UT02.2 CSS Avanzado

01 de Septiembre de 2017

I.E.S. Virgen de la Paz

Grid-View

- Muchas páginas están basadas en Grid-View, esto significa que está dividida en columnas
- Es muy útil a la hora de diseñar una página web, hace mucho más fácil colocar elementos en la web

BY NC ND

960 Grid System

- 960 Grid System es un sistema de rejilla para hacer páginas con 960 píxeles de ancho
- Las columnas que podremos colocar en la rejilla tendrán distintas anchuras, pero siempre el ancho total de la página será de 960 píxeles
- Se eligió 960 al ser divisible por muchos números haciéndolo más versátil para poder alcanzar páginas resultantes de la más variada gama.
- Para hacer páginas web responsive se actualizó con el nuevo nombre de unsemantic

CC BY NC ND

960 Grid System

Contenedor de columnas

- Podemos utilizar un modelo de 12, 16 o 24 columnas siendo la primera la más común
- Todo el contenido de la página debe ir en un contenedor principal
- Este contenedor tiene el nombre de class CSS "container_xx", donde xx es el número de columnas que vamos a utilizar

```
<div class="container_12">
...
</div>
```

- Dentro del container colocaremos los elementos de la página ya con la estructura de rejilla

cc BY NC ND

Elemento de la rejilla

- Cada elemento a colocar en el contenedor tiene una clase como "grid_x", siendo x el número de la clase que tiene la anchura deseada

1: 60px	7: 540px
2: 140px	8: 620px
3: 220px	9: 700px
4: 300px	10: 780px
5: 380px	11: 860px
6: 460px	12: 940px

CC BY NC ND

Elemento de la rejilla

- Si queremos que un elemento utilice una columna con toda la anchura disponible utilizaríamos este código:

```
<div class="container_12">
   <div class="grid_12">
      <p>940 px</p>
   </div>
   <div class="clear"></div>
</div>
```

- Después de cada fila de elementos, hay que colocar un div con class="clear", que elimina posibles problemas de float que pudiéramos tener en el diseño con columnas.

cc BY NC ND

Elemento de la rejilla

- Ejemplo con una fila con dos columnas de 460 px de ancho

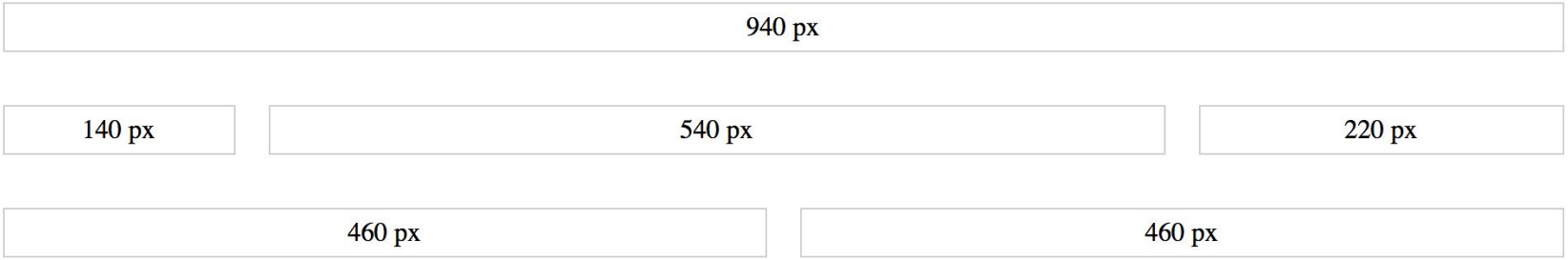
```
<div class="container_12">
   <div class="grid_6">
      <p>460 px</p>
   </div>
   <div class="grid_6">
      <p>460 px</p>
   </div>
   <div class="clear"></div>
</div>
```

