CSS

MAQUETACIÓN WEB

Maquetación WEB

Antes de comenzar propiamente a maquetar deberíamos:

- Realiza siempre un boceto, esquema.
- Diseña siempre “de lo grande a lo chico”.
- Utiliza las herramientas disponibles.
- Copia.
- Prueba en distintos navegadores.
- Paciencia..

Maquetación WEB

Boceto, Esquema.

- Es bueno pararse a pensar antes de lanzarse a desarrollar. Si inviertes un poco de tiempo en pensar tu diseño de antemano al final seguro que ahorras tiempo..
- Para esto lo mejor es realizar bocetos de nuestro diseño. Estos bocetos podremos hacerlo principalmente de dos maneras: **A mano** o con **Herramientas**

Maquetación WEB

Esquemas previos.

Algunas herramientas para realizar estos bocetos son:

- mockflow.com
- moqups.com
- wireframe.cc
- balsamiq.com
- ninjamock

Maquetación WEB

Copia.

Hay millones de webs. Si ves alguna que te guste usa las herramientas de los navegadores e investiga cómo lo han hecho. Utiliza el conocimiento de otras personas en tú favor.

Pero ojo, no te limites a copiar y pegar. Así no aprenderás. Entiende lo que están haciendo y usa la documentación técnica si hay partes que no ves claras.

Maquetación WEB

Usa las herramientas disponibles.

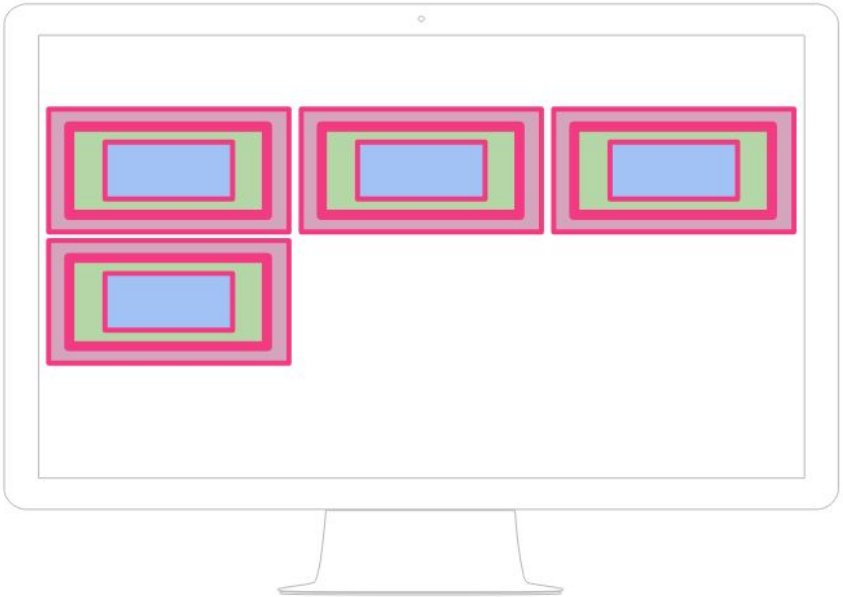
Actualmente todos los navegadores incluyen por defecto herramientas para desarrolladores que nos van a facilitar mucho la vida.

Maquetación WEB

Tener siempre en cuenta:

- Debemos recordar que todas las etiquetas que se van a representar son cajas.
- Los navegadores no hacen NADA para controlar el diseño de nuestra página Web.
- Los navegadores, lo único que hacen es mostrar los elementos de nuestra página HTML en el mismo ORDEN en el que los hemos escrito.
- Siguen solo dos reglas básicas dependiendo de las propiedades de las cajas:

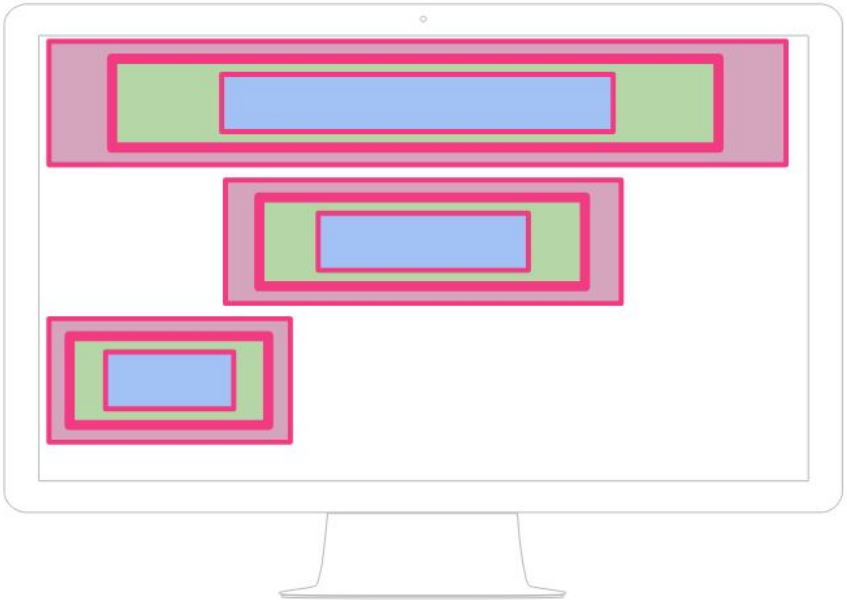
Maquetación WEB



Determinados tipos de caja se van poniendo unas detrás de otros mientras quepan en la pantalla.

Cuando no caben las cajas pasan a la “siguiente línea” del navegador.

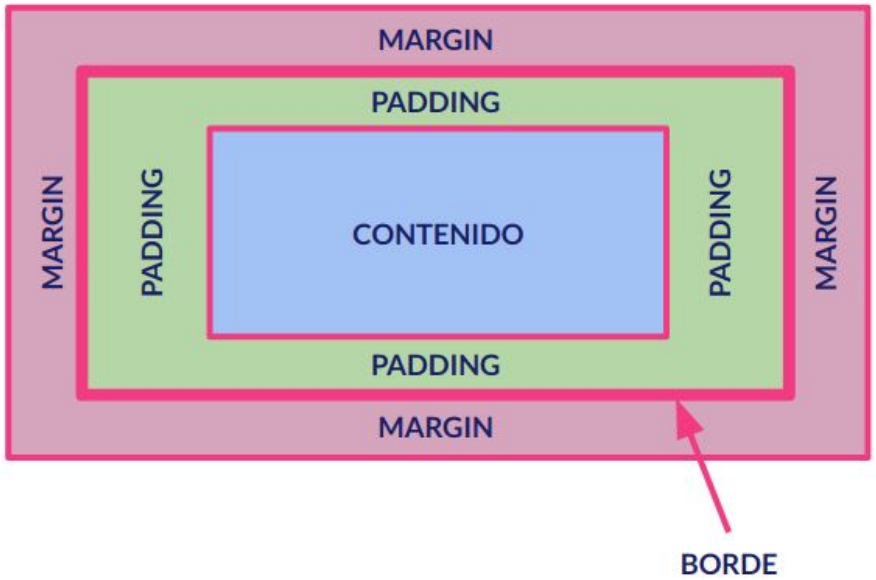
Maquetación WEB



Otros tipos de caja provocan un “salto de línea”.

Y no hay más. Todas estas propiedades de las cajas, toda la maquetación, debemos hacer nosotros usando CSS.

Maquetación WEB



Recuerda

margin : Espacio entre cajas

padding: espacio entre el borde de la caja y el contenido

borde: rodea a toda la caja y puede tener un grosor y un estilo de línea.

La propiedad CSS box-sizing

Por defecto en el modelo de caja de CSS, el ancho y alto asignado a un elemento es aplicado solo al contenido de la caja del elemento. Si el elemento tiene algun borde (border) o relleno (padding), este es entonces añadido al ancho y alto a alcanzar el tamaño de la caja que es desplegada en pantalla. Esto significa que cuando se definen el ancho y alto, se tiene que ajustar el valor definido para permitir cualquier borde o relleno que se pueda añadir.

La propiedad CSS box-sizing

La propiedad box-sizing puede ser usada para ajustar el siguiente comportamiento:

- **content-box** es el comportamiento CSS por **defecto** para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 pixeles, la caja del contenido del elemento tendrá 100 pixeles de ancho, y el ancho de cualquier borde o relleno se añadirá al ancho final desplegado.

La propiedad CSS box-sizing

- **border-box** le dice al navegador tomar en cuenta para cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 pixeles. Esos 100 pixeles **incluirán cualquier borde o relleno** que se añadan, y la caja de **contenido se escogerá para absorber ese ancho extra**. Esto típicamente hace mucho más fácil dimensionar elementos.

CSS

POSICIONAMIENTO DE ELEMENTOS

Elementos en línea

Para explicarlo de manera sencilla, y simplificando, ciertas etiquetas HTML no afectan en absoluto al flujo de los demás elementos, limitándose a marcar ciertos fragmentos con una determinada semántica y dejando que el texto y otros elementos contiguos sigan fluyendo en la misma línea, colocándose a ambos lados de dicho elemento.

Ejemplos de estas etiquetas son: **<a>, ,<del>...** y quizá la más útil de todas: la etiqueta ****, que se usa para envolver elementos en-línea para darles estilo sin cambiar su comportamiento. A estos elementos HTML se les llama elementos de línea o, incluso más a menudo, por su denominación en inglés: elementos "inline".

Elementos de bloque

Por el contrario, **ciertas etiquetas se renderizan en el navegador en líneas independientes**, no mezcladas con el resto del texto. Ejemplos de estas etiquetas son los encabezados (`<h1>` hasta `<h6>`), las citas en bloque (`<blockquote>`), por supuesto los párrafos (`<p>`), y quizá la más conocida de todas que es la etiqueta **`<div>`** usada normalmente para envolver a otros elementos. A estos elementos se les denomina **elementos de bloque**.

Ver Vídeo

En línea y en bloque, la propiedad display

Como hemos visto cada etiqueta tiene un valor por defecto. Con la propiedad display podemos cambiarlo. Los valores que puede tomar son muchos pero los más usados son:

- **inline.** el elemento se renderiza en línea con otros elementos
- **inline-block.** fluyen con el texto y demás elementos como si fueran elementos en-línea y además respetan el ancho, el alto y los márgenes verticales.
- **block**. hace que el comportamiento del elemento sea como un bloque.
- **none.** Son elementos que desaparecen de la página. No dejan un espacio vacío aunque siguen en el código HTML.
- **Valores relacionados con tablas.**
- **Valores flex y grid**

Referencia display

En línea y en bloque, la propiedad display

Elementos con valores relativos a tablas en la propiedad display, al poner uno de estos valores en la propiedad display a una etiqueta HTML esta etiqueta simulará el comportamiento del elemento de tabla análogo. De esta manera tenemos los siguientes posibles valores.

- table
- table-row
- table-cell
- table-caption
- table-column
- table-colgroup
- table-header-group
- table-footer-group
- table-row-group

La propiedad CSS z-index

Indica el orden de un elemento posicionado y sus descendientes.

Cuando varios elementos se superponen, los elementos con mayor valor z-index se superponen a aquellos con menor valor.

