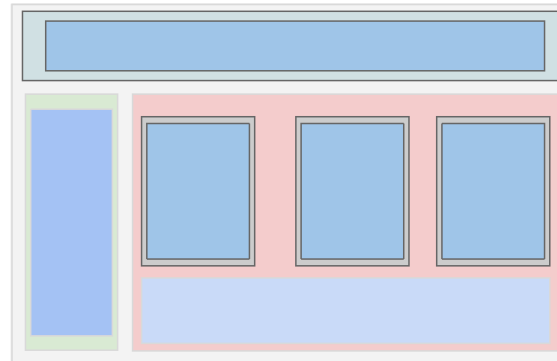


# Aplicaciones Web

## Maquetación con CSS





# Bloque 2 (III)

## Maquetación con CSS

0. [Maquetación web con CSS](#)
1. [Flujo de los elementos](#)
2. [Tipos de layouts](#)
3. [Box-sizing](#)
4. [Centrado del layout](#)
5. [Propiedad position](#)
6. [Columnas](#)
7. [Elementos flotantes](#)
8. [Flexbox](#)
9. [Diseño responsive](#)
- \*. [Referencias](#)



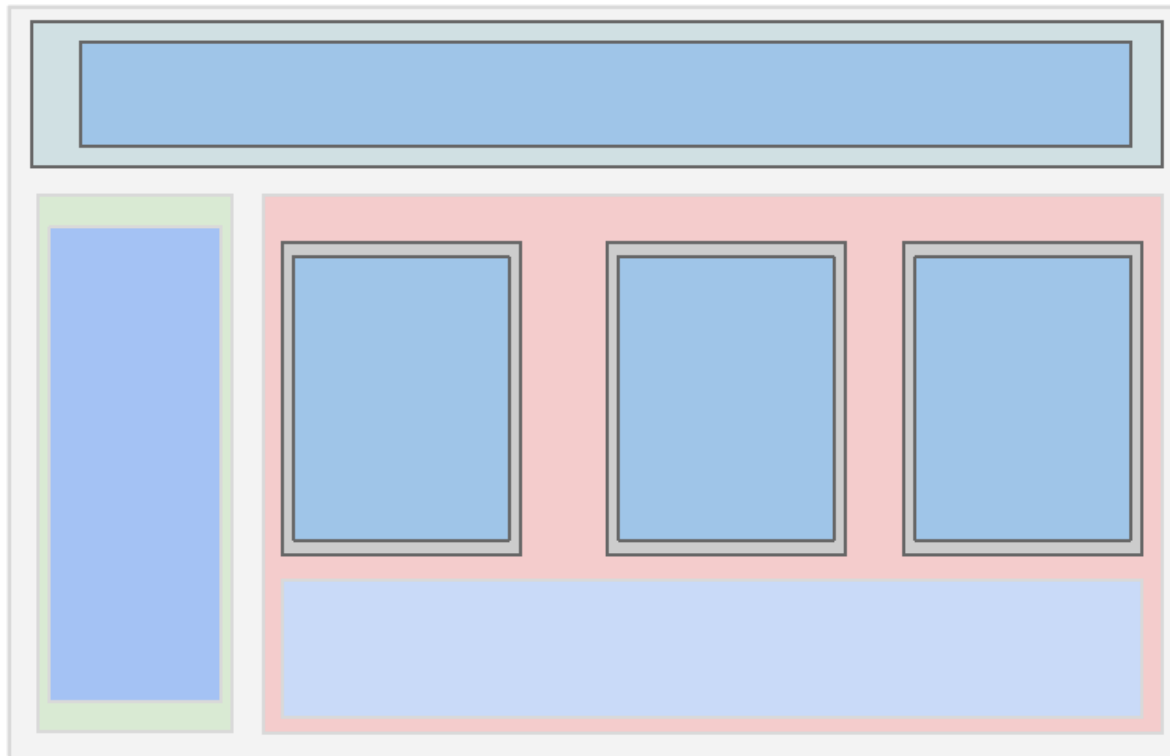
# 0. Maquetación web con CSS

- Definición
  - La **maquetación web** es la parte del diseño que se encarga de definir la **disposición** de los elementos (imágenes, texto, etc...) que componen la página web
  - Define la **estructura visual** de la misma de acuerdo a unos objetivos de comunicación definidos
  - Se realiza mediante etiquetas HTML y propiedades CSS
    - *div, header, footer, nav, section, article, aside, dialog, main, summary...*



# 0. Maquetación web con CSS

- Definición



# 0. Maquetación web con CSS

- Definición





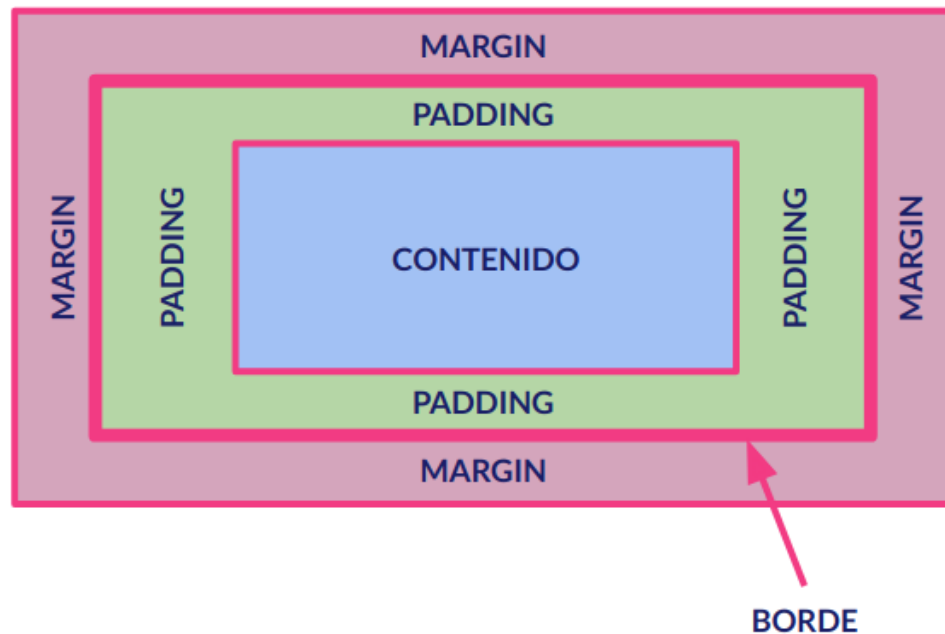
# 0. Maquetación web con CSS

- Recomendaciones
  - Esquema previo
    - “A mano”
    - Software: *MockFlow*, *NinjaMock*, *Wireframe.cc*, ...
  - De lo grande a lo pequeño
    - No colocar lo que hay dentro de una zona del diseño hasta que no se ha colocado el contenedor principal (contenedor padre)
  - Uso de ayuda disponibles en navegadores
  - Probar en distintos navegadores