cc BY NC ND

960 Grid System

Elemento de la rejilla

- https://codepen.io/carlostessier-1472587534/pen/ZXoqNo



CC BY NC ND

960 Grid System

Clases

- **Class container_xx**: Esta clase sirve para definir un contenedor, donde colocaremos luego todos los contenidos de nuestro diseño
- **Class grid_xx:** Define un elemento del diseño que será colocado en un contenedor y ajustado a la rejilla. El valor "xx", de grid_xx, expresa el tamaño de la rejilla que se está definiendo. A un número mayor, más anchura se dedicará a este elemento
- **Class alpha y class omega** Estas dos clases sirven cuando estamos anidando unos grid dentro de otros, para ajustar los márgenes de los distintos elementos anidados.

cc BY NC ND

960 Grid System

Clases

- **Class prefix_xx:** Esta clase sirve para anclar una grid_xx dejando un espacio vacío a la izquierda

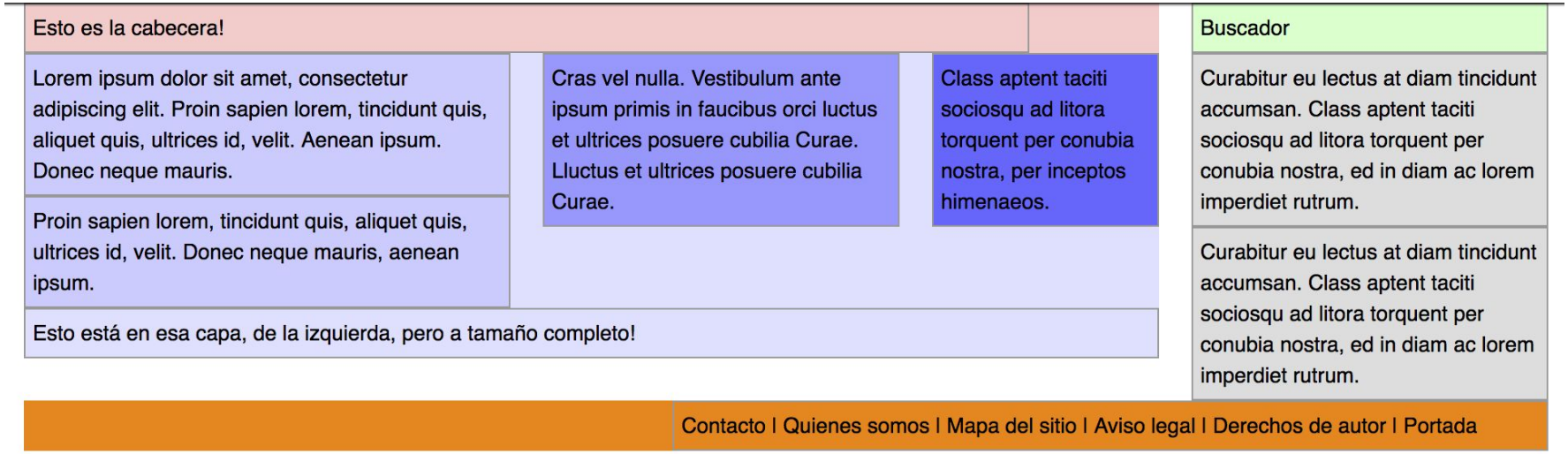
- **Class suffix_xx:** la clase sufix_xx sirve para colocar un espacio vacío a la derecha de la capa.

