

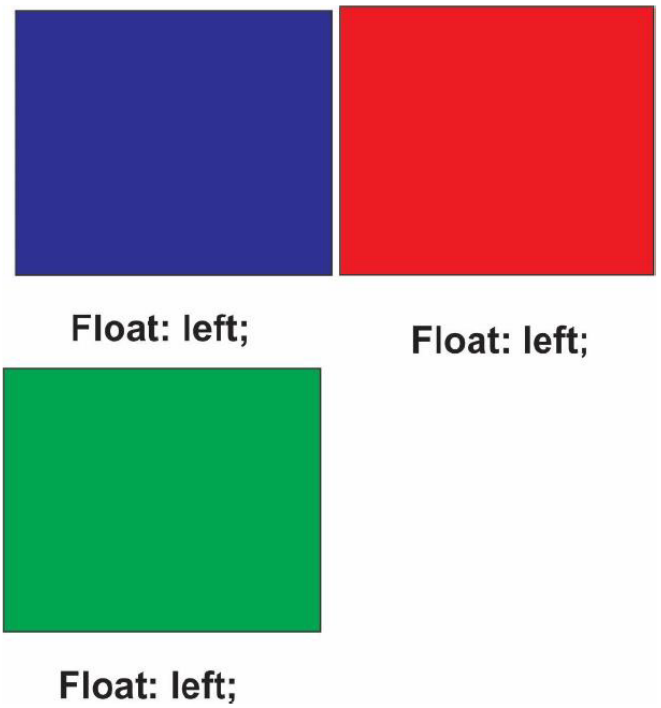
Es importante determinar un ancho para nuestras cajas. De esta forma, vamos a poder controlar cuántas de ellas entran en una misma línea.

Si la suma de los anchos de las cajas supera el ancho del contenedor padre, éstas no van a entrar en la misma línea, sin importar que tengan asignada la propiedad de float

CÓMO SE COMPORTAN

Por más de que floten, si los anchos de las cajas superan el ancho del contenedor, bajarán un renglón. Si cuidamos los tamaños podemos tener tantas cajas en una línea como deseemos.

Es clave usar anchos en porcentajes para poder hacer cálculos sencillos.



Los elementos por debajo del elemento al que le asignamos la flotación asumen que éste no existe más y ocupan el lugar vacío que el elemento flotante dejó.

No sucede lo mismo si los elementos que le siguen son de texto.

La forma más sencilla de "limpiar" flotaciones es usando la propiedad clear: both. Basta con aplicarle dicha propiedad al elemento que se encuentra por debajo de las cajas flotantes, para que éste conserve su posición.

CONTENEDOR PADRE

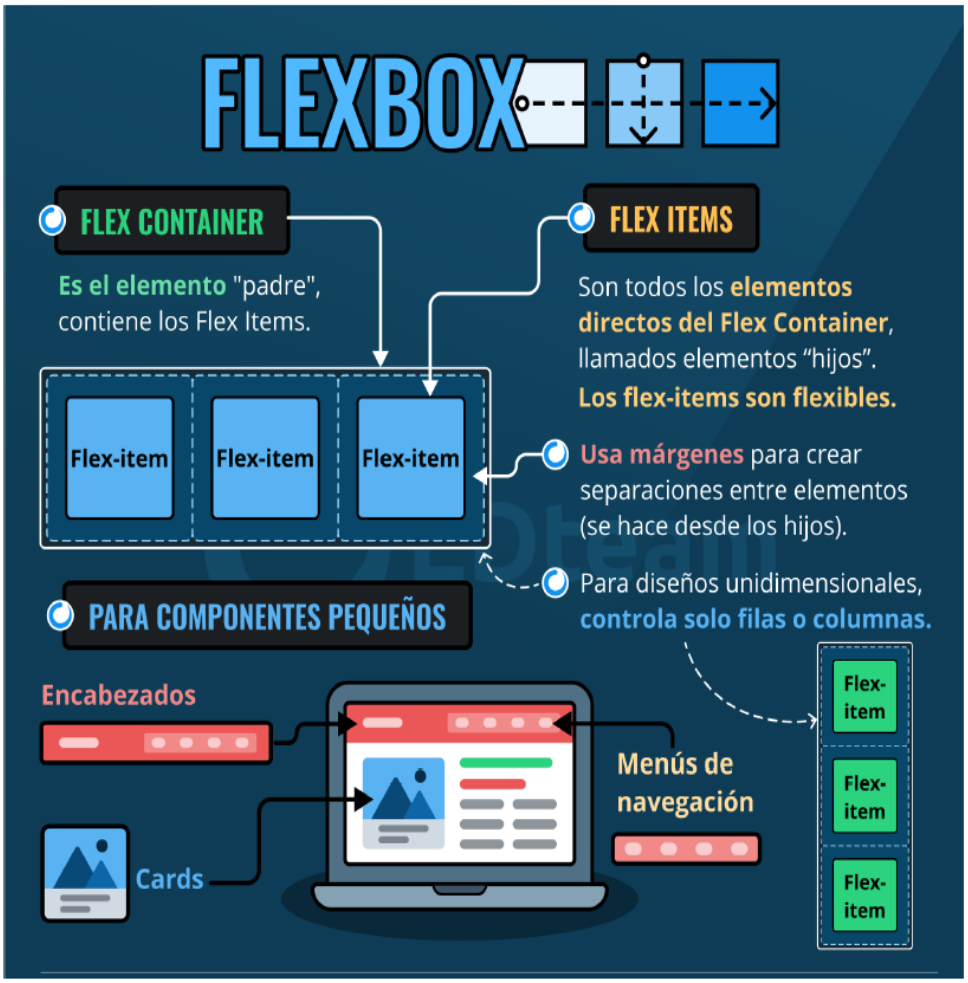
Al asignarle la propiedad float a un elemento, éste va a salir del flujo natural del sitio web. Por ende, saldrá también del flujo de su contenedor padre, quien intentará ocupar ese lugar “vacío”.