# 1. Flujo de los elementos

- Modelo de caja
  - Todo elemento HTML es una caja





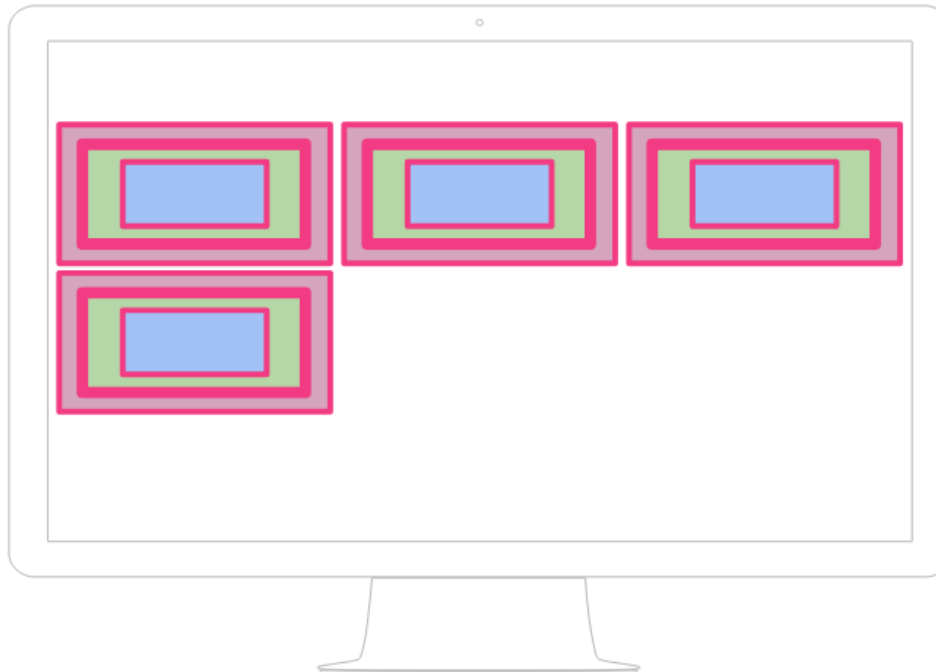
# 1. Flujo de los elementos

- Modelo de caja
  - A tener en cuenta:
    - Los navegadores no hacen NADA para controlar el diseño de nuestra página web
    - Sólo muestran los elementos en ORDEN
    - La maquetación web consiste en disponer estas cajas para que cada una ocupe el lugar que queremos al ser mostradas por el navegador
    - Inicialmente, sólo siguen dos comportamientos básicos:
      - En línea (**inline**)
      - En bloque (**block**)



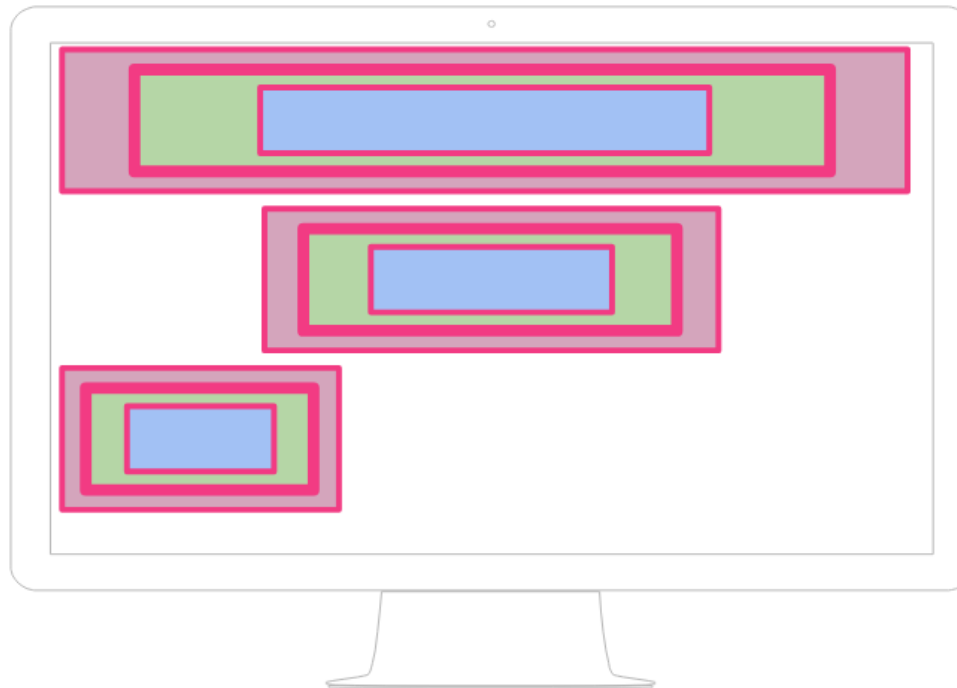
# 1. Flujo de los elementos

- Modelo de caja
  - **En línea (inline)**
    - Tamaño determinado por el contenido
    - No generan saltos de línea



# 1. Flujo de los elementos

- Modelo de caja
  - **En bloque (block)**
    - Tamaño personalizado
    - Generan saltos de línea





# 1. Flujo de los elementos

- Propiedad display
  - No todas las cajas se comportan igual cuando las añadimos
  - Cada etiqueta tiene un comportamiento por defecto pero este se puede modificar con los valores de la propiedad **display** y conseguir el diseño que queremos
  - Valores display:
    - **inline; inline-block**
    - **block**
    - **none**
    - Valores relacionados con **tablas**
    - **flex; grid**



# 1. Flujo de los elementos

- Propiedad display
  - Elementos **inline** e **inline-block**
    - ✓ Los elementos **inline** no rompen el flujo de la línea y se van colocando uno detrás de otro (mientras caben)
    - ✓ Aceptan *margin* y *padding* aunque solo se tienen en cuenta los **valores horizontales**
    - ✓ Ignoran *width* y *height*
  - `<span>`, `<a>`, `<b>`, `<img>*`, ...
  - Los elementos **inline-block** funcionan igual pero podremos asignarles *width* y *height*



# 1. Flujo de los elementos

- Propiedad display
  - Elementos **block**
    - ✓ Los elementos tipo block rompen el flujo de la línea provocando un salto de línea, tanto anterior como posterior
    - ✓ Por defecto, si no lo especificamos, ocuparán toda la anchura de la etiqueta que los contiene (la etiqueta contenedora)
    - ✓ Admiten width y height
  - <h1>, <p>, <section>, <div>, <li>, <nav>, ...



# 1. Flujo de los elementos

- Propiedad display
  - **display: none**
    - ✓ Hace desaparecer al elemento de la página
    - ✓ No dejan un espacio vacío, aunque siguen en el código HTML
    - ✓ La propiedad **visibility: hidden** sí que deja ese hueco aunque no se muestre



# 1. Flujo de los elementos

- Propiedad display
  - Valores relativos a **tablas**
    - Las etiquetas con estos valores simularán el comportamiento del elemento de tabla análogo
    - Valores:
      - ***table***
      - ***table-row***, ***table-cell***
      - ***table-caption***
      - ***table-column***
      - ***table-colgroup***, ***table-header-group***, ***table-footer-group***, ***table-row-group***



## 2. Tipos de layouts

- Tipos de layout
  - Fixed
  - Elastic
  - Fluid (%)
  - Híbrido (con min/max sizing)
  - Responsive





## 2. Tipos de layouts

- Fixed
  - El tamaño en anchura de todos los elementos se establece en **pixeles**
  - Ventajas:
    - Control total sobre el diseño (todos los elementos van a tener siempre el tamaño establecido)
  - Desventajas:
    - En pantallas pequeñas aparecerá *scroll horizontal* (¡NO!)
    - En pantallas grandes genera mucho espacio en blanco a los lados si el contenedor principal no es lo suficientemente ancho



## 2. Tipos de layouts

- Elastic
  - El tamaño en anchura de todos los elementos se establece en **em** (múltiplos del tamaño de letra)
  - Ventajas:
    - El tamaño del texto en em escala correctamente al hacer zoom (se mantienen las proporciones)
  - Desventajas:
    - Si escalamos elementos que ocupen el mismo espacio pueden solaparse
    - En este caso, habría que hacer pruebas en todo tipo de dispositivos y con todo tipo de tamaños de fuente
    - También puede aparecer scroll horizontal, ya que no deja de ser un diseño “fixed”



## 2. Tipos de layouts

- Fluid (Liquid)
  - El tamaño en anchura de todos los elementos se establece con % (con respecto al contenedor/etiqueta padre)
  - Ventajas
    - Los elementos mantienen sus proporciones independientemente del tamaño de pantalla
  - Desventajas
    - En pantallas pequeñas las columnas pueden ser muy estrechas
    - En columnas estrechas los textos largos generan celdas muy alargadas
    - Problemas si hay imágenes y videos con tamaño fijo