Ver

POSICIONAMIENTO DE ELEMENTOS

El posicionamiento de elementos siempre ha provocado muchos ''dolores de cabeza'' a los diseñadores web. El posicionamiento ''clásico'' se basa en un **modelo de bloques** que se van colocando para llenar el espacio de la página **de izquierda a derecha** y de **arriba abajo**. El problema de esta opción es que **no se adapta fácilmente a dispositivos móviles**, donde nos encontramos con cambios de tamaño u orientación del navegador, además de no permitir reordenar los elementos para ajustarlos a los distintos tipos de dispositivo.

POSICIONAMIENTO DE ELEMENTOS

Por suerte existe un nuevo **modelo de posicionamiento flexible** que simplifica todas estas tareas considerablemente. El modelo flexible es quizás demasiado detallado para organizar la estructura general de la página web. En su lugar se está imponiendo el **modelo de rejilla**, donde **se divide el espacio en una rejilla virtual y a continuación se indica cuántas celdas de dicha rejilla ocupa cada componente** dependiendo del tamaño y orientación del dispositivo. Estos dos modelos permiten realizar diseño adaptativo

Modelo de bloque

Las cajas de los elementos se colocan por defecto siguiendo el ''flujo normal'', es decir, empujando las cajas hacia la izquierda y hacia arriba. Sin embargo, podemos cambiar el ''flujo normal'', haciendo que las cajas se posicionen según uno de los siguientes modos de acuerdo a la **propiedad position**:

Estática, Relativa, Absoluta, Fija, Flotante

Modelo de bloque

Estática (position: static): es el modo de posicionamiento por defecto. La caja ocuparía la posición en la que quedaría en un ''flujo normal''.

Modelo de bloque

Relativa (position : relative): se desplaza de su posición estática estándar usando propiedades top, right, bottom y left. Al igual que con las dimensiones de la caja, estas dimensiones indican el espaciado que se deja a cada lado de acuerdo con la posición original de la caja. El resto de las cajas no se ven afectadas por este desplazamiento relativo, por lo que la reservan espacio.

Modelo de bloque

Absoluta (position: absolute): elimina la caja del flujo normal y coloca la caja en una posición fija de manera absoluta con respecto a su caja contenedora. Para colocarla se usan de nuevo las propiedades **top, right, bottom y left**. Puede superponerse a otros elementos, ya que no se le reserva espacio.

Modelo de bloque

Adhesivo Sticky (position: sticky**):** Se comporta como relative hasta llegar a una posición de scroll y a partir de entonces fixed.

Modelo de bloque

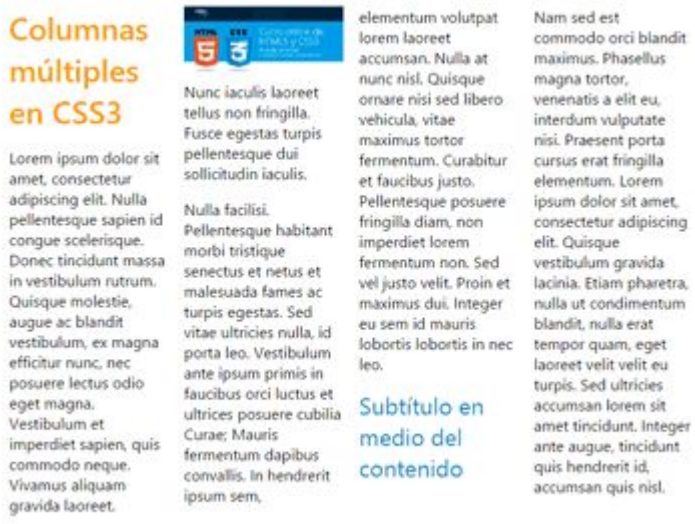
Fija (position : fixed): igual que el posicionamiento absoluto, pero especificando la posición con respecto a la ventana. Esto implica que la caja mantiene su posición y no se mueve cuando se usan las barras de desplazamiento.

Modelo de bloque

Flotante (position: float): es el modo de posicionamiento flotante, de modo que la caja se desplaza hacia un lado (float: left o float : right) llevándola a uno de los extremos de la caja contenedora o hasta la primera caja flotante que se encuentre en esa dirección. Esto implica que se puede superponer a otras cajas no flotantes

Columnas

EN CSS podemos dividir el **contenido** que queremos mostrar en varias columnas tal y como podemos hacen los periódicos.



Columnas múltiples en CSS3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla pellentesque sapien id congue scelerisque. Donec tincidunt massa in vestibulum rutrum. Quisque molestie, augue ac blandit vestibulum, ex magna efficitur nunc, nec posuere lectus odio eget magna. Vestibulum et imperdiet sapien, quis commodo neque. Vivamus aliquam gravida laoreet.

Nunc iaculis laoreet tellus non fringilla. Fusce egestas turpis pellentesque dui sollicitudin iaculis.

Nulla facilisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed vitae ultricies nulla, id porta leo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Mauris fermentum dapibus convallis. In hendrerit ipsum sem,

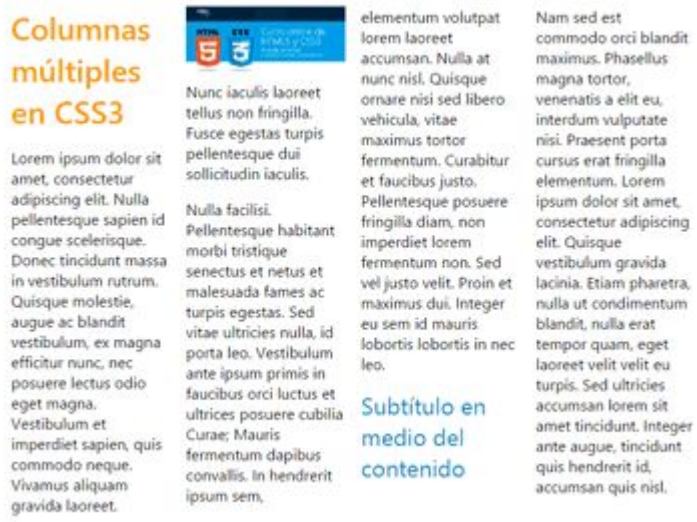
elementum volutpat lorem laoreet accumsan. Nulla at nunc nisl. Quisque ornare nisi sed libero vehicula, vitae maximus tortor fermentum. Curabitur et faucibus justo. Pellentesque posuere fringilla diam, non imperdiet lorem fermentum non. Sed vel justo velit. Proin et maximus dui. Integer eu sem id mauris lobortis lobortis in nec leo.

Subtítulo en medio del contenido

Nam sed est commodo orci blandit maximus. Phasellus magna tortor, venenatis a elit eu, interdum vulputate nisi. Praesent porta cursus erat fringilla elementum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vestibulum gravida lacinia. Etiam pharetra, nulla ut condimentum blandit, nulla erat tempor quam, eget laoreet velit velit eu turpis. Sed ultricies accumsan lorem sit amet tincidunt. Integer ante augue, tincidunt quis hendrerit id, accumsan quis nisl.

Columnas

Por otro lado, hay que tener cuidado con la altura final del contenido, porque la experiencia del usuario se puede degradar al tener que ir haciendo scroll de arriba para abajo para poder leer todo.



Columnas

Por ejemplo, tenemos una página web que es un simple **elemento main** y dentro de él tenemos un **título h1**, **varios párrafos**, un **subtítulo h2** en medio del contenido y una **imagen**.

Ver

Columnas

Con la propiedad **column-count** indicamos el **nº de columnas**

Ahora, simplemente añadiendo la propiedad **column-count:**4; al elemento contenedor tendremos todo el contenido dividido en 4 columnas.

```
main {
  column-count:4;
}
```

Ver

Columnas

Al **título del contenido** y a la **imagen** le vamos a aplicar la propiedad **column-span:all**; de esta forma se **expandirán** hasta ocupar el total del ancho que ocupan todas las columnas:

```
h1 {column-span:all;}
img {display:block;column-span:all;}
```

A la imagen hay que añadirle un `display-block;` porque **la propiedad `column-span` sólo se aplica a elementos de bloque**.

Ver

Columnas

Podemos definir el espacio entre columnas (usando **column-gap**) o incluso ponerle un filete **separador** (una línea entre las columnas) con la propiedad **column-rule**:

```
main { column-count:4;
       column-gap: 3em;
       column-rule: 1px solid #bbb;
        }
```

Ver

Columnas

Podemos darles un **ancho mínimo a las columnas** usando la propiedad **column-width** para que nuestro contenido se vaya adaptando al espacio disponible. En este caso el navegador **colocará todas las columnas que pueda con ese ancho mínimo, hasta un máximo de 4**, que es el valor que le hemos dado a column-count.

Ver

La propiedad float

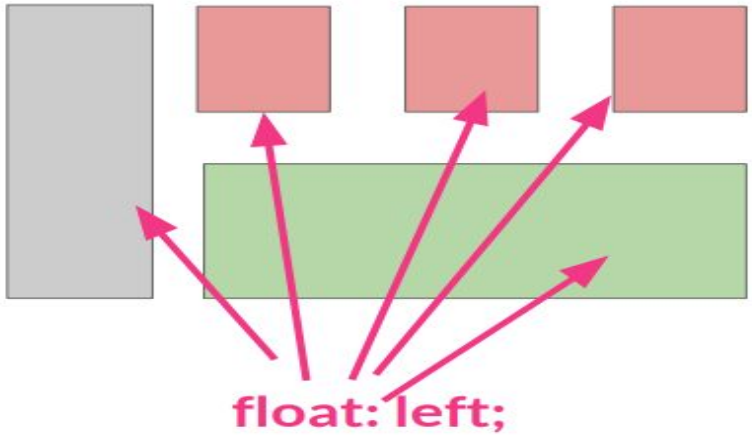
Permite que los **elementos floten sobre el lado que se indica**, mientras que el resto de los elementos que no tienen la propiedad float definida se encuentran alrededor.

Los elementos flotantes pueden tener tres valores: **left, right y none**. El primero define que el elemento fluirá hacia la izquierda, el segundo a la derecha, mientras que el tercero quita la propiedad.

Por defecto, los elementos cuya propiedad float no se encuentra definida, es none.

La propiedad float

la propiedad float (left o right) está pensada para especificar cómo se dispone un texto alrededor de una imagen, pero se utiliza en maquetación web para disponer contenedores en el lugar apropiado.



Ver

La propiedad float

El html del ejemplo es el siguiente, como podemos observar hemos creado un contenedor principal main y dentro tenemos un elemento aside 3 article y un footer.

```
<body>
  <main>
    <aside></aside>
    <article></article>
    <article></article>
    <article></article>
    <footer></footer>
  <main>
</body>
```

Ver

La propiedad float

Al elemento main le hemos dado un ancho del 90% y un margen superior e inferior de 20px y unos márgenes automáticos a izquierda y derecha para que se centre el contenido.