CC BY NC ND

960 Grid System

Clases

https://codepen.io/carlostessier-1472587534/full/NaMEba/



cc BY NC ND

Flexbox

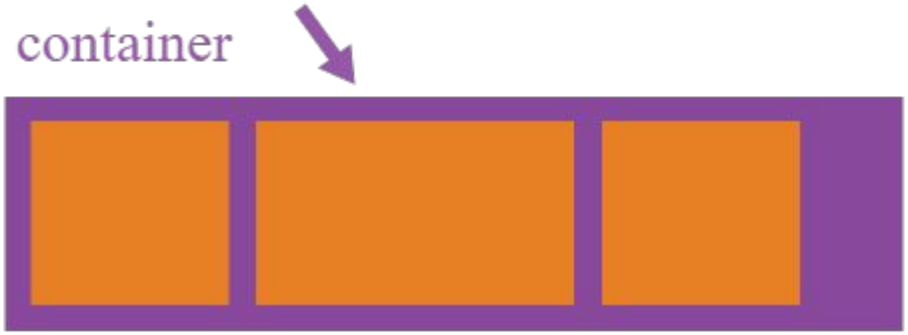
- El módulo Flexbox Layout es una herramienta CSS que hace muy fácil la creación de **diseños líquidos** para **apps** y **pequeñas webs**
- Esta nueva herramienta permite apilar los elementos de tu página en columnas y filas de forma flexible
- Su objetivo principal es dar la capacidad al contenedor de alterar el ancho, alto y orden de sus elementos para que se adapte a diferentes tamaños de pantalla
- Aunque ya lleva tiempo con nosotros, ha tenido grandes revisiones, y aunque está soportado actualmente por iOS y Android deberíamos probar en diferentes navegadores
- Para páginas web más complejas existe el layout Grid

CC BY NC ND

Contenedor Flex

display

- Define un contenedor flex



- Puede ser de bloque o en línea

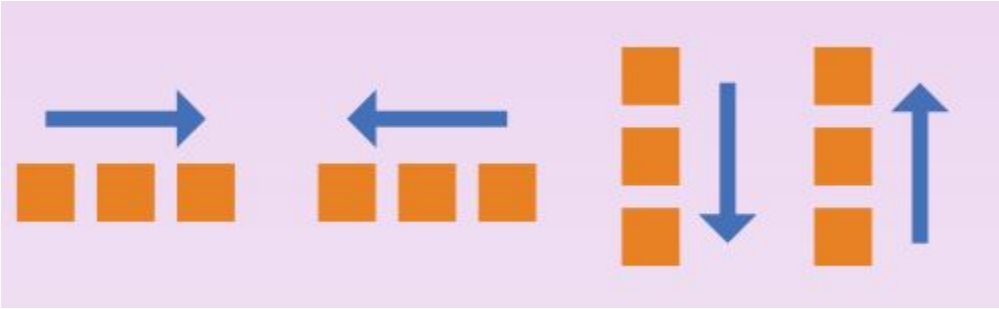
```css
.container {
  display: flex; /* or inline-flex */
}
```

CC BY NC ND

Contenedor Flex

flex-direction

- Define la dirección de los elementos de un contenedor



- `flex-direction: row | row-reverse | column | column-reverse` :

CC BY NC ND

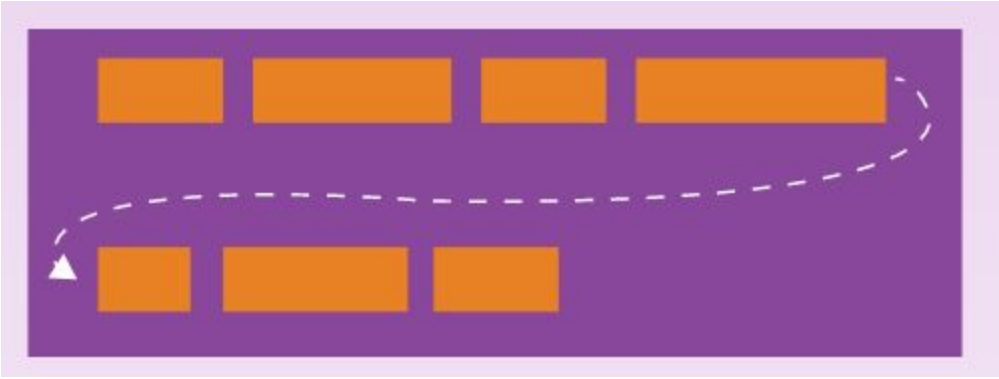
Contenedor Flex

flex-wrap

- Por defecto los elementos del contenedor intentarán caber en una sola línea.