## 2. Tipos de layouts

- Híbrido (con max/min width)
  - Es posible combinar layouts de distinto tipo y asignarle a los contenedores un valor de anchura máximo y mínimo
  - **max-width**: en caso de crecer, el contenedor no superará la anchura máxima establecida en pixeles
  - **min-width**: en caso de encoger, el contenedor no será menor que la anchura mínima establecida en pixeles



## 2. Tipos de layouts

- Responsive
  - Layout que cambia dependiendo de las características de la pantalla donde se muestra (normalmente dependiendo sobre todo de la anchura de la pantalla) [“cambio fluido”]
  - Se realiza mediante un diseño “fluid”
  - Si hay más de un layout para la página se denomina **adaptative** [“cambio brusco”]

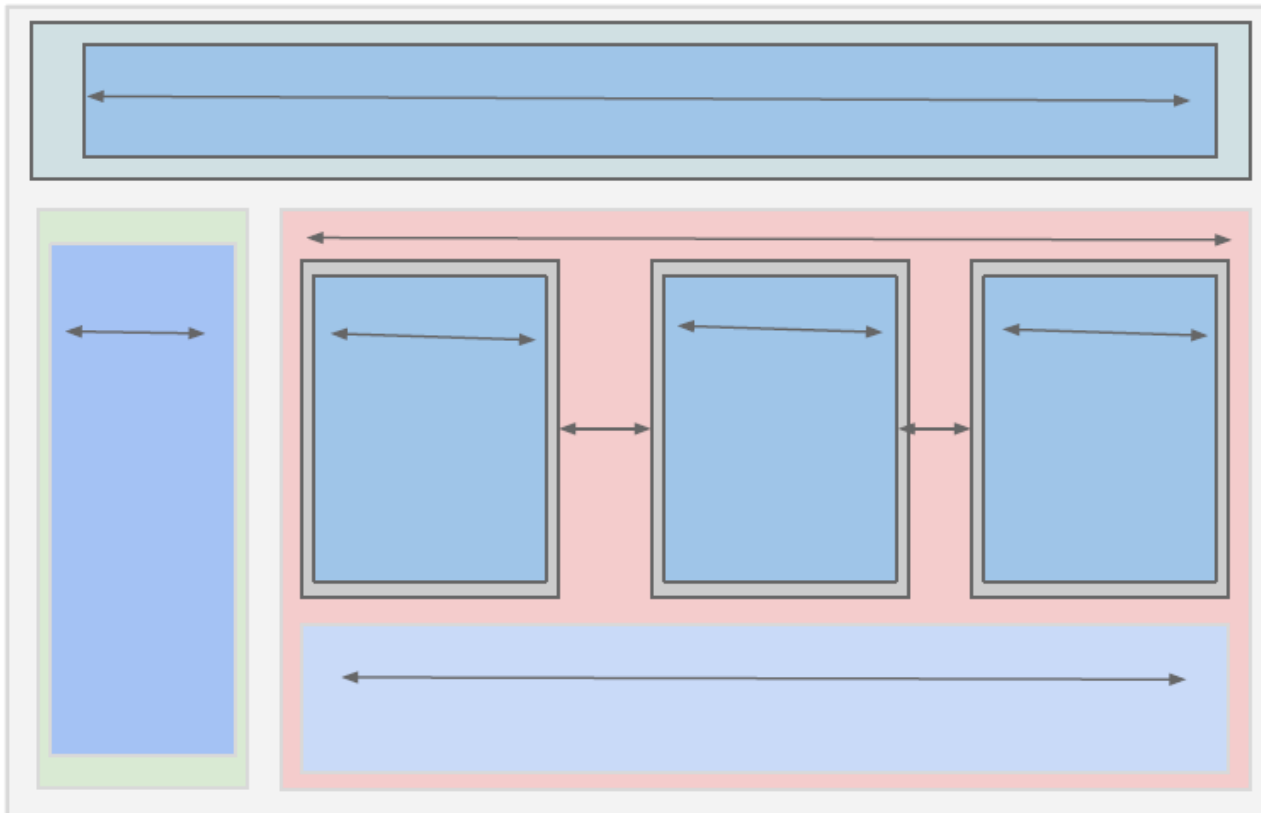


### 3. Box-sizing

- Dimensiones por defecto
  - Según el modelo de caja, los elementos al ser representados por el navegador ocupan el siguiente espacio:
    - **Altura:** *altura* del contenido + *padding* + *borde*
    - **Anchura:** *anchura* del contenido + *padding* + *borde*
  - En diseños complejos, para cuadrar todos los elementos HTML se hace necesario calcular manualmente sus dimensiones: sumar todos los espacios de todas las cajas, añadir los márgenes, etc.
  - Trabajar con las dimensiones por defecto es una **tarea complicada**

### 3. Box-sizing

- Dimensiones por defecto





## 3. Box-sizing

- Propiedad box-sizing: border-box
  - Una solución para maquetar más fácilmente es establecer la propiedad **box-sizing** con el valor **border-box**
  - De esta manera no tendremos que calcular con bordes y paddings, facilitando la tarea
  - El tamaño que demos al elemento será la suma de todo
  - Para ello, añadiremos en nuestro archivo CSS:

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

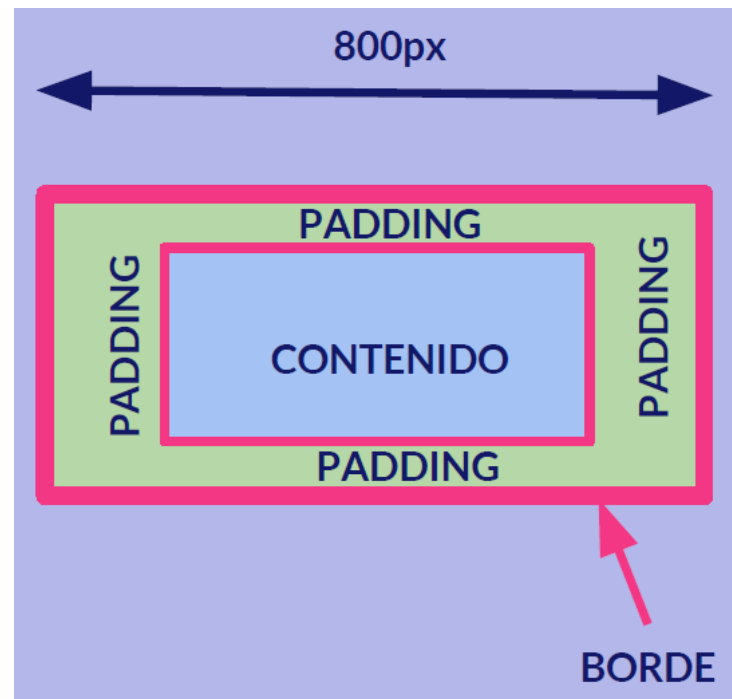




# 3. Box-sizing

- Ejemplo
  - Ya estaremos incluyendo padding y border en la dimensión de la caja


```
* {  
  width: 800px;  
}
```





# 3. Box-sizing

- Otras valores de box-sizing
  - **content-box**: solo dimensionamos el contenido, habría que sumar todos los valores para maquetar (valor por defecto)
  - **padding-box**: no tiene en cuenta el borde pero sí el padding y el contenido (no soportado por todos los navegadores)

```
.padding {  
   box-sizing: padding-box;  
}
```



## 4. Centrado del layout

- Centrar un elemento
  - Cuando hablamos de centrar un elemento nos referimos siempre a un contexto
  - El contexto de referencia siempre es su etiqueta padre o su etiqueta contenedora
  - Suele ser una tarea algo compleja inicialmente
  - Se distingue **centrado horizontal** (más importante) y **centrado vertical**