La forma de controlar este comportamiento es asignándole al padre la propiedad overflow: hidden.

Flexbox

Es una metodología de CSS que permite maquetar un sitio web utilizando una estructura de filas y columnas.

Cuando usamos flotación para posicionar un elemento en un sitio web, el mismo deja de formar parte del flujo natural de la estructura de elementos. Esto genera solapamiento de cajas y estructuras difíciles de mantener.



Estructura Básica de Flexbox

Esta metodología propone una estructura basada en el uso de un contenedor padre (Flex-container) y sus elementos hijos (Flex-items).

Para empezar a trabajar conflexbox tenemos que definir un flex-container. Para eso usamos la propiedad con el valor. De esta forma estamos habilitando un contextoflex para trabajar con los hijos directos del elemento.

```
.contenedor-padre {
  display: flex;
}
```

Estructura:

Cuando hablamos de unflex-container, hablamos de un elemento HTML que contiene a uno o más elementos. A estos elementos anidados, los llamamos flex-items.

Es en el flex-container en donde configuramos la mayoría de las propiedades flex.

Flex-Wrap

Por defecto, los elementos hijos de un contenedor flex, van a tratar de entrar todos en una misma línea. Para aclararle al contenedor que debe respetar el ancho definido de sus hijos, usamos la propiedad con el valor.

```
.contenedor-padre {
  display: flex;
  flex-wrap: wrap;
```

}

Un flex-item, a su vez, puede convertirse en un flex-container.

Para eso, sólo hace falta asignarle la regla display:flex , para que así sus elementos hijos pasen a ser flex-items.

Los ejes

Flexbox trabaja con dos ejes para desarrollar todo su flujo interno: el eje X y el eje Y.

Dentro de un flex-container, tanto el eje x como el eje y toman otros nombres. Cuando trabajamos en un flujo flex hablamos del main axis y el cross axis.

El concepto de trabajo de flexbox está basado en una sola dirección, es decir, los elementos se distribuyen o en filas horizontales o en columnas verticales.

Definiendo el eje principal de nuestro contenedor flex, estamos determinando el flujo que tendrán los elementos dentro del contenedor. Los mismos se ordenarán en base al eje que definamos.

flex-direction

Con esta propiedad definimos el main axis del contenedor (eje principal), que puede ser tanto horizontal como vertical. El crossaxis (eje transversal), será la dirección perpendicular al main axis.

flex-direction: row

Los ítems se disponen en el eje x, de izquierda a derecha.

Si no le aclaramos la propiedad flex-direction al contenedor, row es el valor por defecto

flex-direction: row-reverse

Los ítems se disponen en el eje x, de derecha a izquierda.

flex-direction: column

Los ítems se disponen en el eje y, de arriba hacia abajo.

flex-direction: column-reverse

Los ítems se disponen en el ejey, de abajo hacia arriba.

Flexbox nos da dos propiedades para alinear fácilmente los elementos tanto a través del main axis como del cross axis

justify-content

Con esta propiedad alineamos los ítems a lo largo del main axis. Si es horizontal, se alinearán en función de la fila. Si es vertical, se alinearán en función de la columna.

justify-content: flex-start

Los ítems se alinean respecto del inicio del main axis que hayamos definido. Si no le aclaramos el justify-content al contenedor, flex-start es el valor por defecto.

justify-content: flex-end

Los ítems se alinean respecto del final del main axis que hayamos definido.

justify-content: center

Los ítems se alinean en el centro del main axis.

justify-content: space-between

Los ítems se distribuyen de manera uniforme. El primer ítem será enviado al inicio del main axis, el último ítem será enviado al final del main axis.

justify-content: space-around

Los ítems se distribuyen de manera uniforme, con igual espacio alrededor de cada uno.