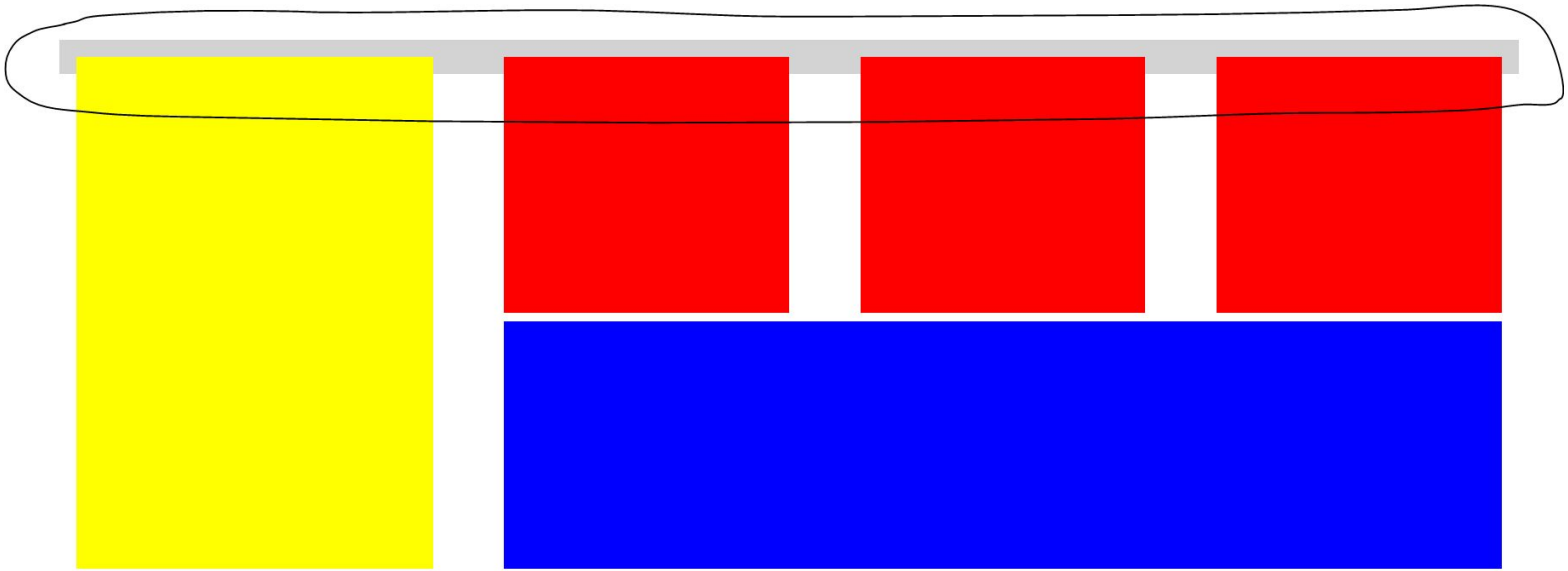
overflow-y: auto; Conseguimos que el contenido no desborde como en la siguientes imagen.

```
main {
  background-color: lightgrey;
  margin: 20px auto;
  padding: 20px;
  overflow-y: auto;
  width: 90%;
}
```

Ver

La propiedad float

overflow-y: auto; Conseguimos que el contenido no desborde como en la siguientes imagen. También podríamos haberlo solucionado dandole una altura.



La propiedad float

Al elemento aside le hemos dado un ancho del 25% y que flote a la izquierda.

```
aside {
  background-color: yellow;
  float: left;
  width: 25%;
  height: 600px;
}
```

Ver

La propiedad float

A los elementos article le hemos dado un ancho del 20% y que flote a la izquierda.

```
article {
  background-color: red;
  float: left;
  height: 300px;
  margin-left: 5%;
  width: 20%;
}
```

Ver

La propiedad float

A los elementos article le hemos dado un ancho del 70% y que flote a la izquierda.

```
footer {
  background-color: blue;
  float: left;
  height: 290px;
  margin-left: 5%;
  margin-top: 10px;
  width: 70%;
}
```

Ver

CSS

MAQUETACIÓN FLEX

Ver

MAQUETACIÓN FLEX

La idea principal de la maquetación con elementos FLEX es que vamos a tener un elemento (una etiqueta) padre que va a poder controlar propiedades de los elementos que contiene hijos.

ELEMENTOS FLEX

Elemento principal (contenedor)

El contenedor flexible se vuelve flexible estableciendo la propiedad `display` en *flex*

```
display: flex;
```

Ver

PROPIEDADES MODIFICABLES (de los elementos flexibles)

El contenedor Flex nos permite controlar las propiedades de los elementos flexibles que hay dentro

- La altura
- La anchura
- El orden
- La alineación vertical
- La alineación horizontal
- La distribución a lo largo del contenedor

LA DIRECCIÓN: FLEX-DIRECTION

- row (default)
- row-reverse
- column
- column-reverse

row y rowreverse

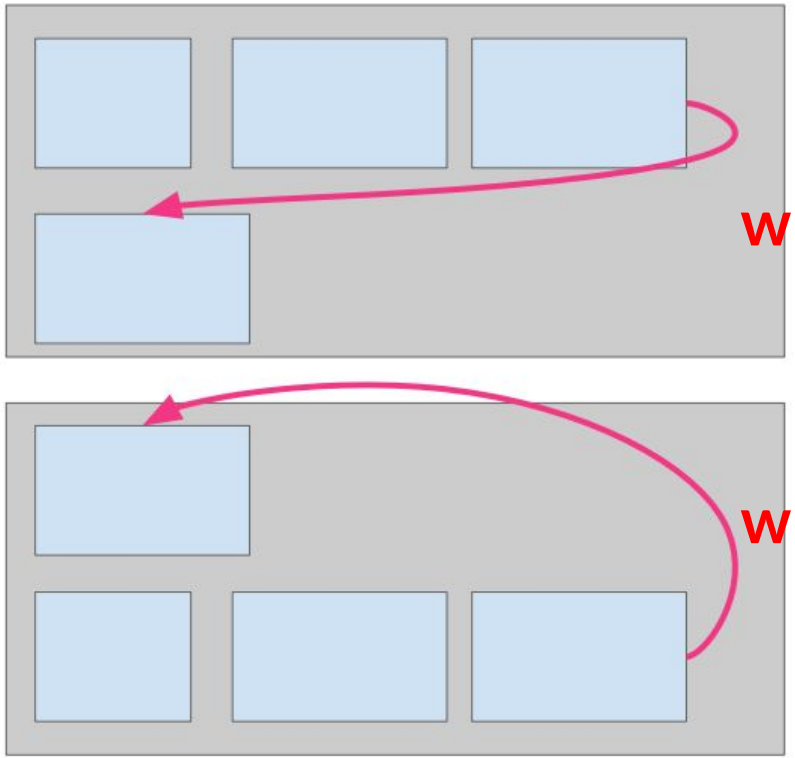
column y column-reverse

AJUSTE FLEX-WRAP

Si los elementos no caben podemos utilizar la propiedad Flex-Wrap

nowrap (default) Una sola línea
wrap: los elementos flexibles se disponen en múltiples líneas si no caben. De arriba abajo.
wrap-reverse: lo mismo que el anterior pero de abajo a arriba.

AJUSTE FLEX-WRAP



Wrap y wrap-reverse

AJUSTE FLEX-FLOW

Las dos propiedades, flex-direction y flew-wrap podemos juntarlas en la propiedad flex-flow con dos partes como por ejemplo:

flex-flow: row wrap;

Alineación horizontal de los elementos flexibles: justify-content

Podemos alinear horizontalmente los elementos flexibles, tengan o no tengan establecida una anchura, añadiendo la propiedad **justify-content** al elemento contenedor. Esta propiedad puede tener 6 valores distintos:

- **flex-start**: Los elementos flexibles se sitúan al principio.
- **flex-end**: Los elementos flexibles se situán al final.
- **center**: Los elementos se centran horizontalmente

Ver

Alineación horizontal de los elementos flexibles

- **space-beetween**: Distribuye el espacio restante entre los elementos pero el primero y el último están en los bordes.
- **space-around:** Distribuye el espacio restante entre los elementos pero no tiene en cuenta la distancia a los bordes.
- **space-evenly**: Distribuye el espacio restante entre los elementos y tiene en cuenta la distancia a los bordes.

Ver

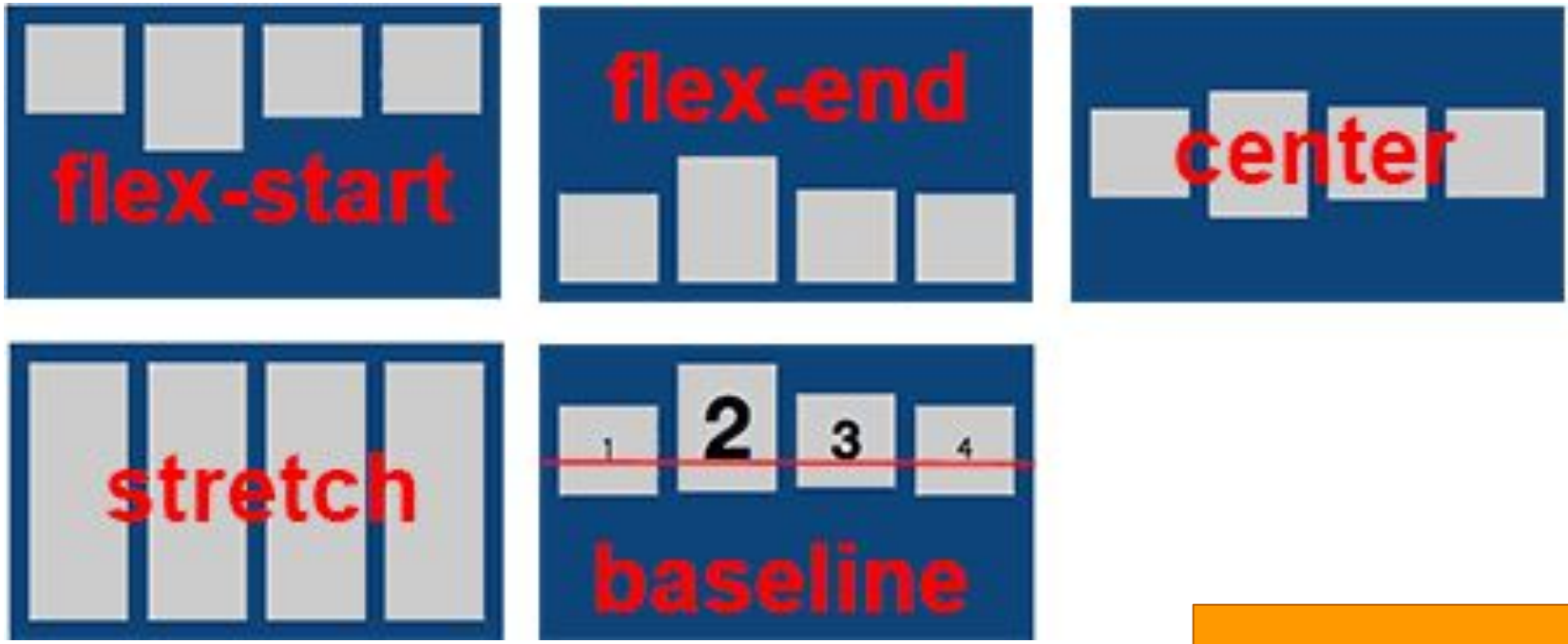
Alineación vertical: align-items

Podemos alinear verticalmente los elementos flexibles añadiendo la propiedad **align-items** que puede tomar los siguientes valores:

- flex-start: Los elementos se ponen junto al borde superior.
- flex-end: Los elementos se ponen junto al borde inferior.
- center: Los elementos flexibles se centran verticalmente.
- stretch: Los elementos crecen en altura para ocupar toda la altura del contenedor flexible. No deben tener altura fija establecida.
- baseline: Los elementos se alinean en relación con la primera línea de texto que posean los elementos flexibles.