## 4. Centrado del layout

- Centrado horizontal
  - Centrar elementos en **línea**:
    - ✓ **text-align: center;** (al contenedor padre)
  - Centrar elementos en **bloque**:
    - ✓ **margin: X auto;** al elemento que queremos centrar
    - ✓ El elemento debe de tener anchura
  - Varios elementos en bloque dentro de un contenedor:
    - ✓ **text-align: center;** (al padre)
    - ✓ **display: inline-block;** a los elementos a centrar y anchura
  - Con contenedores flex



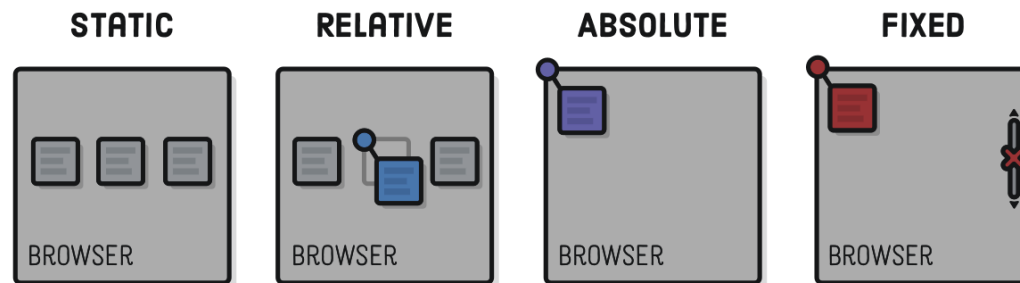
## 4. Centrado del layout

- Centrado vertical
  - Elementos en **línea**:
    - Con el mismo **padding** arriba y abajo
    - **vertical-align: middle** si estamos dentro de una celda de una tabla o lo estamos simulando con la propiedad display (el padre debe de tener altura fija)
  - Elementos en **bloque**:
    - Utilizando la propiedad **position** en el contenedor y en el elemento
  - Con contenedores flex



# 5. Propiedad position

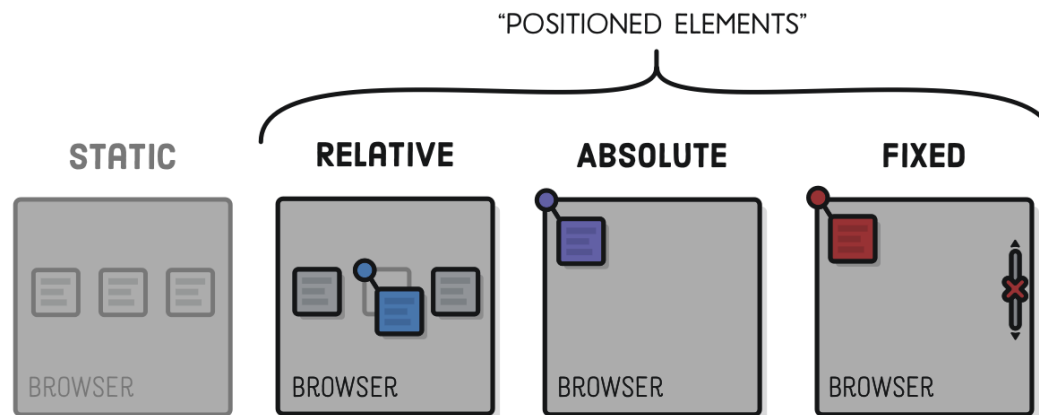
- Posicionamiento de elementos
  - Para posicionamientos más exactos se usa la propiedad CSS **position**
  - Valores: **static**, **relative**, **fixed**, **absolute**, **sticky**, **inherit**
  - Estos valores están relacionados con las propiedades CSS **top**, **bottom**, **left** y **right** (desplazamientos conforme a un punto de referencia concreto) y **z-index** (capas)





# 5. Propiedad position

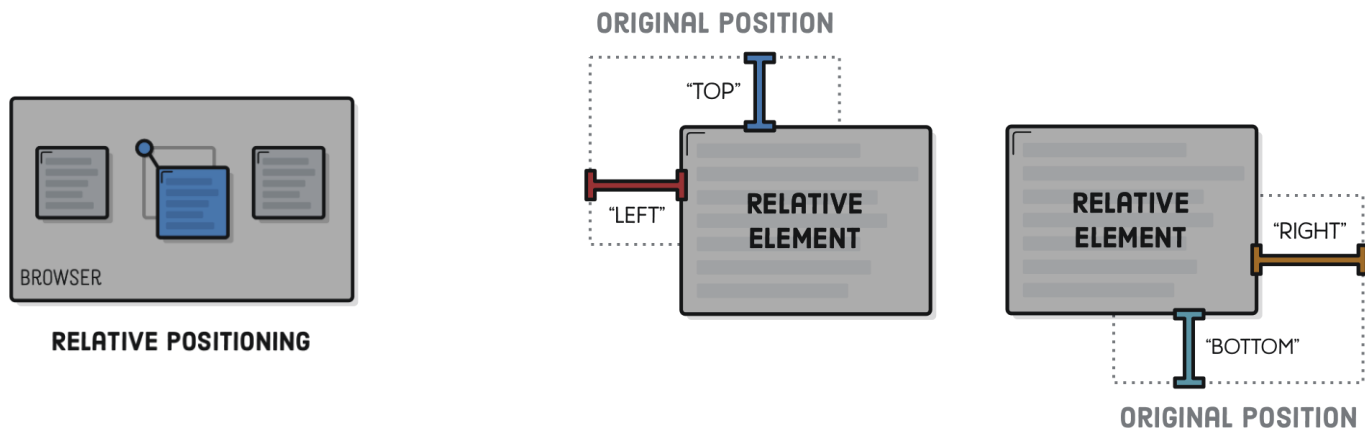
- Posicionamiento de elementos
  - Valores de **position**
    - **static** (valor por defecto): el elemento sigue el flujo que le corresponde; aunque tenga valores en *top*, *bottom*, *left*, *right* o *z-index* NO se aplican





# 5. Propiedad position

- Posicionamiento de elementos
  - Valores de **position**
    - **relative**: como *static* pero SÍ atiende a los desplazamientos *top*, *bottom*, *left*, *right* o *z-index* a partir de la posición que le corresponde según el flujo normal

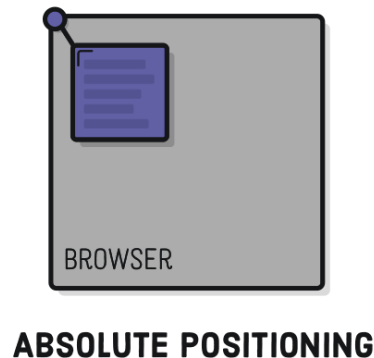






# 5. Propiedad position

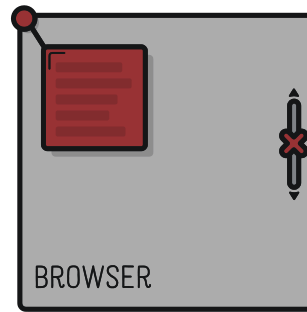
- Posicionamiento de elementos
  - Valores de **position**
    - **absolute**: el elemento se coloca en el sitio que se le indica con *top*, *bottom*, *left*, *right* o *z-index*, al margen del flujo normal de la página; toma como referencia la ventana del navegador o la primera etiqueta padre que tenga *position:relative*





# 5. Propiedad position

- Posicionamiento de elementos
  - Valores de **position**
    - **fixed**: igual que *absolute* pero no atiende al scroll, su posición permanece fija al desplazarnos por la página