El primer ítem tendrá una unidad de espacio contra el borde del contenedor, y dos unidades de espacio contra el siguiente ítem, porque el mismo tiene su propio espacio que se aplica. Lo mismo sucede con el último ítem.

align-items

Con esta propiedad alineamos los ítems a lo largo del cross axis. Si no aclaramos esta propiedad, el valor por defecto es stretch.

align-items: stretch

Los ítems se ajustan para abarcar todo el contenedor. Si el cross axis es vertical, se ajustan en función de la columna. Si el cross axis es horizontal, se ajustan en función de la fila.

align-items: flex-start

Los ítems se alinean al inicio del eje transversal.

align-items: flex-end

Los ítems se alinean al final del eje transversal.

align-items: center

Los ítems se alinean al centro del eje transversal.

Pseudo-clases

Una pseudoclase CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento

seleccionado. Por ejemplo, **:hover** aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

```
div:hover {
  background-color: #F89B4D;
}
```

Las pseudoclases, junto con los pseudoelementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como el historial del navegador (:visited, por ejemplo), el estado de su contenido (como :checked en algunos elementos de formulario), o la posición del ratón (como :hover que permite saber si el ratón está encima de un elemento o no).

Los selectores descendientes

El combinador de un espacio en blanco (que se supone que representan un espacio, o mejor dicho uno o más espacios en blanco) combina dos selectores tales que el selector combinado incluye sólo los elementos que coinciden con el segundo selector para los que hay un elemento ancestro que coincide con el primer selector. Los selectores descendientes son similares a selectores hijos, pero que no requieren que la relación entre los elementos coincidentes ser estrictamente entre padres e hijos.

Sintaxis

```
selector1 selector2 { propiedades de estilos }
```

Ejemplo

```
span { background-color: white; }
div span { background-color: blue; }
```

Span 1.

Span 2.

Span 3.

CSS 3 añade tres nuevos selectores de atributos:

• elemento[atributo^="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada.

• elemento[atributo$="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor termina exactamente por la cadena de texto indicada.

• elemento[atributo*="valor"], selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.

De esta forma, se pueden crear reglas CSS tan avanzadas como las siguientes:

/* Selecciona todos los enlaces que apuntan a una dirección de correo electrónico */

a[href^="mailto:"] { ... }

/* Selecciona todos los enlaces que apuntan a una página HTML */

a[href$=".html"] { ... }

/* Selecciona todos los títulos h1 cuyo atributo title contenga la palabra "capítulo" */

h1[title*="capítulo"] { ... }

Otro de los nuevos selectores de CSS 3 es el "selector general de elementos hermanos", que generaliza el selector adyacente de CSS 2.1.

Su sintaxis es elemento1 ~ elemento2 y selecciona el elemento2 que es hermano de elemento1 y se encuentra detrás en el código HTML. En el selector adyacente la condición adicional era que los dos elementos debían estar uno detrás de otro en el código HTML, mientras que ahora la única condición es que uno esté detrás de otro.

Si se considera el siguiente ejemplo:

h1 + h2 { ... } /* selector adyacente */

h1 ~ h2 { ... } /* selector general de hermanos */

<h1>...</h1>

<h2>...</h2>

...

<h2>...</h2>

<h2>...</h2>

El primer selector (h1 + h2) sólo selecciona el primer elemento <h2> de la página, ya que es el único que cumple que es hermano de <h1> y se encuentra justo detrás en el código HTML. Por su parte, el

segundo selector (h1 ~ h2) selecciona todos los elementos <h2> de la página salvo el segundo. Aunque el segundo <h2> se encuentra detrás de <h1> en el código HTML, no son elementos hermanos porque no tienen el mismo elemento padre.