- flex-wrap permite que los elementos se adapten en varias líneas

```
.container {
    flex-wrap: nowrap | wrap | wrap-reverse
}
```



CC BY NC ND

Contenedor Flex

flex-flow

- Es una forma corta de utilizar flex-direction y flex-wrap que definen en conjunto los ejes horizontal y vertical del contenedor
- Por defecto está como row nowrap.

```
.container {
    flex-flow:
        <‘flex-direction’> ||  <‘flex-wrap’>
}
```

CC BY NC ND

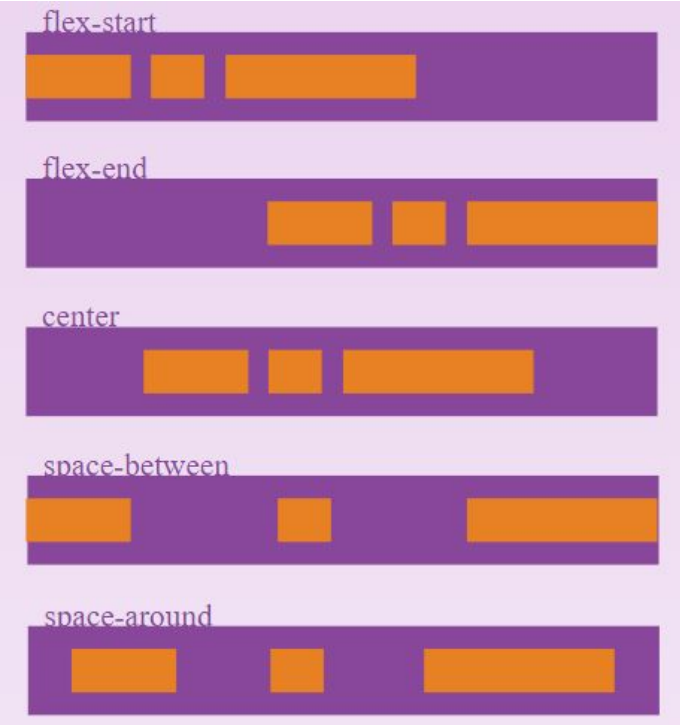
Contenedor Flex

justify-content

- Define la alineación en el eje principal

```
.container {
justify-content:
flex-start | flex-end |
center | space-between |
space-around
}
```

- Esto ayuda a distribuir el espacio extra en cada elemento flex si tienen tamaño fijo o han alcanzado su tamaño máximo