**FIXED POSITIONING**



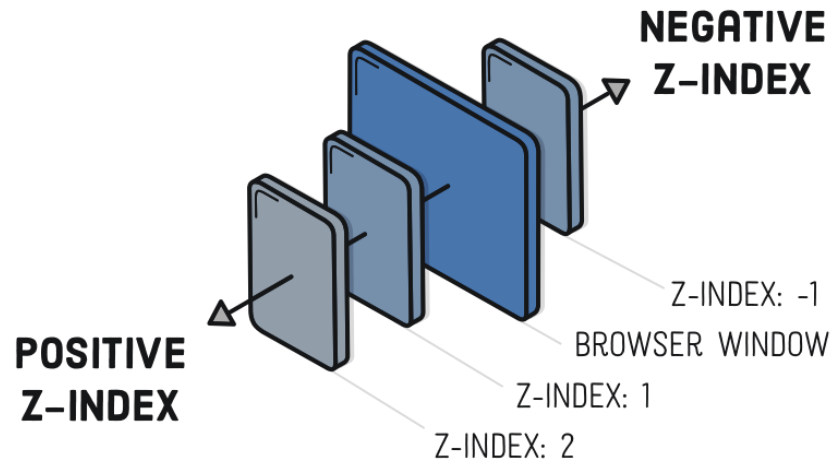
# 5. Propiedad position

- Posicionamiento de elementos
  - Valores de **position**
    - **sticky**: se comporta como *relative* hasta llegar a una posición de scroll y a partir de entonces pasa a *fixed*
    - **inherit**: *position* no se propaga en cascada. Si queremos que sea así, añadiremos el valor *inherit* a los hijos



# 5. Propiedad position

- Propiedad z-index
  - Con z-index podemos establecer capas decidiendo el orden de solapamiento de los elementos
  - Se mostrará arriba aquel elemento con mayor valor z-index





## 5. Propiedad position

- Centrado vertical de elementos en bloque
  - Una vez ya sabemos usar la propiedad *position* podemos centrar verticalmente elementos de bloque
  - Nos encontraremos con dos casos:
    - Cuando conocemos la altura del elemento
    - Cuando desconocemos la altura del elemento



## 5. Propiedad position

- Centrado vertical de elementos en bloque
  - Si conocemos la altura del elemento (por ejemplo 150px):

```
css {  
  .contenedor {  
    position: relative;  
  }  
  
  .elemento_a_centrar {  
    height: 150px;  
    margin-top: -75px; /** La mitad de la altura **/  
    position: absolute;  
    top: 50%;  
  }  
}
```



## 5. Propiedad position

- Centrado vertical de elementos en bloque
  - Si desconocemos la altura del elemento:

```
.contenedor {  
  position: relative;  
}  
.elemento_a_centrar {  
  position: absolute;  
  top: 50%;  
  transform: translateY(-50%);  
}
```



## 6. Columnas

- División del contenido en columnas
  - Propiedades CSS relacionadas con columnas
    - **column-count**: número de columnas del contenedor
    - **column-width**: ancho de las columnas (el navegador calculará cuántas columnas caben con ese ancho)
    - **column-gap**: distancia entre columnas
    - **column-rule**: similar a borde; permite especificar el estilo, color y anchura de la línea que separa las columnas





## 6. Columnas

- División del contenido en columnas
  - Propiedades CSS relacionadas con columnas
    - **column-span**: valores *all* o *none*; indica si el elemento que estará dentro del contenedor donde hemos especificado que habrá columnas, sigue el flujo en columnas o no
    - **column-fill**: cómo se rellenan las columnas; el contenedor debe tener altura. Valores: *auto* o *balance* (todas las columnas la misma altura)
    - **break-inside**: *void* si queremos que el elemento no quede roto de una columna a otra



## 7. Elementos flotantes

- Propiedad float
  - float (**left** o **right**) es una propiedad CSS pensada para especificar cómo se dispone el texto alrededor de una imagen

```
img {  
    float: right;  
    margin: 0px 1em;  
}
```

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?





## 7. Elementos flotantes

- Propiedad float
  - Si sigue habiendo “hueco vertical” en el lugar contrario al que se ha flotado la imagen, los elementos se siguen añadiendo ahí hasta que sobrepasan en la “vertical” al elemento flotante

{ Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?

{ Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?

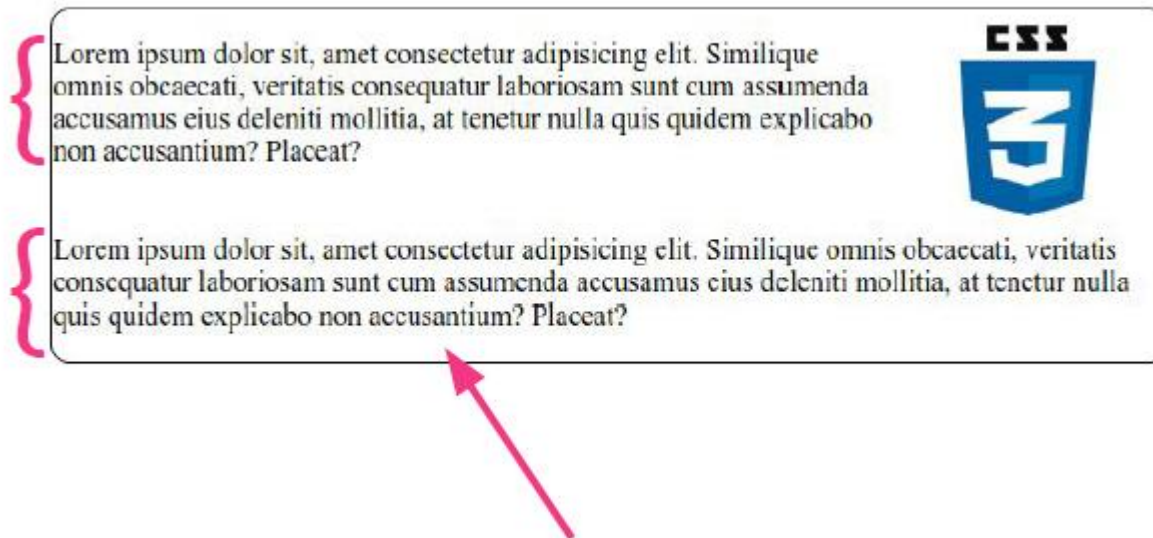
{ Lorem ipsum dolor sit, amet consectetur adipisicing elit. Similique omnis obcaecati, veritatis consequatur laboriosam sunt cum assumenda accusamus eius deleniti mollitia, at tenetur nulla quis quidem explicabo non accusantium? Placeat?





## 7. Elementos flotantes

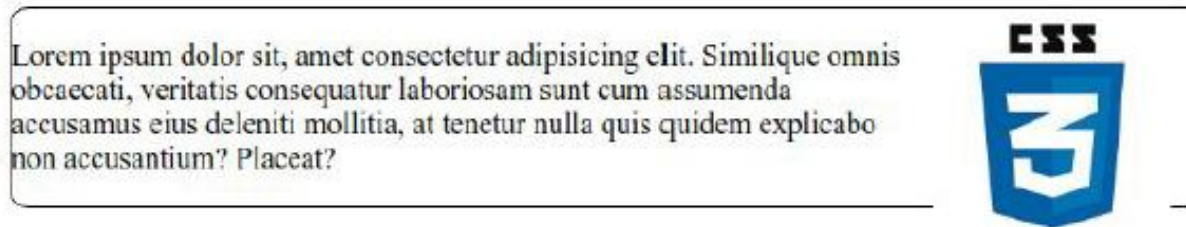
- Propiedad clear
  - Si queremos forzar a que un elemento deje de flotar respecto a otro hay que añadir la propiedad **clear: right** (o **left** o **both**) y ese elemento ya se añadirá tras el fin en vertical del elemento flotante





## 7. Elementos flotantes

- Problema: clearfix hack
  - Posible problema al posicionar:



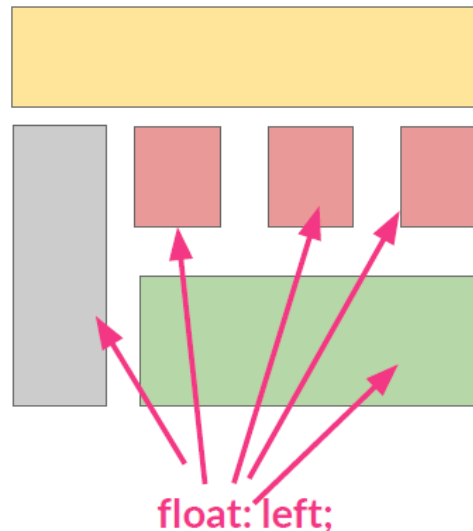
- El contenedor solo tiene en cuenta la altura del párrafo, no la de la imagen (pierde la referencia del elemento flotante)
- Solución (propiedades aplicadas al elemento contenedor):

```
selector {  
    overflow-y: auto;  
    height: altura_suficiente; /*Una de las dos*/  
}
```



# 7. Elementos flotantes

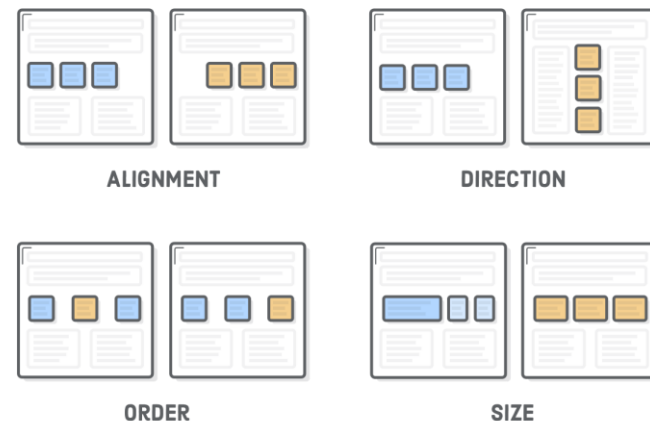
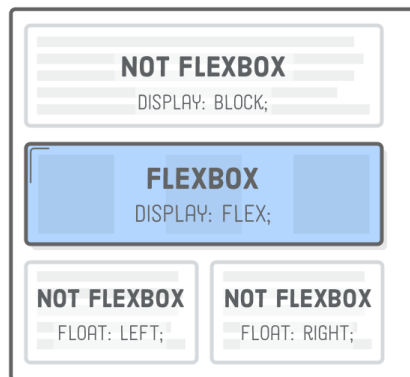
- Usando float para maquetar
  - También se usan elementos flotantes para crear estructuras
  - ¿Cómo?
    - Flotando los elementos según necesitemos -div, nav, header...- (generalmente a la izquierda)
    - Dándoles las dimensiones adecuadas





# 8. Flexbox

- ¿Qué es Flexbox?
  - “Modelo de caja flexible”
  - Módulo de CSS3 que proporciona un control específico sobre la alineación, la dirección, el orden y el tamaño de las cajas
  - Trabaja en **una dimensión**
  - “Gran alternativa a los floats tradicionales”





## 8. Flexbox

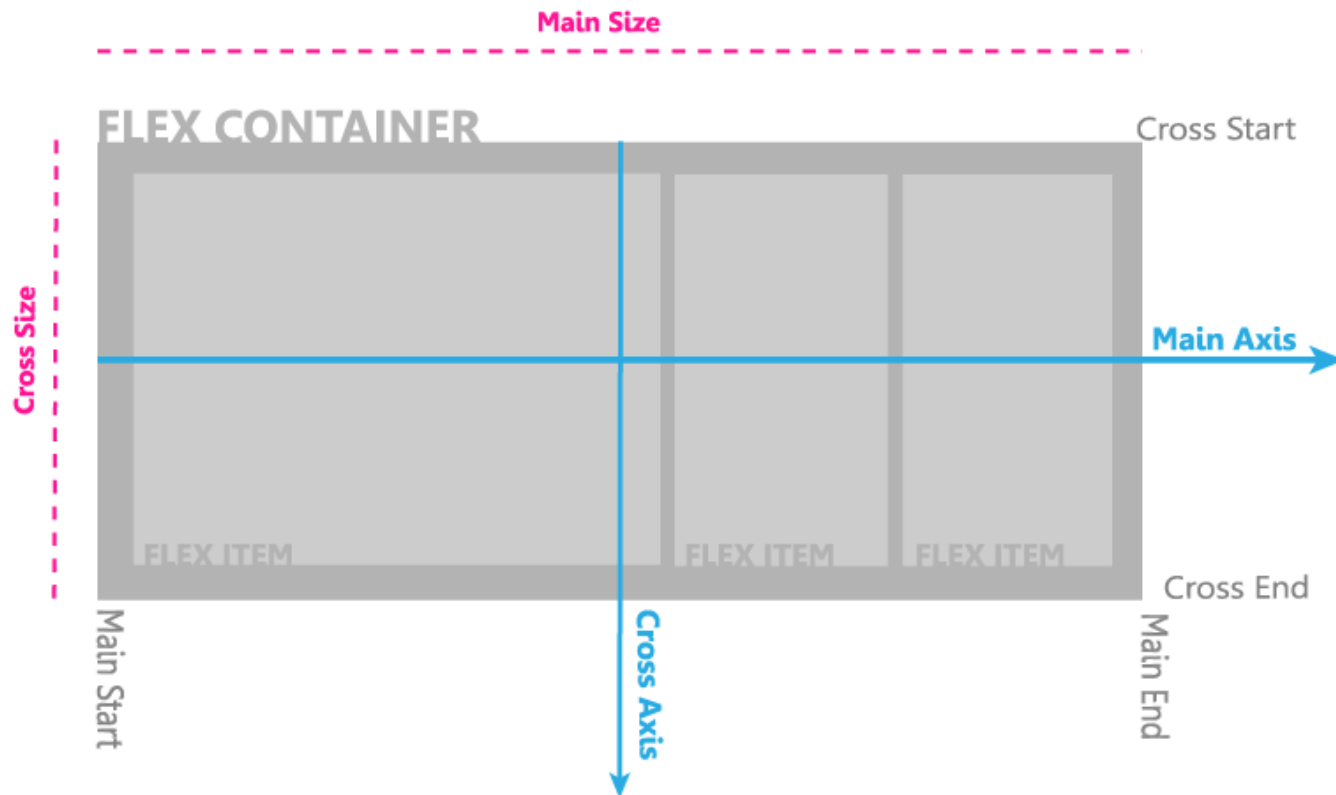
- Ejes en flexbox
  - En Flexbox tenemos **2 ejes**
    - Eje principal (**Main Axis**)
    - Eje cruzado (**Cross Axis**)
  - Por defecto, el eje principal es el eje horizontal y el eje cruzado es el vertical
  - Podemos cambiar este comportamiento con la propiedad ***flex-direction***;





# 8. Flexbox

- Ejes en flexbox





# 8. Flexbox

- Flex container y flex items
  - Distinguiremos entre las propiedades del **contenedor flex** (*flex container*) y las propiedades del **contenido flex** (*flex items*)
  - En el contenedor flex es donde aplicamos **display: flex;**
  - Esta propiedad no se hereda, por tanto **solo afecta a los hijos directos del contenedor flex**



“FLEX CONTAINER”