Ver

Alineación vertical: align-items



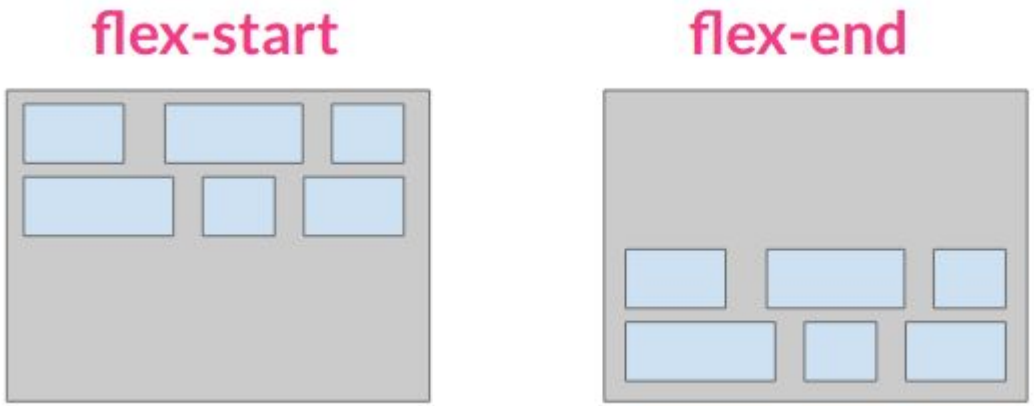
Ver

Alineación vertical - Wrap (cuando tengo varias líneas)

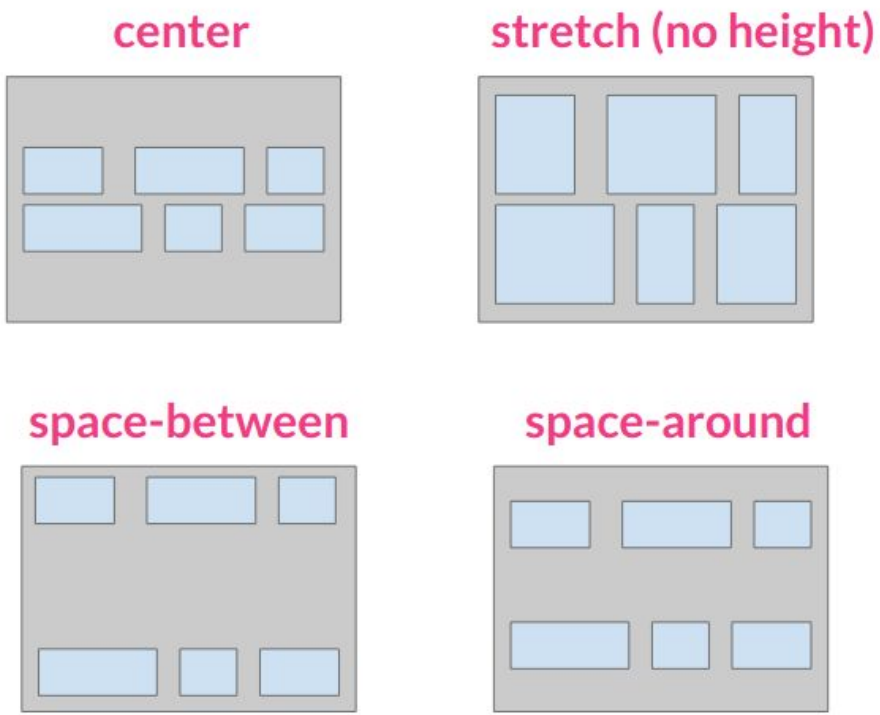
Si he usado la propiedad **flex-wrap:wrap** y resulta que tengo varias líneas de elementos flexibles también puedo alinearlas usado la propiedad CSS **align-content** con valores que son análogos a los vistos antes:

- flex-start
- flex-end
- center
- stretch
- space-between
- space-around

Alineación vertical - Wrap (cuando tengo varias líneas)



Alineación vertical - Wrap (cuando tengo varias líneas)



El orden de los elementos flexibles.

Los elementos flexibles se muestran dentro del contenedor flex en el mismo orden en que están escritos en nuestro código HTML.

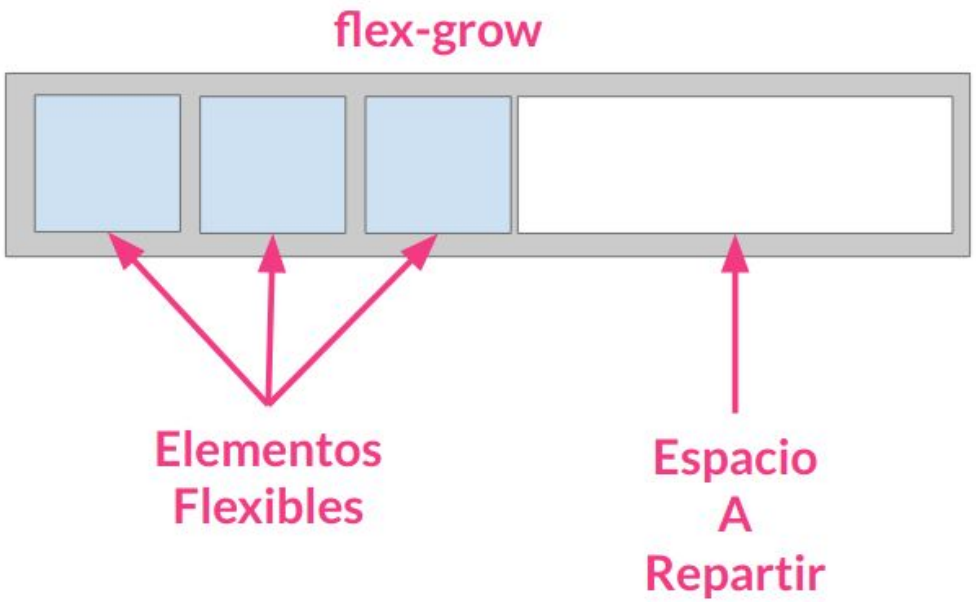
Si queremos modificar esto debemos añadir la propiedad CSS ***order*** a los elementos cuyo orden queremos modificar.

Por **defecto este valor es 0** y se mostrarán en orden ascendente (los de menor valor primero). En caso de empate se muestra antes el que primero estuviera en el código.

Ver

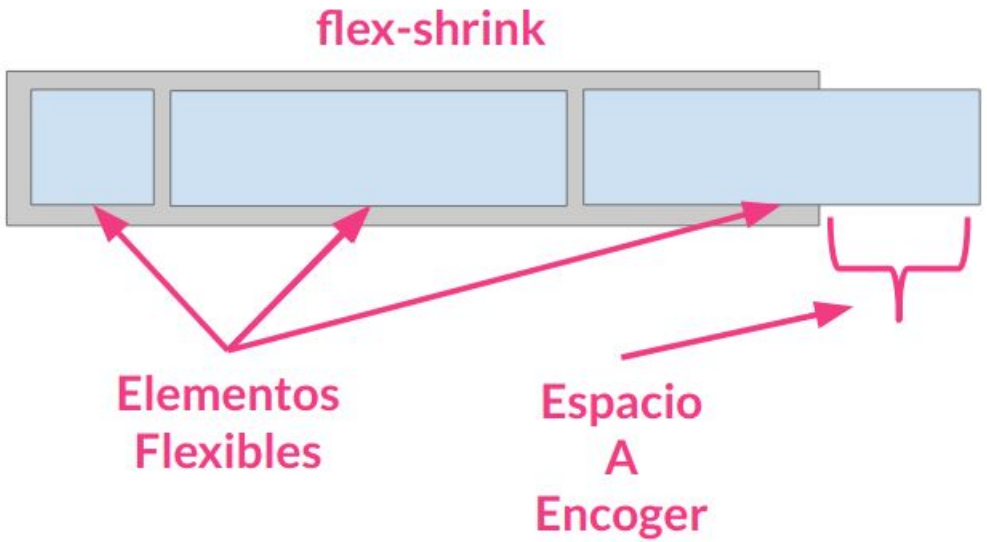
Ajustando el tamaño de los elementos flexibles.

flex-grow: que sirve para indicar, mediante un número, el factor de crecimiento de un elemento flexible cuando se distribuye entre los elementos flexibles el espacio restante. Por defecto es 1 pero si quiero que un elemento participe en el reparto debo añadirle esta propiedad.



Ajustando el tamaño de los elementos flexibles.

flex-shrink: que sirve para indicar, mediante un número el factor de contracción de un elemento flexible cuando el tamaño de todos sobrepasa el tamaño del contenedor. Por defecto es 1 pero si quiero que un elemento participe en la contracción debo añadirle esta propiedad.



Ajustando el tamaño de los elementos flexibles.

flex-basic: que sirve para indicar el tamaño de un elemento antes de que el espacio restante (negativo positivo) se distribuya. Por defecto el valor de esta propiedad es auto y hace que la anchura del elemento flexible se ajusta a su contenido.

Ver

Alineación vertical

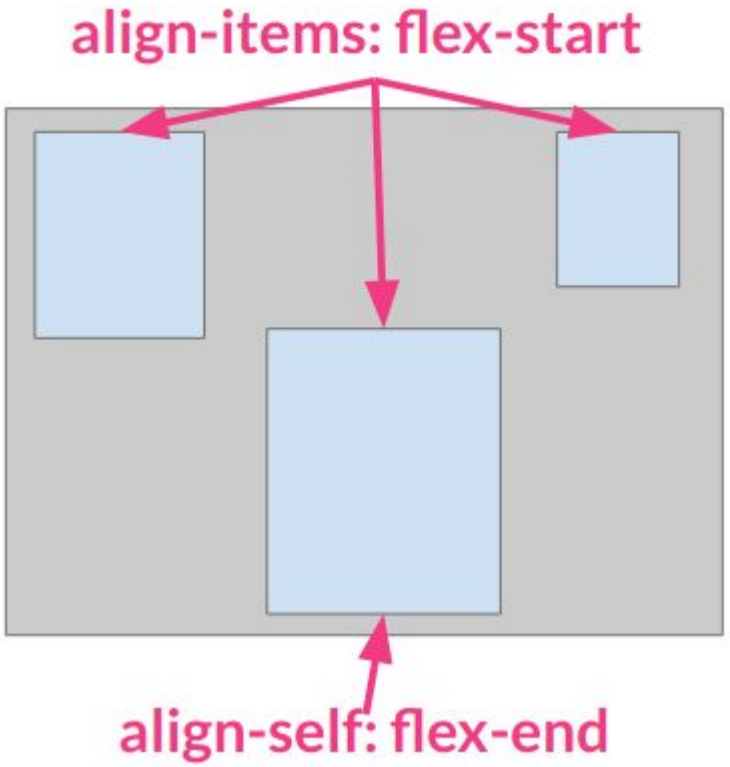
Desde el contenedor FLEX podíamos controlar la alineación vertical de todos los elementos flexibles a la vez con la propiedad **align-items**.