CC BY NC ND

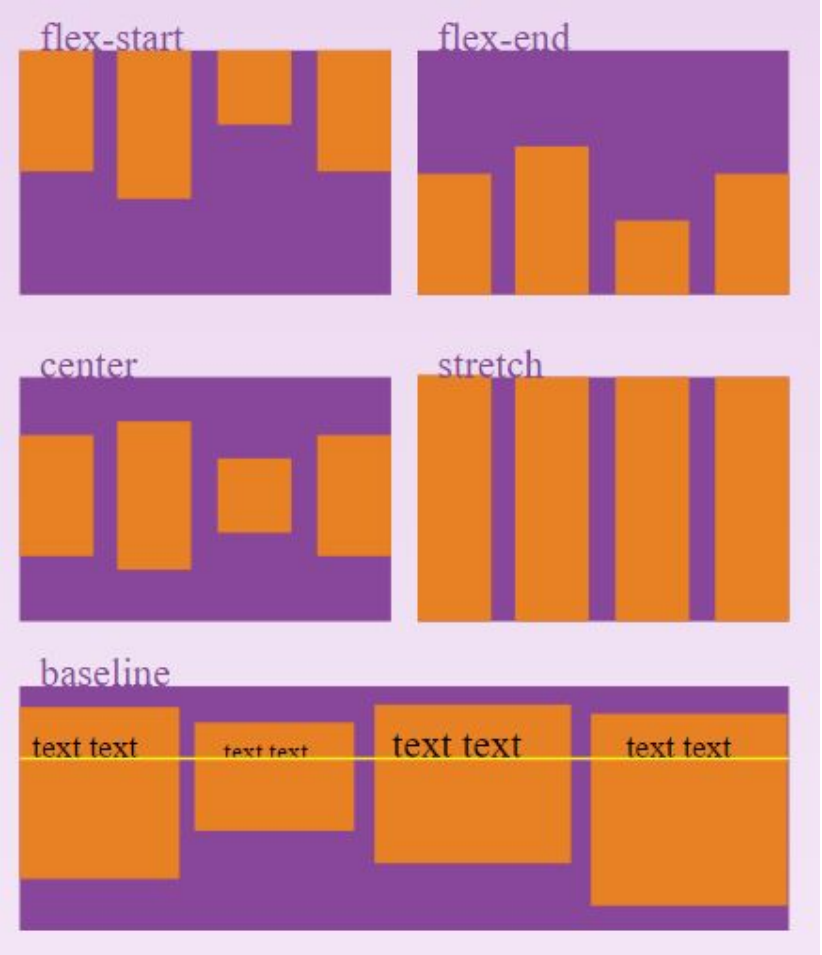
Contenedor Flex

align-items

- Define el comportamiento de los elementos flex en el eje transversal

```
.container {
align-items: flex-start | flex-end | center | baseline | stretch
}
```