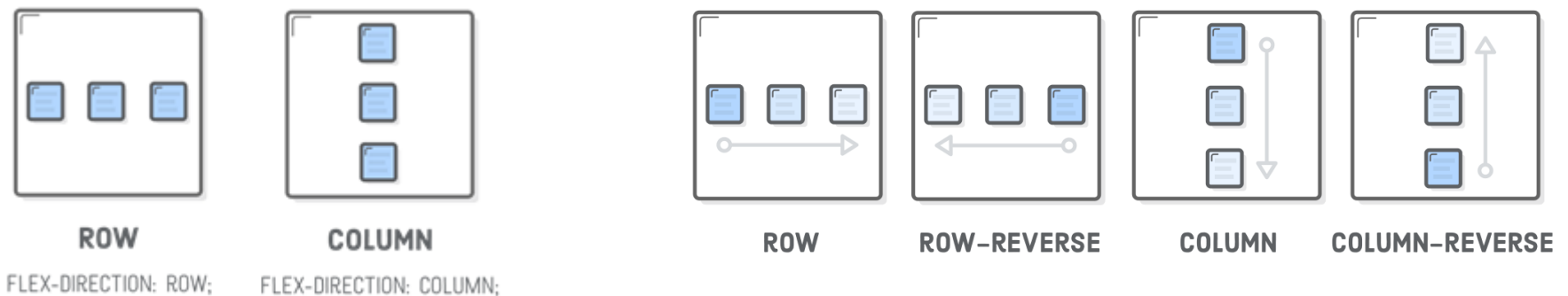


“FLEX ITEMS”



# 8. Flexbox

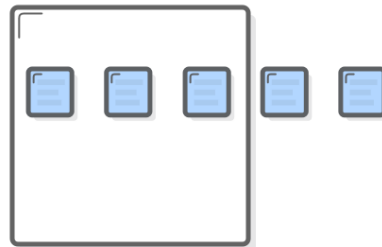
- Propiedades principales contenedores flex
  - **flex-direction**
    - flex-direction: **row**
      - El eje principal es el horizontal y el eje cruzado el vertical
    - flex-direction: **column**
      - Eje principal es el vertical y el eje cruzado el horizontal





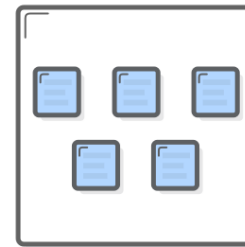
# 8. Flexbox

- Propiedades principales contenedores flex
  - **flex-wrap**
    - wrap / no-wrap / wrap-reverse



**NO WRAPPING**

FLEX-WRAP: NOWRAP;



**WITH WRAPPING**

FLEX-WRAP: WRAP;



## 8. Flexbox

- Propiedades principales contenedores flex
  - **flex-flow**
    - Shorthand (forma reducida) de las propiedades **flex-direction** y **flex-wrap**
    - Ejemplo: **flex-flow**: row wrap;
    - Por defecto flex-flow: row nowrap;



## 8. Flexbox

- Propiedades contenido flex
  - **flex-grow**
    - “Capacidad de crecimiento” de los elementos flex
    - Permite a una caja crecer cuando sobra espacio dentro del espacio flex
    - Si **flex-grow: 1**; la caja crece cuando hay espacio
    - Se puede aplicar a uno o más elementos flex
    - Es una proporción entre los elementos
    - Por defecto **flex-grow: 0**; (no crece aunque haya espacio)



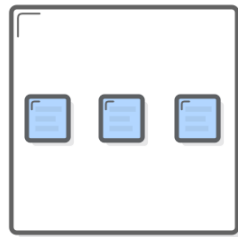
## 8. Flexbox

- Propiedades contenido flex
  - **flex-shrink**
    - “Capacidad de estrechamiento” de los elementos flex
    - Por defecto todos los elementos flex tienen **flex-shrink: 1;**
    - Esto significa que cuando no hay espacio suficiente en el espacio flex, los elementos se reducen proporcionalmente en base a 1
    - Si aplico un valor más alto de flex-shrink a un elemento concreto, se reduciría a un ritmo mayor que otros elementos con un valor menor
    - Si **flex-shrink: 0;** el elemento no reduce

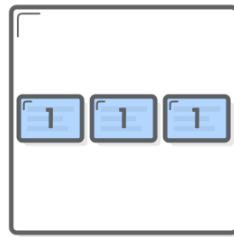


# 8. Flexbox

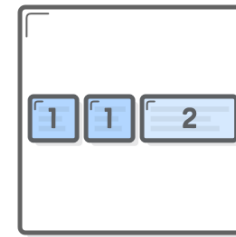
- Propiedades contenido flex
  - **flex-basis**
    - Define la anchura individual de un ítem en un contenedor flex
    - Un elemento con un valor en la propiedad **flex-basis** intentará ajustarse a ese tamaño dentro del eje principal
    - Funciona de manera similar a *width* pero teniendo en cuenta el eje principal según valor de flex-direction



NO FLEX



EQUAL FLEX



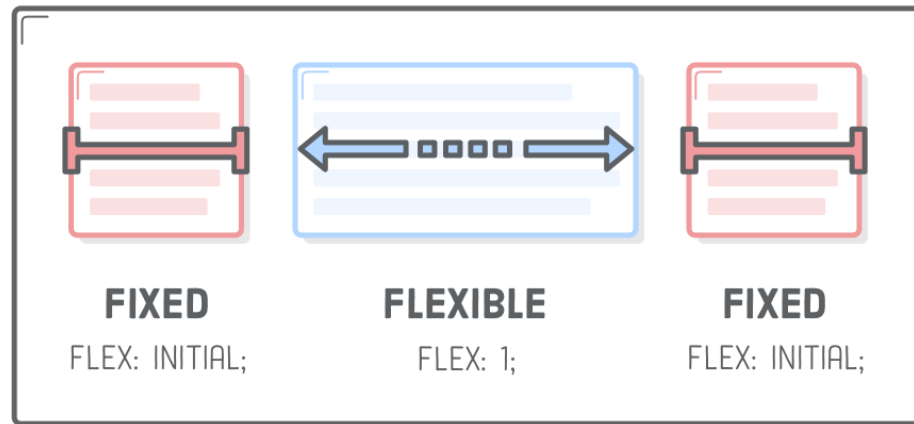
UNEQUAL FLEX





# 8. Flexbox

- Propiedades contenido flex
  - **flex**
    - Shorthand de las propiedades **flex-grow**, **flex-shrink** y **flex-basis** (en este orden):
    - Ejemplo: **flex: 1 0 200px;**





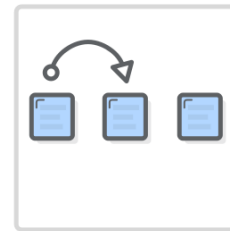
# 8. Flexbox

- Propiedades contenido flex
  - **order**
    - Permite alterar el orden de los elementos flex
    - Por defecto todos los elementos tiene **order: 0**; (aparecen según su flujo HTML)
    - Primero van los valores más negativos
    - Es un cambio visual, ya que no afecta a la navegación por teclado y a los lectores de pantalla (*usar con moderación*)



**FLEX-DIRECTION**

(WHOLE CONTAINER)



**ORDER**

(INDIVIDUAL ITEMS)

## 8. Flexbox

- Alineaciones en el eje principal
  - **justify-content**
    - Permite alinear los elementos en el **eje principal**
    - Valores: *flex-start* (por defecto) / *flex-end* / *center* / *space-between* / *space-around*



JUSTIFY-CONTENT



FLEX-START



CENTER



FLEX-END



SPACE-AROUND



SPACE-BETWEEN

## 8. Flexbox

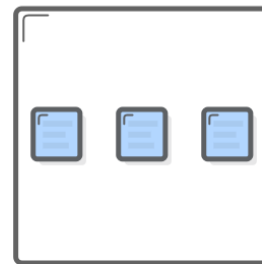
- Alineaciones en el eje cruzado
  - **align-items**
    - Permite alinear los elementos en el **eje cruzado**
    - Valores: *stretch* (por defecto) / *flex-start* / *center* / *flex-end* / *baseline*