Si queremos que un elemento tenga una alineación propia debemos usar la propiedad **align-self**.

VALORES DE ALIGN-SELF

- flext-start
- flex-end
- center
- stretch (no height)
- baseline

Los elementos flexibles no se ven afectados por vertical-align.



CSS

MAQUETACIÓN GRID

Ver

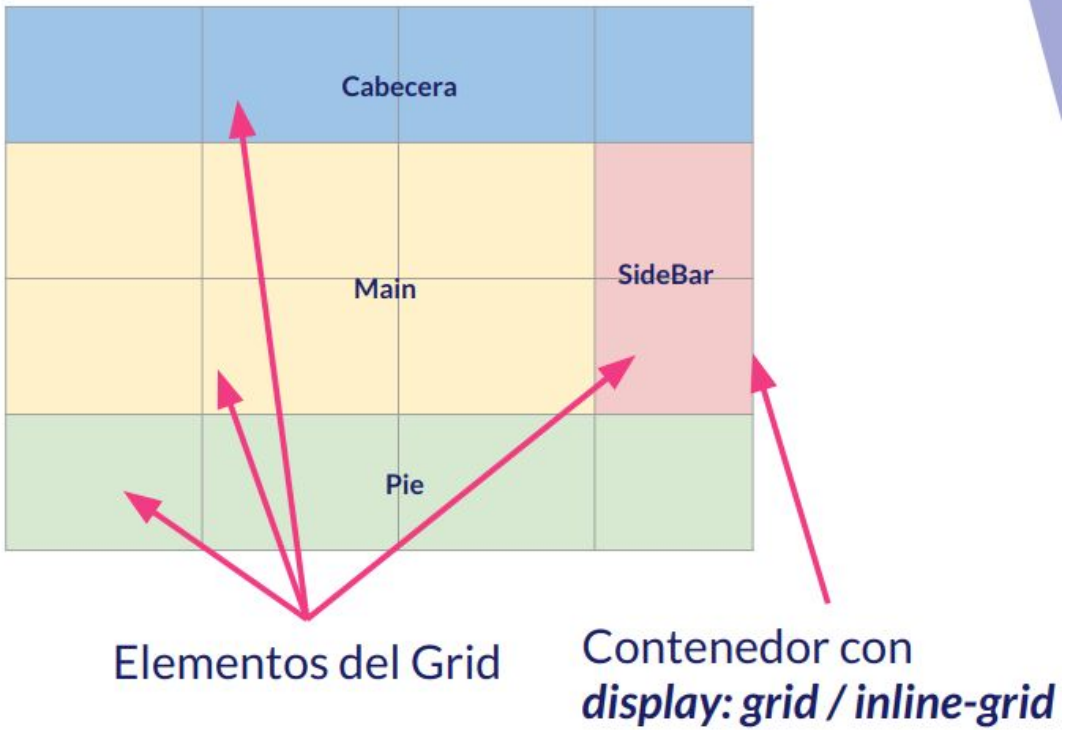
MAQUETACIÓN GRID

La idea principal que hay detrás de la maquetación GRID es que vamos a tener un elemento, es decir, un etiqueta que nos va a permitir *controlar* propiedades de los elementos que contiene y establecer estructuras complejas para distribuirlos.

```
.container {
    display: grid;
}
```

es un sistema de maquetación basado en rejillas, celdas y se caracteriza por ser bidimensional, independiente del orden del markup y flexible. Extremadamente flexible.

MAQUETACIÓN GRID



MAQUETACIÓN GRID

Qué vamos a controlar

- La estructura en filas y columnas y la separación entre ellas.
- Definir áreas del GRID con nombre.
- La alineación horizontal y vertical de los elementos del GRID y del propio GRID dentro del elemento que lo contiene.

MAQUETACIÓN GRID

Para empezar a maquetar usando GRID lo primero que tenemos que hacer es definir cuál de nuestras etiquetas HTML se va a convertir en el contenedor GRID. Una vez lo hemos decidido le daremos una de estas propiedades:

display:grid si queremos que nuestra rejilla (nuestro grid) sea un elemento de bloque.

display:inline-grid si queremos que nuestro grid sea un elemento en línea.

MAQUETACIÓN GRID. Definición de la estructura del GRID

Una vez hemos decidido que estructura queremos usaremos una serie de propiedades para definirla. Las más importantes para empezar son las siguientes:

grid-template-columns: Para definir el **número y tamaño** de las diferentes **columnas** de mi estructura. Debo de poner tantos valores de anchura como columnas quiero que tenga el GRID.
grid-template-rows: Para definir el **número y tamaño** de las diferentes **filas** de mi estructura. Debo de poner tanto valores de altura como filas quiero que tenga el GRID.
grid-row-gap: Para establecer la **separación entre las diferentes columnas**.
grid-column-gap: Para establecer la s**eparación entre las diferentes filas**.

MAQUETACIÓN GRID. Definición de la estructura del GRID

Ejemplos con columnas:

```
/* Tres columnas que se reparten el 100% del contenedor*/
grid-template-columns: 20% 50% 30%;

/* Cuatro columnas. Tres de tamaño fijo 100px y la otra ocupa el espacio libre restante */
grid-template-columns: 100px auto 100px 100px;

/* Cuatro columnas. Todas con un tamaño igual */
grid-template-columns: auto auto auto auto;

/* Tres columnas cada una con nombre (entre []). Dos con tamaño fijo y la otra ocupando el espacio restante */
grid-template-colums: [id] 100px [nombre] 300px [apellidos] auto;
```

MAQUETACIÓN GRID. Definición de la estructura del GRID

Ejemplos con filas:

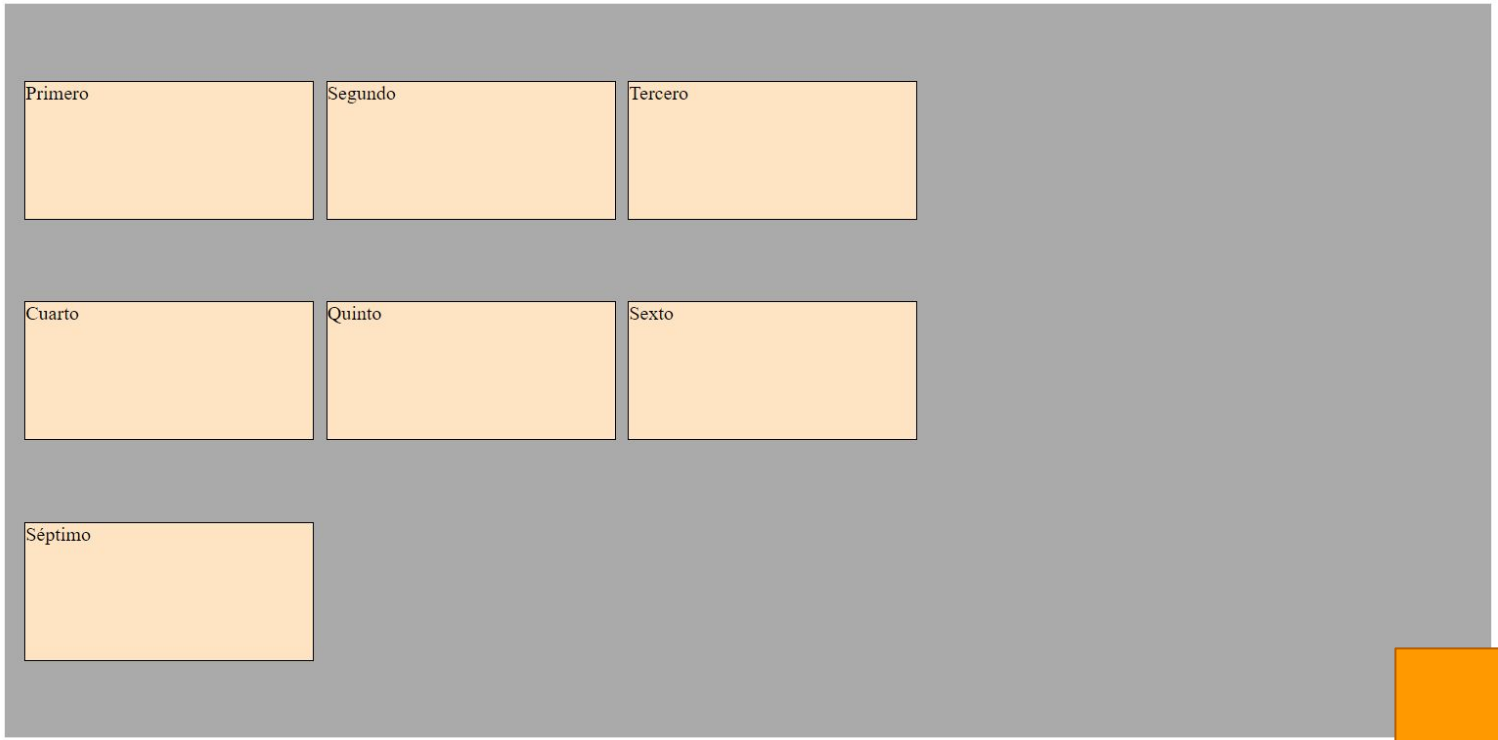
```
 /* Tres filas que se reparten toda la altura del contenedor (la que sea) */
    grid-template-rows: 20% 50% 30%;

  /* Cuatro filas. Tres de altura fija 100px y la otra ocupará el resto del espacio libre hasta llenar todo el contenedor en altura.*/
    grid-template-rows: 100px auto 100px 100px;

      /* Cuatro filas que se reparten de manera equitativa el alto del contenedor */
    grid-template-rows auto auto auto auto;

    /* Tres filas (todas con nombre, entre corchetes) Dos de ellas con tamaño fijo y la restante ocupará todo el alto libre. */
    grid-template-rows: [uno] 100px [dos] 300px [tres] auto;
```

MAQUETACIÓN GRID. Definición de la estructura del GRID

En estas dos propiedades también puedo repetir valores y usar la unidad fr que me sirve para establecer ratios para que los elementos se repartan el espacio restante. Podemos verlo mejor con un par de ejemplos.

```
/* Cuatro columnas. Tres de 20% con nombre col-start. Y la último que ocupará el resto del espacio libre */
grid-template-columns: repeat(3, 20% ) auto;
/* Cuatro columnas. Una de tamaño fijo y las demás se reparten el espacio libre en 5 partes de la siguiente manera (2+1+2) */
grid-template-columns: 2fr 100px 1fr 2rf;
```

MAQUETACIÓN GRID. Definición de la estructura del GRID



Ver

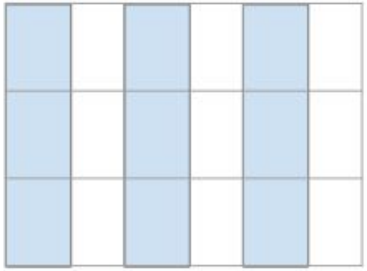
MAQUETACIÓN GRID. Alineación Horizontal

Por defecto los elementos del GRID ocupan todo el ancho de la celda que le corresponde pero podemos optar por otro tipo de alineaciones horizontales dando valores a la propiedad **justify-items**. Los diferentes valores que puede tomar son los siguientes:

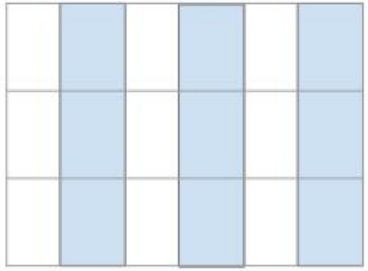
- start
- end
- center
- stretch que es la opción por defecto.