- Es como la versión vertical de `justify-content`



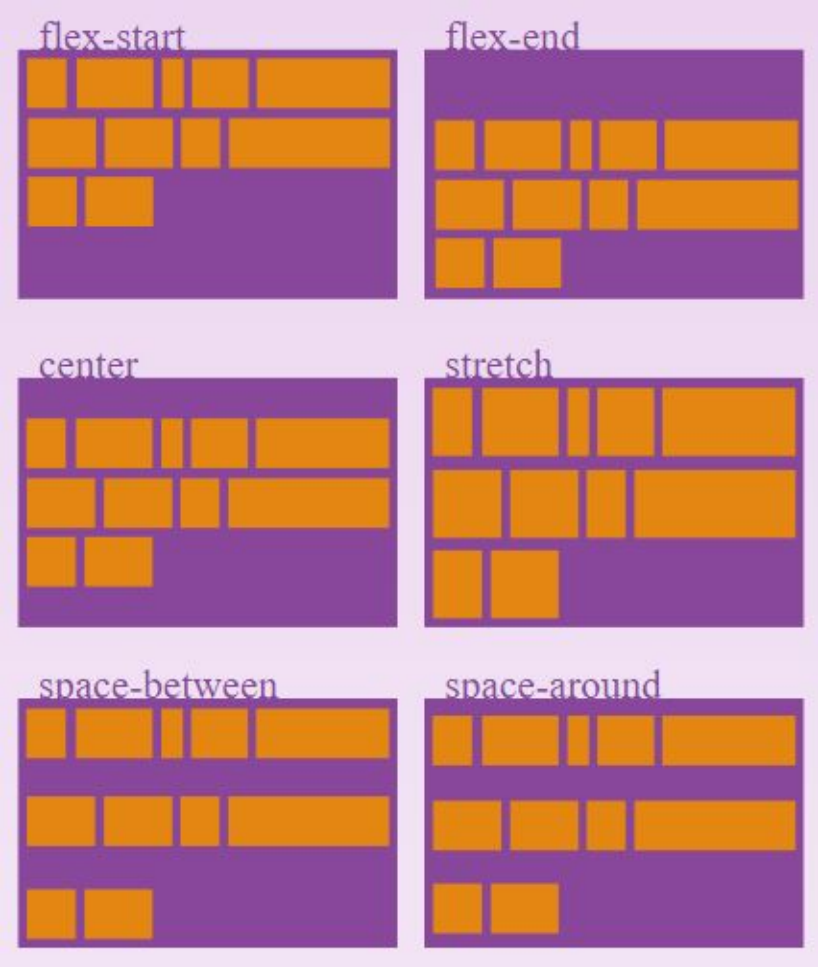
CC BY NC ND

Contenedor Flex

align-content

- Define el comportamiento de las líneas en el eje transversal

```
.container {
align-content:
flex-start | flex-end |
center | space-between
| space-around |
stretch
}
```

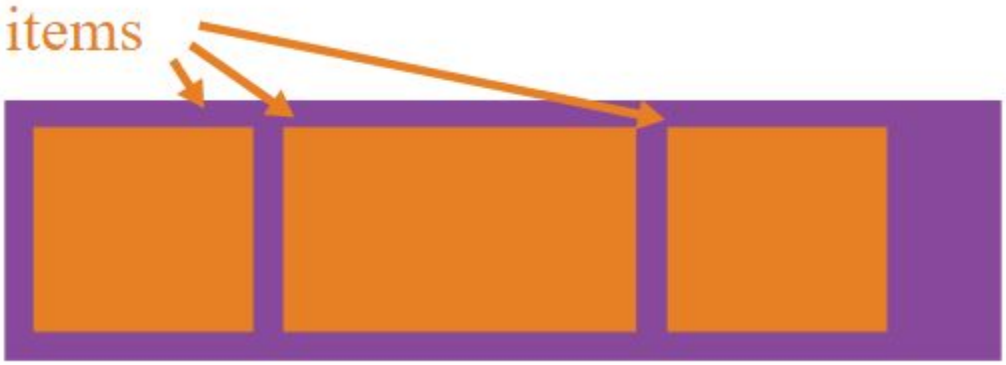


CC BY NC ND

Elementos flex

order

- Por defecto los elementos flex se posicionan en el orden en el que aparecen



- Con la propiedad order podemos controlar cómo aparecen en el contenedor

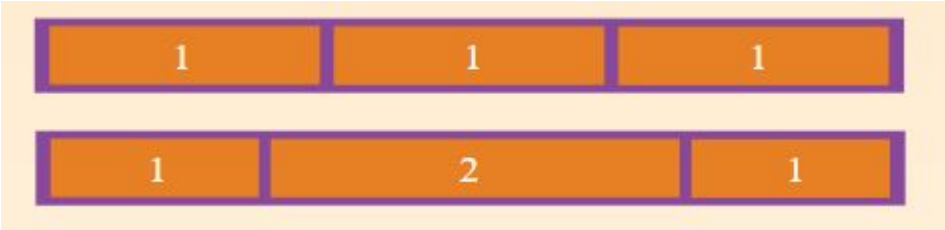
```css
.item {
  order: <integer>;
}
```

CC BY NC ND

Elementos flex

flex-grow

- Define la habilidad de un elemento flex para crecer lo necesario
- Admite una unidad que sirve como proporción
- Todos los elementos tienen por defecto fijado el valor a 1, el espacio disponible del contenedor se repartirá de forma equitativa a todos los hijos
- Si un elemento tiene valor dos, este elemento tendrá el doble de espacio disponible que el resto de hijos



CC BY NC ND

Elementos flex

flex-shrink

- Define la habilidad de un elemento flex para encoger
- No admite valores negativos