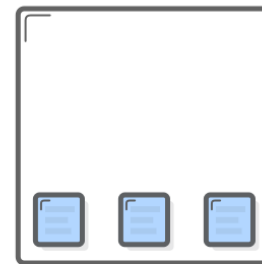
ALIGN-ITEMS



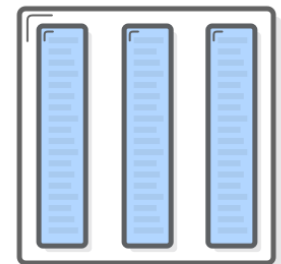
FLEX-START



CENTER



FLEX-END

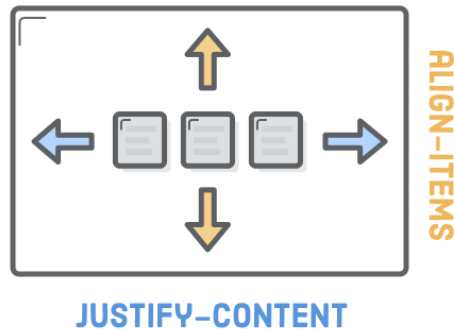


STRETCH

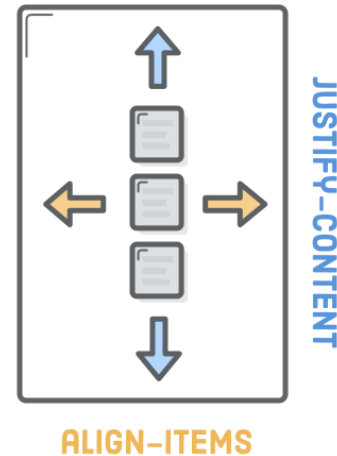
## 8. Flexbox

- Consideraciones sobre la alineación
  - Cambio de eje según valor de flex-direction

FLEX-DIRECTION: ROW;



FLEX-DIRECTION: COLUMN;



## 8. Flexbox

- Agrupación de contenedores flex
  - Es posible agrupar contenedores flex
  - Un elemento flex puede ser a su vez un contenedor flex



**NO GROUPING**

[3 FLEX ITEMS]



**GROUPED ITEMS**

[2 FLEX ITEMS]

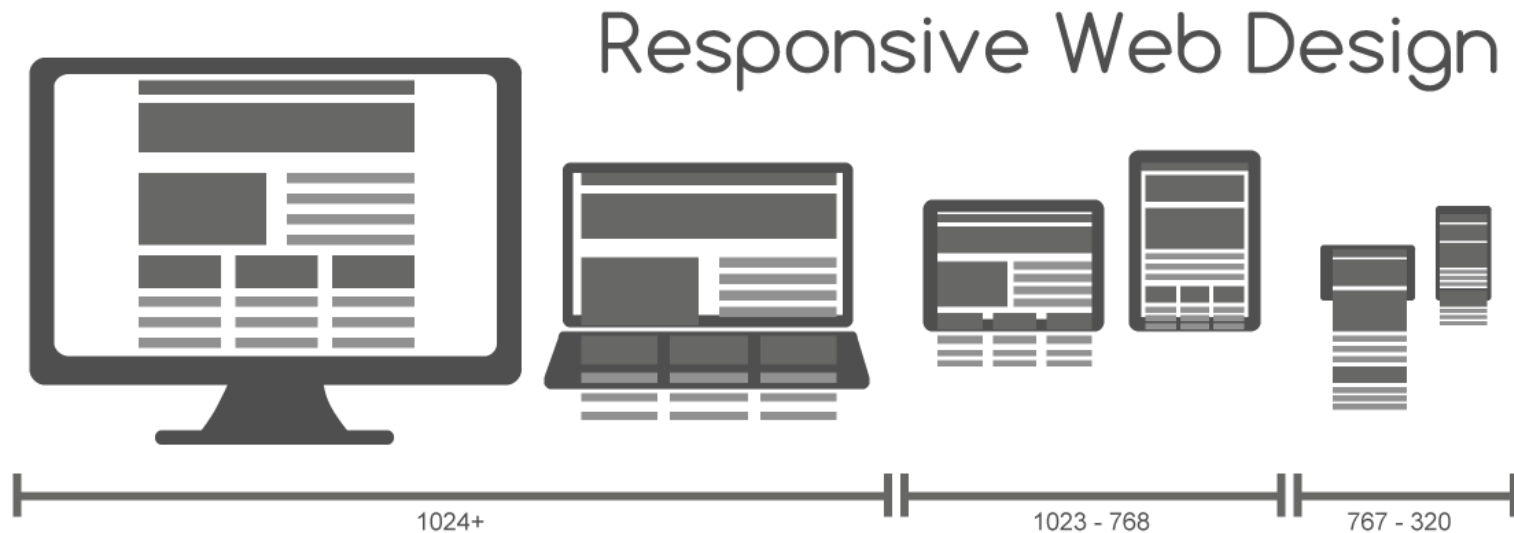


## 9. Diseño responsive

- ¿Qué es “Responsive Web Design”?
  - **Responsive Web Design (RWD) o diseño “responsivo”** es una tendencia de diseño, en la que se desarrolla una única página que se adapta a los diferentes dispositivos en los que se visualizará
  - Tiene en cuenta tamaños de pantalla, densidad de píxeles, orientación (vertical u horizontal) y elementos de interacción (ratón, pantalla táctil,...)
  - Se desarrolla mediante HTML5 + CSS 3 + *media queries*

# 9. Diseño responsive

- ¿Qué es “Responsive Web Design”?







# 9. Diseño responsive

- Breakpoints
  - Punto en el que cambian propiedades de una página, generalmente atendiendo a su anchura
  - Breakpoints habituales:
    - **< 576px** (pantallas pequeñas)
    - **576px - 768px** (móviles apaisados)
    - **768px - 992px** (tablets)
    - **992px - 1200px** (desktop)
    - **> 1200px** (pantallas grandes)



## 9. Diseño responsive

- Media queries
  - Módulo de CSS3 que permite aplicar estilos según el tipo de medio en el que se muestran los documentos
  - A través de la regla `@media` y aplicando distintas condiciones (ancho del medio, orientación, tipo de dispositivo, etc.) se pueden variar los estilos que se utilizan en el documento

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

```
@media print {
    body {
        font-size: 1.1 em;
    }
    nav {
        display: none;
    }
}
```



## 9. Diseño responsive

- Media queries
  - La etiqueta @media reconoce varios tipos de medios predefinidos, al margen de sus propiedades
    - **all** → todos los tipos de medios
    - **screen** → pantallas de ordenadores, tablets, smartphones,...
    - **print** → impresoras
    - **tv** → televisores
    - **tty** → terminales
    - **speech** → utilizado por lectores de pantalla



# 9. Diseño responsive

- Media queries
  - Condiciones
    - Existe un amplio conjunto de propiedades que se pueden comprobar para aplicar o no determinados estilos
      - *width* | *min-width* | *max-width*
      - *height* | *min-height* | *max-height*
      - *orientation* (*landscape* / *portrait*)
      - *aspect-ratio* | *min-aspect-ratio* | *max-aspect-ratio*
      - *color* | *min-color* | *max-color*



# 9. Diseño responsive

- Ejemplos

```
/* Estilos para todo tipo de pantallas con una anchura máxima  
de 576px*/  
@media all and (max-width: 576px) {  
    .....;  
}  
  
/* Estilos para pantallas con al menos 992px de anchura y que estén apaisadas (más ancho que alto)*/  
@media screen and (min-width: 992px) and (orientation: landscape) {  
    .....;  
}  
  
/* Estilos sólo para pantallas que tengan al menos 768px de anchura*/  
@media only screen and (min-width: 768px) {  
    .....;  
}
```



# \*. Referencias

- Bibliografía y referencias
  - Cursos de Openwebinars.net “*Curso de HTML5 y CSS3*”, “*Maquetación web con CSS*” y “*Curso de Flexbox y CSS Grid*” de Juan Diego Pérez
  - Curso Openwebinars “Flexbox y CSS Grid” de Juan Diego Pérez
  - Sitio web <https://internetingishard.com/html-and-css/>