MAQUETACIÓN GRID. Alineación Horizontal

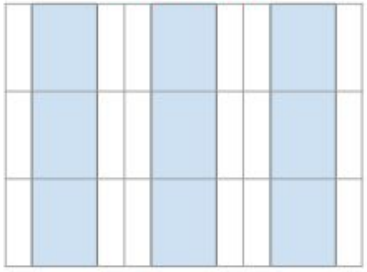
start



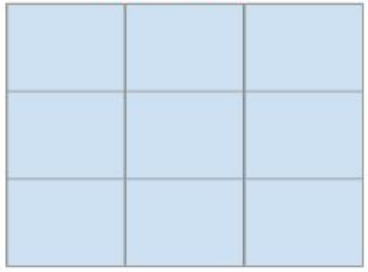
end



center



stretch



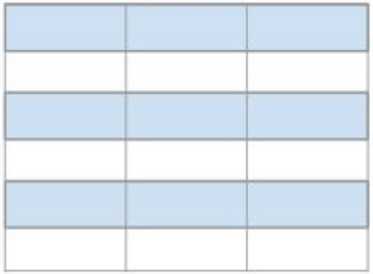
MAQUETACIÓN GRID. Alineación Vertical

Por defecto los elementos del GRID ocupan todo el alto de la celda que le corresponde pero podemos optar por otro tipo de alineaciones verticales dando valores a la propiedad **align-items**. Los diferentes valores que puede tomar son los siguientes:

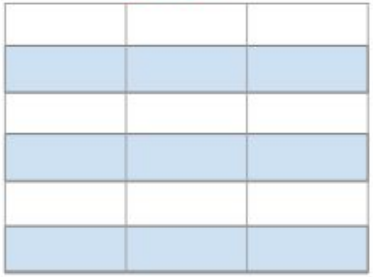
- start
- end
- center
- stretch que es la opción por defecto.

MAQUETACIÓN GRID. Alineación Horizontal

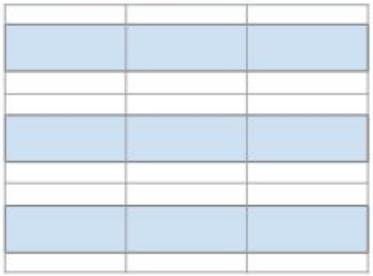
start



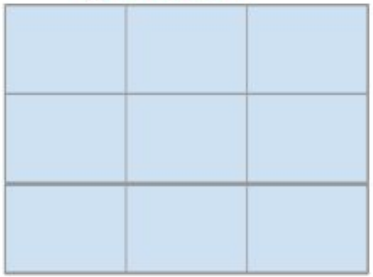
end



center



stretch

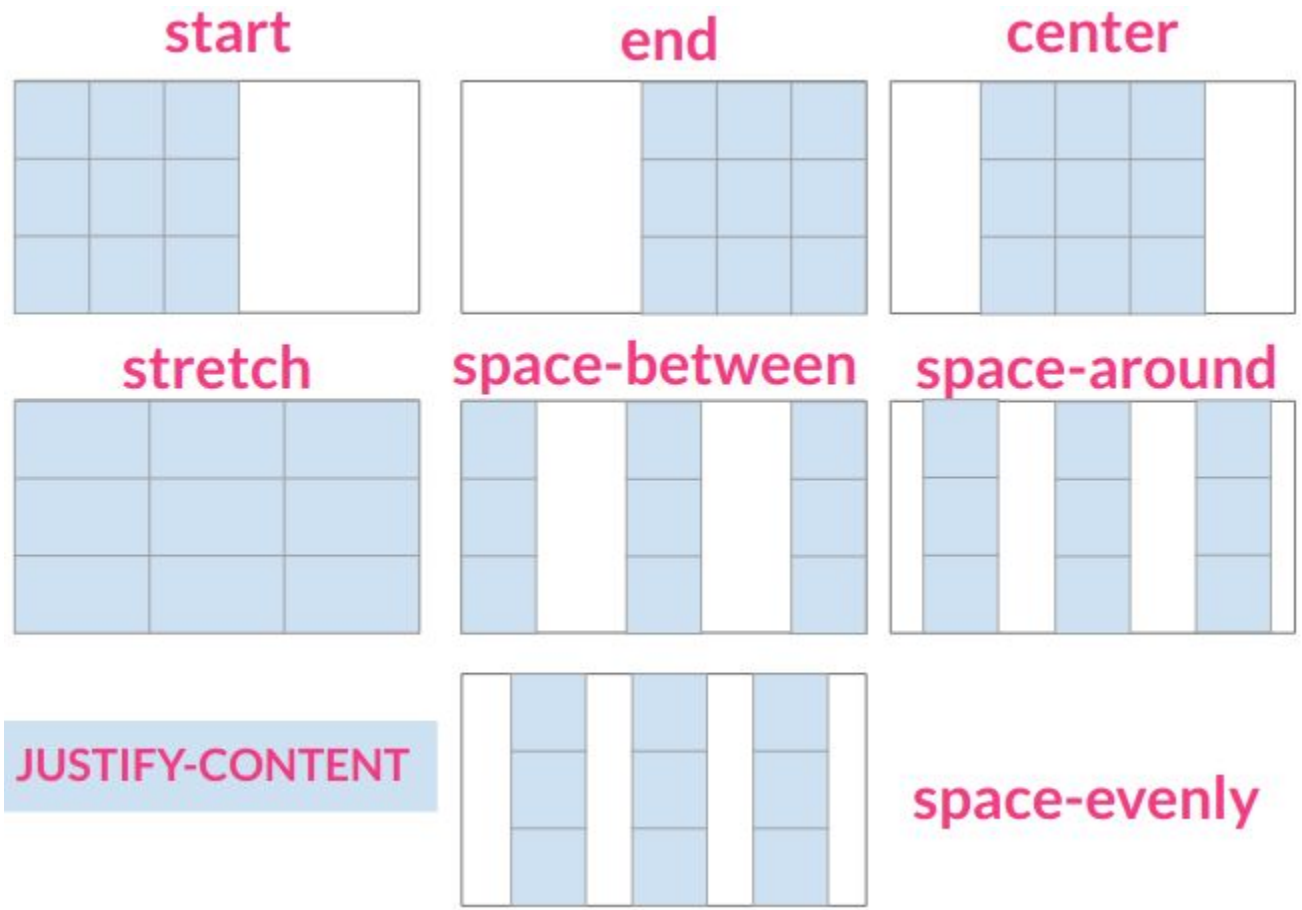


MAQUETACIÓN GRID. Distribución dentro del contenedor

En determinados casos puede suceder que los elementos del GRID no ocupen todo el ancho o todo el alto del contenedor GRID. En estas ocasiones puedo distribuir las columnas y las filas usando las propiedades **justify-content** (horizontal) y **align-content** (vertical). Ambas puede tomar los mismos valores y para entender mejor esos valores y cómo funcionan vamos a presentar dos imágenes.

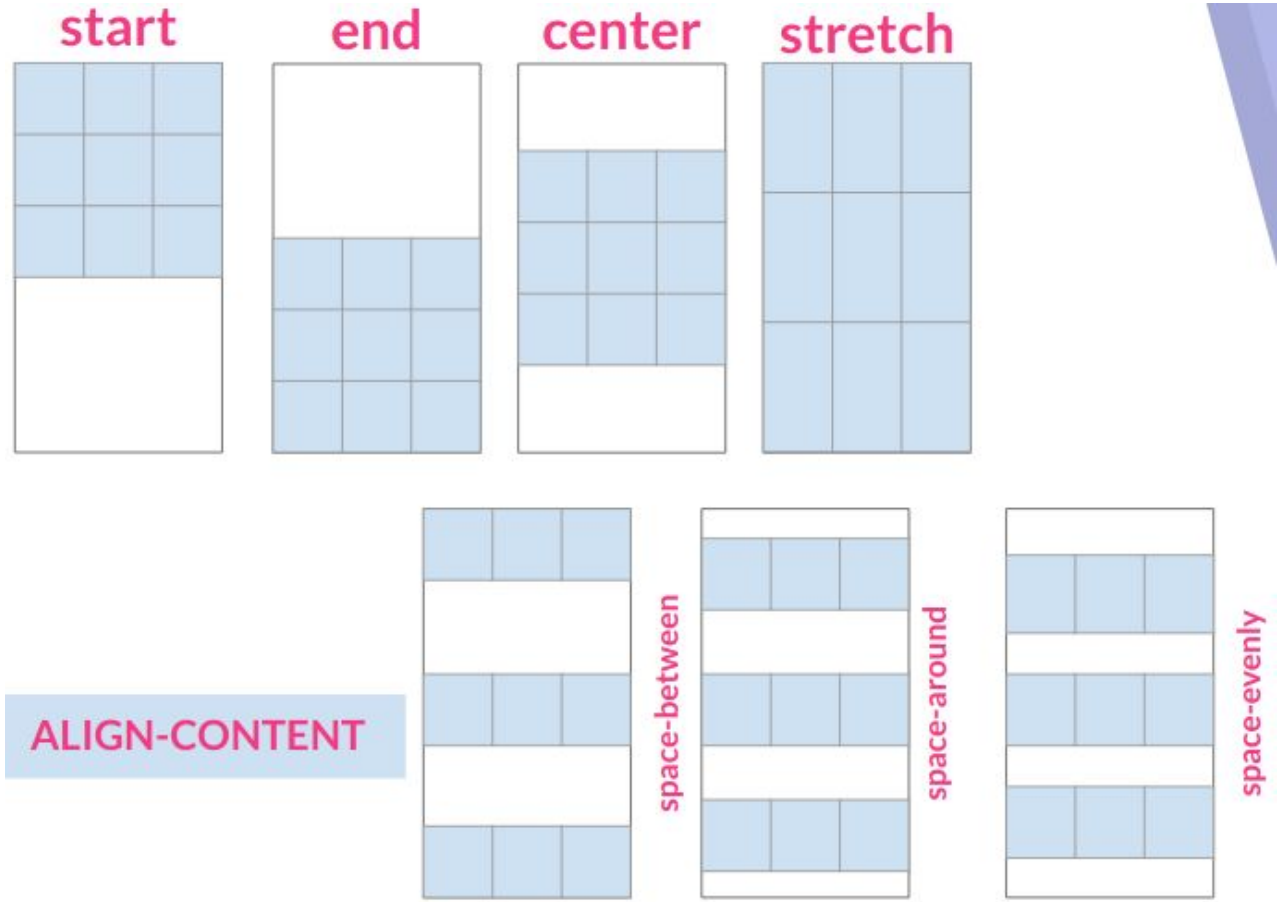
MAQUETACIÓN GRID. Distribución dentro del contenedor

Para la propiedad justify-content:



MAQUETACIÓN GRID. Distribución dentro del contenedor

Para la propiedad align-content:



MAQUETACIÓN GRID. Distribución dentro del contenedor

Si quiero juntar estas dos últimas alineaciones usaré la propiedad **place-content** indicando primero el valor para align-content y después el valor para justify-content.