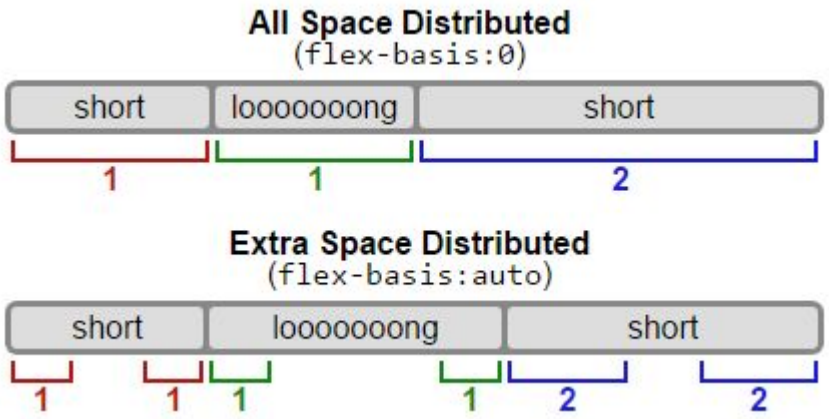
```css
.item {
  flex-shrink: <number>; /* default 1 */
}
```

CC BY NC ND

Elementos flex

flex-basis

- Define el tamaño por defecto de un elemento antes de que se distribuya el espacio
- Puede ser:
  - Una longitud: 20% 5em, etc.
  - Una palabra clave: auto, content, etc.
- Está fijado por defecto como auto, teniendo en cuenta el contenido para distribuir el espacio
- Si lo fijamos como 0, se distribuye sin importar el contenido.



CC BY NC ND

Elementos flex

flex

- Es la forma corta de las propiedades flex-grow, flex-shrink y flex-basis combined.
- El segundo y tercer parámetro (flex-shrink y flex-basis) son opcionales

```
.item {
  flex: none | [ <'flex-grow'>
<'flex-shrink'>? || <'flex-basis'> ]
}
```

- Por defecto su valor es 0 1 auto.