CSS 3 añade además un nuevo **pseudo-elemento**:

::selection, selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

Las mayores novedades de CSS 3 se producen en las **pseudo-clases**, ya que se añaden 12 nuevas, entre las cuales se encuentran:

elemento:nth-child(numero), selecciona el elemento indicado pero con la condición de que sea el hijo enésimo de su padre. Este selector es útil para seleccionar el segundo párrafo de un elemento, el quinto elemento de una lista, etc.

elemento:nth-last-child(numero), idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

elemento:empty, selecciona el elemento indicado pero con la condición de que no tenga ningún hijo. La condición implica que tampoco puede tener ningún contenido de texto.

elemento:first-child y **elemento:last-child,** seleccionan los elementos indicados pero con la condición de que sean respectivamente los primeros o últimos hijos de su elemento padre.

elemento:nth-of-type(numero), selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo.

elemento:nth-last-of-type(numero), idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

Algunas pseudo-clases como: **nth-child(numero)** permiten el uso de expresiones complejas para realizar selecciones avanzadas:

li:nth-child(2n+1) { ... } /* selecciona todos los elementos impares de una lista */

li:nth-child(2n) { ... } /* selecciona todos los elementos pares de una lista */

/* Las siguientes reglas alternan cuatro estilos diferentes para los párrafos */

p:nth-child(4n+1) { ... }

p:nth-child(4n+2) { ... }

p:nth-child(4n+3) { ... }

p:nth-child(4n+4) { ... }

Empleando la pseudo-clase: **nth-of-type(numero)** se pueden crear reglas CSS que alternen la posición de las imágenes en función de la posición de la imagen anterior:

img:nth-of-type(2n+1) { float: right; }

img:nth-of-type(2n) { float: left; }

Otro de los nuevos selectores que incluirá CSS 3 es: **not()**, que se puede utilizar para seleccionar todos los elementos que no cumplen con la condición de un selector:

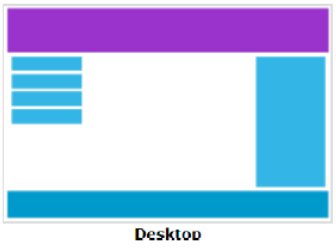
:not(p) { ... } /* selecciona todos los elementos de la página que no sean párrafos */

:not(#especial) { ... } /* selecciona cualquier elemento cuyo atributo id no sea "especial" */

Responsive Web Design

El diseño web responsivo consiste en crear páginas web que se vean bien en todos los dispositivos.

Un diseño web responsive se ajustará automáticamente a diferentes tamaños de pantalla y ventanas gráficas.



The Viewport

Es el visor del área visible del usuario de una página web.

→ varía con el dispositivo y será menor en un teléfono móvil que en una pantalla de computadora.

Configuración de la ventana de visualización (viewport)

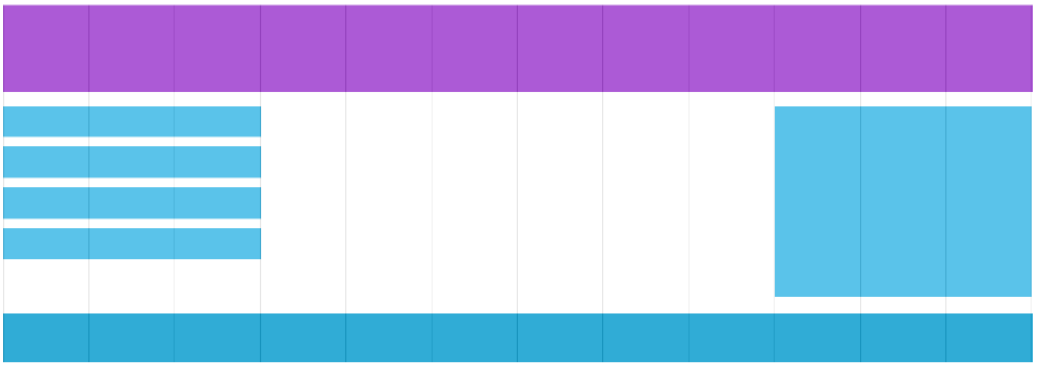
HTML5 introdujo un método para permitir que los diseñadores de páginas web tomen el control sobre la ventana gráfica, a través de la etiqueta **<meta>.**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- **<meta>** → da al navegador instrucciones sobre cómo controlar las dimensiones y la escala de la página.
- **width=device-width** → define el ancho de la página para seguir el ancho de pantalla del dispositivo (que variará dependiendo del dispositivo).
- **initial-scale=1.0** → establece el nivel de zoom inicial cuando la página es cargada por primera vez por el navegador.

Responsive Web Design - Grid-View

Una vista de cuadrícula responsive tiene a menudo tiene 12 columnas y tiene un ancho total del 100%, y se reducirá y expandirá a medida que cambia el tamaño de la ventana del navegador.



Enlaces de interés

https://uniwebsidad.com/libros/css/capitulo-5/posicionamiento-flotante?from=librosweb

https://uniwebsidad.com/libros/css-avanzado/capitulo-1/limpiar-floats?from=librosweb

https://es.learnlayout.com/box-sizing.html

https://www.w3schools.com/html/html_responsive.asp