MAQUETACIÓN GRID. Área del elemento GRID

Para especificar el área que va a ocupar el elemento GRID lo haremos con las siguiente propiedades:

grid-column-start

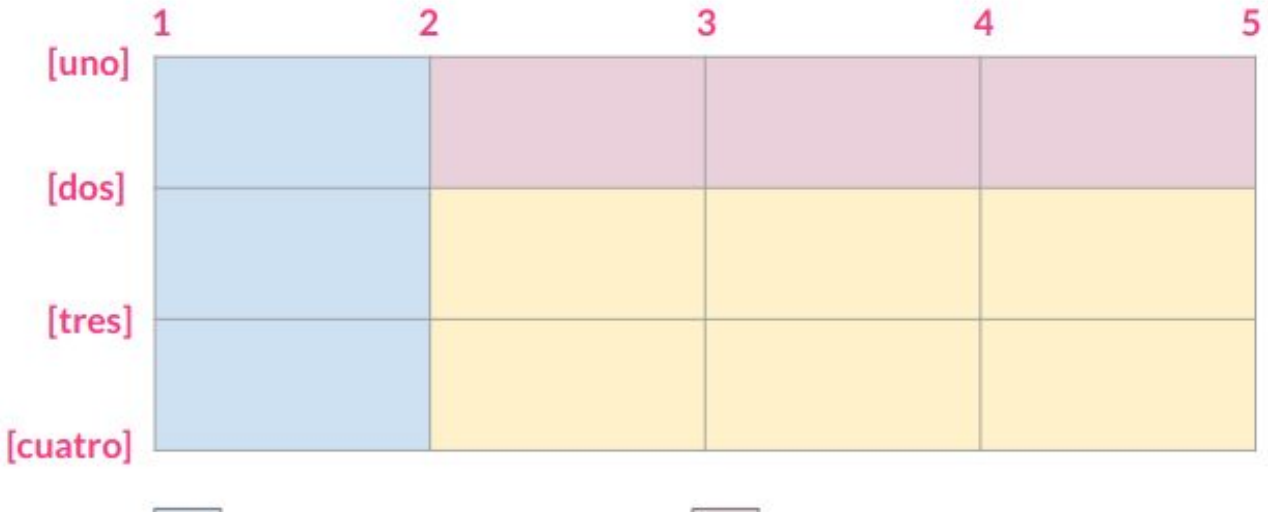
grid-column-end

grid-row-start

grid-row-end

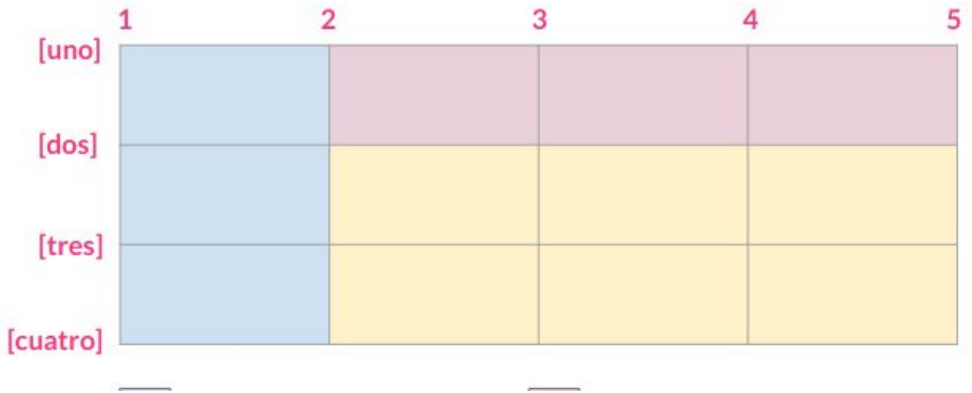
MAQUETACIÓN GRID. Área del elemento GRID

Para definir el área de los elementos del GRID, usaremos el siguiente CSS:



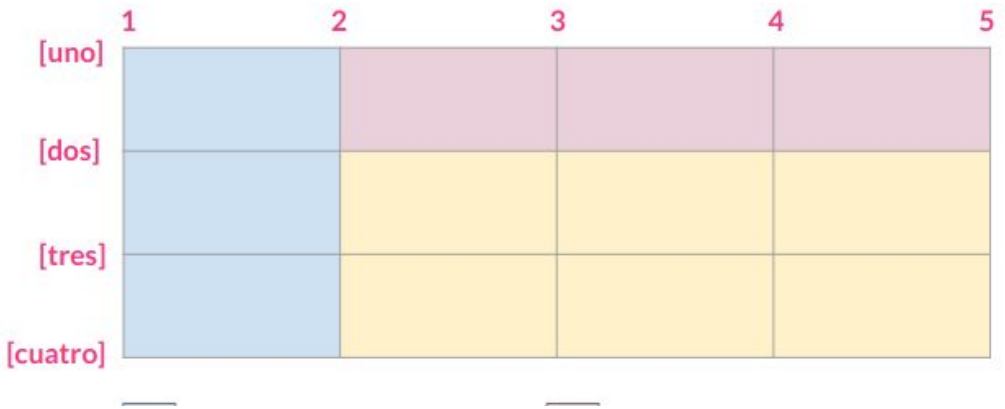
MAQUETACIÓN GRID. Área del elemento GRID

```
#azul {
  background-color: blue;
  grid-column-start: 1;
  grid-column-end: 2;
  grid-row-start: uno;
  grid-row-end: cuatro;
}
```



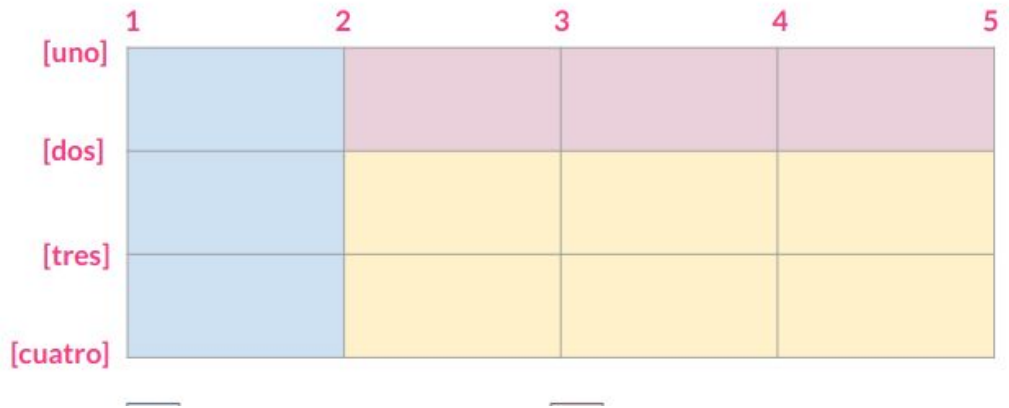
MAQUETACIÓN GRID. Área del elemento GRID

```
#rojo {
  background-color: red;
  grid-column-start: 2;
  grid-column-end: 5;
  grid-row-start: uno;
  grid-row-end: dos;
}
```



MAQUETACIÓN GRID. Área del elemento GRID

```
#amarillo {
  background-color: yellow;
  grid-column-start: 2;
  grid-column-end: 5;
  grid-row-start: dos;
  grid-row-end: span 2;
}
```



MAQUETACIÓN GRID. Área del elemento GRID

Se puede especifiar el área que ocupa de tres maneras principalemente:

- Indicando el número de línea donde empieza y donde acaba.
- Indicando con nombres de líneas, que habremos definido al definir el contenedor, donde empieza y donde acaba.
- Indicando cuánto ocupa en la dirección en cuestión (fila o columna) y usando span y el valor de la extensión.

MAQUETACIÓN GRID. Área del elemento GRID

Podemos juntar estas propiedades:

```
/* Para juntar las dos propiedades referentes a columnas */

grid-column: start / end;


/* Para juntar las dos propiedades referentes a filas */

grid-row: start / end;


/* Para junta las cuatro propiedades que nos permiten definir el área */

grid: row-start column-start row-end column-end;
```

MAQUETACIÓN GRID. Alineación horizontal.

La alineación horizontal de un elemento de manera individual se consigue usando la propiedad **justify-self** que puede tomar los mismos valores y funciona igual que la propiedad justify-items que dábamos al contenedor GRID. Estos valores son:

- start
- end
- center
- stretch (por defecto)

MAQUETACIÓN GRID. Alineación Vertical

La alineación vertical de un elemento de manera individual se consigue usando la propiedad **align-self** que puede tomar los mismos valores y funciona igual que la propiedad align-items que dábamos al contenedor GRID. Estos valores son:

- start
- end
- center
- stretch (por defecto

MAQUETACIÓN GRID. Colocación Implícita

En el caso de que no hayamos especificado el área que le corresponde a un elemento de GRID podemos establecer ciertas reglas del comportamiento mediante la propiedad grid-auto-flow que añadiremos al contenedor. Esta propiedad puede tomar varios valores:

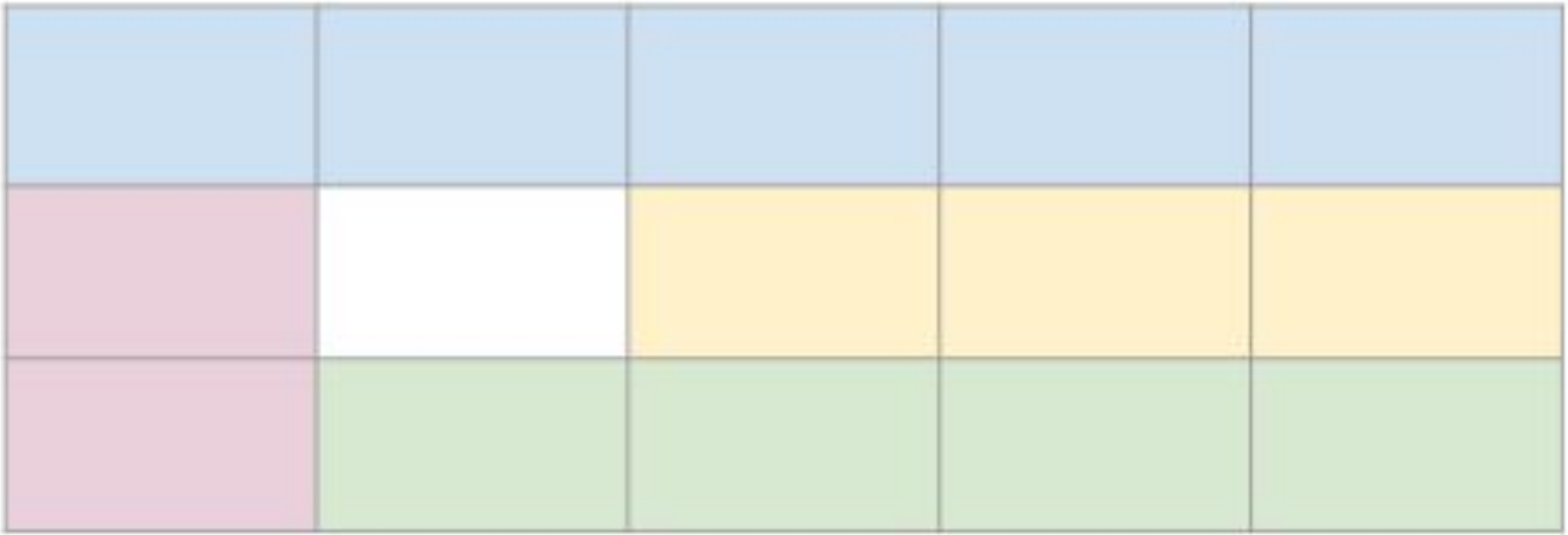
- row: Rellena primero las filas. Es la opción por defecto.
- column: Rellena primero las columnas.