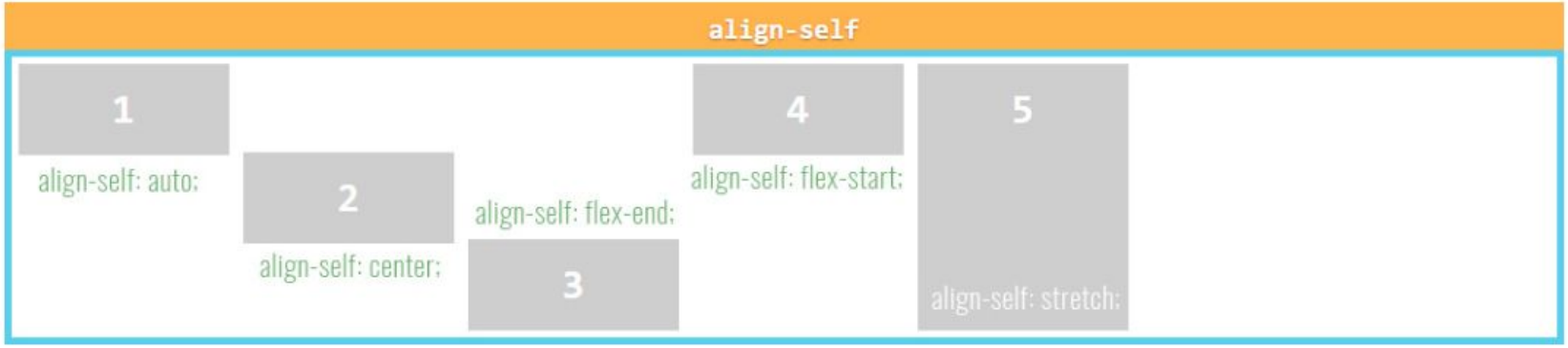
CC BY NC ND

Elementos flex

align-self

- Cambia el comportamiento por defecto de align-content del contenedor

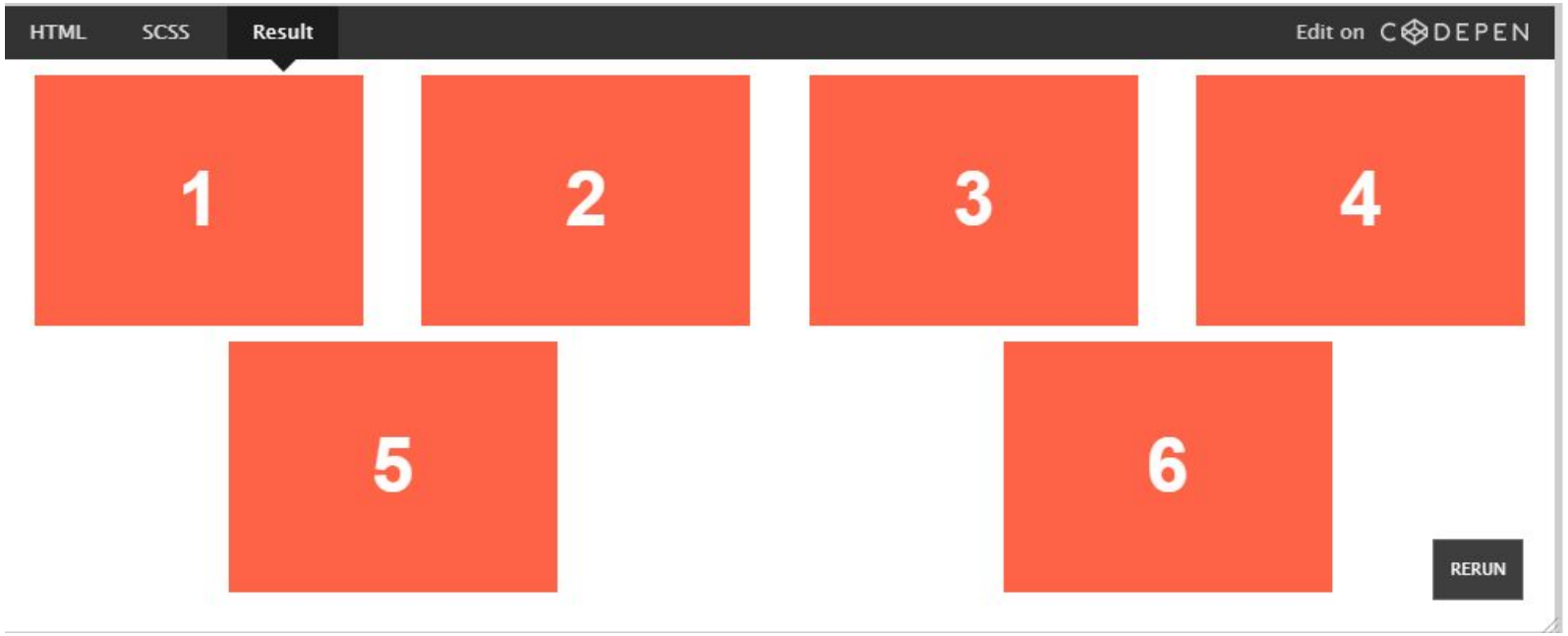
```
.item {
align-self: auto | flex-start | flex-end |
center | baseline | stretch;
}
```



CC BY NC ND

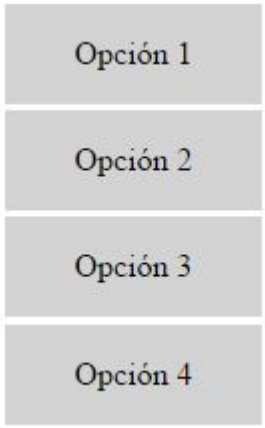
Ejemplo flexbox

https://codepen.io/carlostessier-1472587534/pen/WjMerR



CC BY NC ND

- Menú vertical
  http://codepen.io/carlostessier-1472587534/pen/XRzYaQ



BY NC ND

BY NC ND