MAQUETACIÓN GRID. Ejemplo

Vamos a ver como podemos maquetar únicamente nombrando áreas. Para ello usaremos las siguiente propiedades:

- **grid-area en los elementos GRID**
- **gird-template-area en el contenedor GRID.**

MAQUETACIÓN GRID. Ejemplo



MAQUETACIÓN GRID. Ejemplo

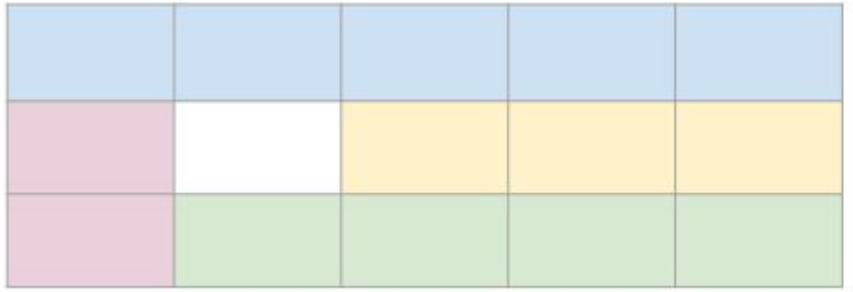
Creamos un contenedor general

```
.container {
  grid-template-columns: repeat(5, 20%);
  grid-template-rows: repeat(3, 100px);
}
```

MAQUETACIÓN GRID. Ejemplo

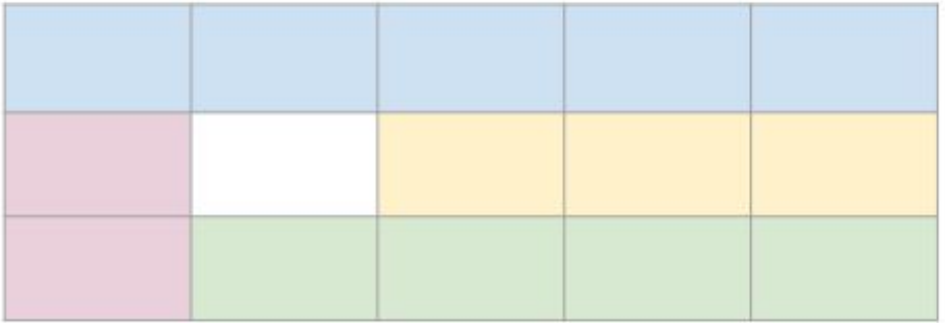
Daremos nombre al área que va a ocupar cada elemento del grid mediante la propiedad grid-area:

```
#cab {
  background-color: blue;
  grid-area: cab;
}
```



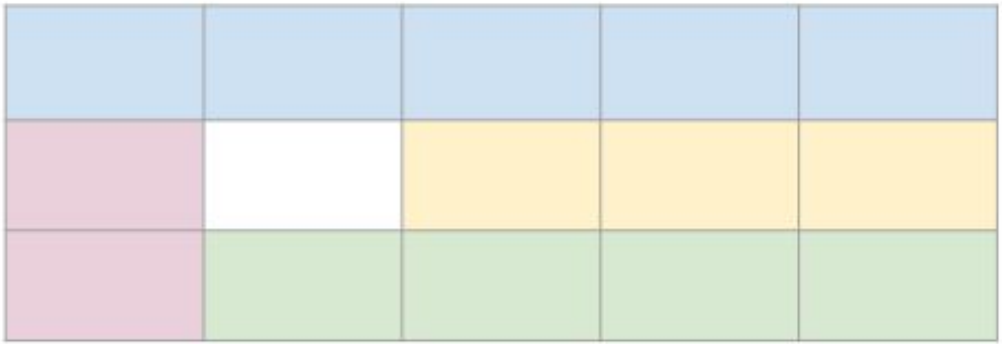
MAQUETACIÓN GRID. Ejemplo

```
#pie {
  background-color: green;
  grid-area: pie;
}
```

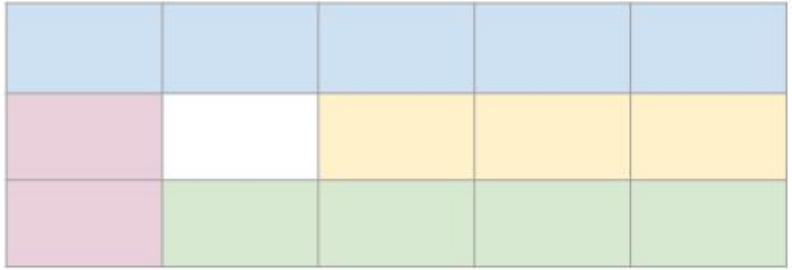


MAQUETACIÓN GRID. Ejemplo

```
#menu {
  background-color: red;
  grid-area: menu;
}

#principal {
  background-color: yellow;
  grid-area: main;
}
```



MAQUETACIÓN GRID. Ejemplo



```
.container {
  display: grid;
  grid-template-columns: repeat(5, 20%);
  grid-template-rows: repeat(3, 100px);
  grid-template-areas:
    "cab cab cab cab cab"
    "menu  .  main main main"
    "menu  pie pie pie pie";
}
```

Ejemplo de Maquetación con Grid

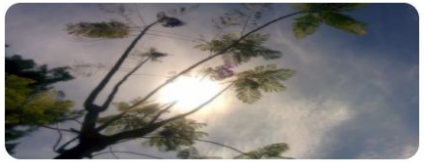
By @pekechis para @openwebinars

- Inicio
- Galería
- Productos
- Clientes
- Sobre nosotros
- Contacto

Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure harum dolore expedita recusandae omnis obcaecati praesentium numquam, labore ad libero placeat saepe nemo possimus. Eligendi, magni culpa? Voluptate, eligendi eum.

Una image bonita de un cielo

Una image bonita de un cielo



Una image bonita de un cielo

Una image bonita de un cielo

C/ Principal
Sevilla

1 2 3 4 5

2

Ejemplo de Maquetación con Grid

By @pekechis para @openwebinars

3

- Inicio
- Galería
- Productos
- Clientes
- Sobre nosotros
- Contacto

Lorem ipsum dolor sit amet consectetur adipisicing elit. Iure harum dolore expedita recusandae omnis obcaecati praesentium numquam, labore ad libero placeat saepe nemo possimus. Eligendi, magni culpa? Voluptate, eligendi eum.

Una image bonita de un cielo

4

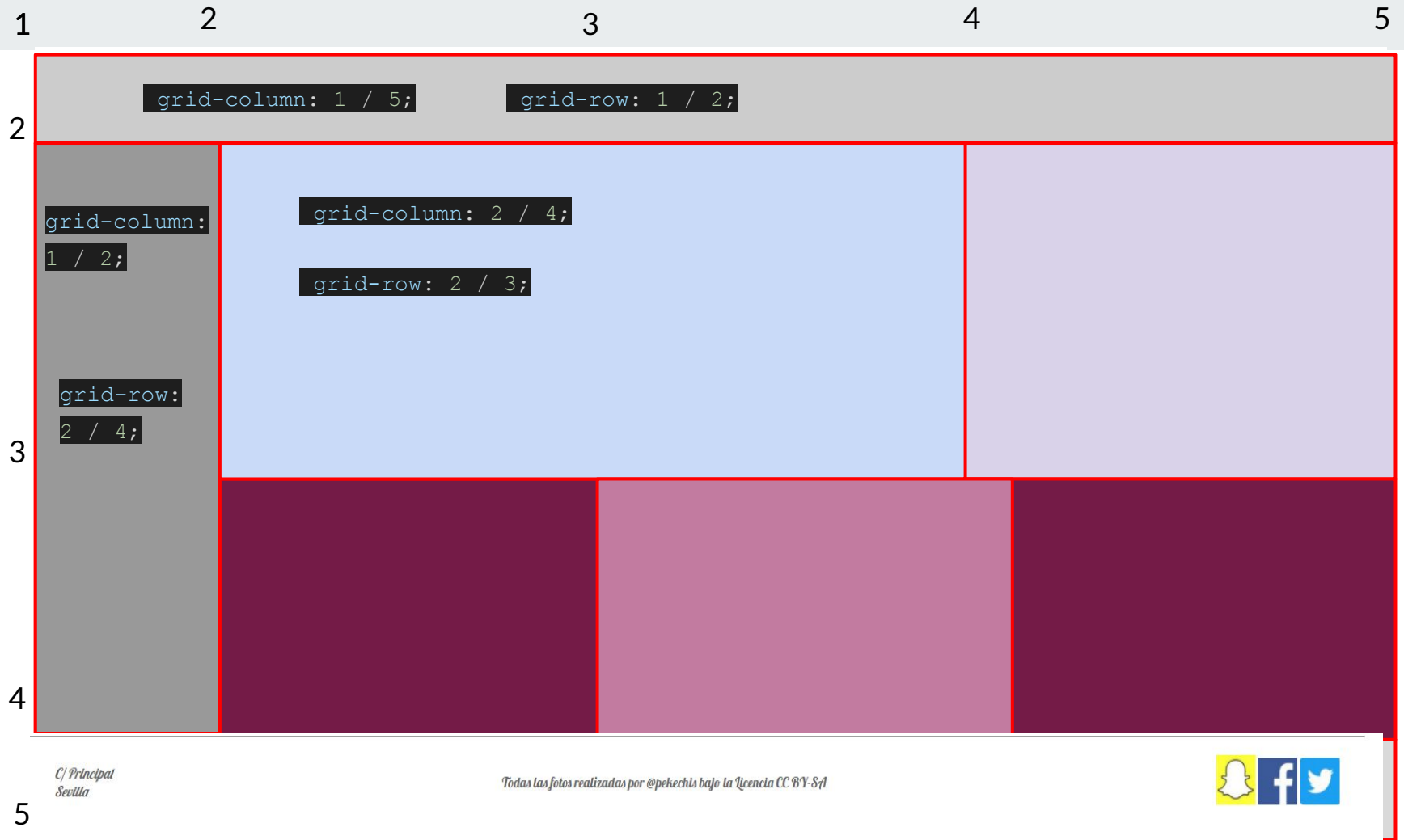
Una image bonita de un cielo

Una image bonita de un cielo

Una image bonita de un cielo

5

C/ Principal
Sevilla


1
2
3
4
5
grid-column: 1 / 5;
grid-row: 1 / 2;
grid-column: 1 / 2;
grid-row: 2 / 4;
grid-column: 2 / 4;
grid-row: 2 / 3;
C/ Principat
Sevilla
Todas las fotos realizadas por @pekechis bajo la Licencia CC BY-SA