UT03. CSS: PRESENTACIÓN Y VISIBILIDAD

Curso: 2DAW

Lenguajes de Marcas y sistemas de gestión de información

Presentación

- **display**:
  - Especifica cómo debe ser la caja del elemento.
  - Valores:
    - →**none**: no se crea caja. **No es visible el elemento**.
    - →**block**: se crea la caja en una nueva línea. El resto de los elementos quedan por encima o por debajo
    - →**inline**: se crea una caja en la misma línea y puede haber elementos inmediatamente a cada lado de este. **Valor por defecto**.
    - →**inline-block**: las cajas se muestran en línea y si no caben se muestran en la siguiente línea.
  - **Ejemplo40.css**: Observar qué pasa si ponemos en titulo-principal, **display: none;**

Display. div vs. span

- Piensa cómo sería el resultado de los siguientes códigos (cada HTML con cada CSS)

HTML

```html
<p>Lorem ipsum dolor...</p>
<span class="caja">CAJA 1</span>
<span class="caja">CAJA 2</span>
<span class="caja">CAJA 3</span>
<p>Lorem ipsum dolor...</p>
```

CSS

```css
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
}
```

CSS

```css
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
  display: inline;
}
```

HTML

```html
<p>Lorem ipsum dolor...</p>
<div class="caja">CAJA 1</div>
<div class="caja">CAJA 2</div>
<div class="caja">CAJA 3</div>
<p>Lorem ipsum dolor...</p>
```

CSS

```css
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
  display: inline-block;
}
```

CSS

```css
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
  display: block;
}
```

display

span

HTML

```
<p>Lorem ipsum dolor...</p>
<span class="caja">CAJA 1</span>
<span class="caja">CAJA 2</span>
<span class="caja">CAJA 3</span>
<p>Lorem ipsum dolor...</p>
```

CSS

```
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
}
```

Lorem ipsum dolor...

CAJA 1 CAJA 2 CAJA 3

Lorem ipsum dolor...

span es una etiqueta en línea y no respeta ni tamaños ni márgenes verticales

div

HTML

```
<p>Lorem ipsum dolor...</p>
<div class="caja">CAJA 1</div>
<div class="caja">CAJA 2</div>
<div class="caja">CAJA 3</div>
<p>Lorem ipsum dolor...</p>
```

CSS

```
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
}
```

Lorem ipsum dolor...

CAJA 1

CAJA 2

CAJA 3

Lorem ipsum dolor...

div es una etiqueta en bloque y sí respeta tamaños y márgenes verticales

display

div con display inline

HTML

```
<p>Lorem ipsum dolor...</p>
<div class="caja">CAJA 1</div>
<div class="caja">CAJA 2</div>
<div class="caja">CAJA 3</div>
<p>Lorem ipsum dolor...</p>
```

CSS

```
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
  display: inline;
}
```

Lorem ipsum dolor...

CAJA 1 CAJA 2 CAJA 3

Lorem ipsum dolor...

Mismo efecto que en span

display

div con display inline-block

HTML

```
<p>Lorem ipsum dolor...</p>
<div class="caja">CAJA 1</div>
<div class="caja">CAJA 2</div>
<div class="caja">CAJA 3</div>
<p>Lorem ipsum dolor...</p>
```

CSS

```
.caja {
  background-color: yellow;
  border: 1px solid black;
  width: 100px;
  height: 100px;
  padding: 5px;
  margin: 50px 10px;
  display: inline-block;
}
```

Lorem ipsum dolor...

CAJA 1 CAJA 2 CAJA 3

Lorem ipsum dolor...

Efecto mezcla de los dos. **inline,** respetando tamaños y márgenes

Presentación

- **display: table**
  - Ofrece la posibilidad de maquetar contenidos de manera que su comportamiento se parezca al de las tradicionales tablas del HTML
  - Asignar comportamiento visual a los elementos como si fueran celdas de las tablas tradicionales.
- **display: table-row**
  - Actúa como un elemento fila (TR)
- **display: table-cell**
  - Etiquetas que deben trabajar como una celda de tabla (TD).

Presentación

- **table-caption;** Se comportan como la etiqueta CAPTION.
- **table-column;** Se comportan como la etiqueta COL.
- **table-column-group;** Se comportan como la etiqueta COLGROUP.
- **table-footer-group;** Se comportan como la etiqueta TFOOT.
- **table-header-group;** Comportamiento como la etiqueta THEAD.
- **table-row-group;** Comportamiento como la etiqueta TBODY.
- **display: inline-table;** Igual que un **display: table** con comportamiento de un elemento "inline". Es decir, una tabla normal actúa como un bloque y con este elemento puedes hacer que se comporte como un elemento "inline“.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta content="text/html; charset=windows-1252" http-equiv="content-type">
    <title>display:table</title>
  </head>
  <body>
    <div class="tablagen">
      <div class="fila">
        <div class="col">fila1.columna1</div>
        <div class="col">fila1.columna2</div>
        <div class="col">fila1.columna3</div>
        <div class="col">fila1.columna4</div>
      </div>
      <div class="fila">
        <div class="col">fila2.columna1</div>
        <div class="col">fila2.columna2</div>
        <div class="col">fila2.columna3</div>
      </div>
      <div class="fila">
        <div class="col">fila3.columna1</div>
        <div class="col">fila3.columna2</div>
      </div>
    </div>

  </body>
</html>

```

Los elementos div actúan como block ➔ se visualiza cada div en una línea

fila1.columna1
fila1.columna2
fila1.columna3
fila1.columna4
fila2.columna1
fila2.columna2
fila2.columna3
fila3.columna1
fila3.columna2

Presentación

- Añadimos **<style>**

```
6       <style>
7         .tablagen{
8           display: table;
9         }
10
11        .fila{
12          display: table-row;
13        }
14        .col{
15          display: table-cell;
16          padding: 12px;
17          background: #ddd;
18        }
19
20      </style>
```

fila1.columna1	fila1.columna2	fila1.columna3	fila1.columna4
fila2.columna1	fila2.columna2	fila2.columna3	
fila3.columna1	fila3.columna2		

Presentación

- Para conseguir un espacio entre celdas
  - No **cellspacing**
  - Sí **border-spacing**

Presentación

- **CSS Flexible Box Layout** o "Diseño con cajas Flexibles" intenta cambiar la forma de diseñar y generar contenidos totalmente adaptables a cualquier dispositivo.
- Elementos
  - **flex container** .- los contenidos que son hijos directos se comportaran de manera totalmente flexible
  - **flex items .-** para poder cambiar sus posiciones, orden alineamiento y otras propiedades más.
- Propiedades
  - **display** (flex/inline-flex)
  - **flex-direction** (row, row-reverse, column, column-reverse)
  - **flex-wrap** (no-wrap, wrap, wrap-reverse)
  - **justify-content** (space-between, flex-end, flex-start, center, space-around)
  - **align-ítem** (flex-start, flex-end, center, strech, baseline)
  - **align-self** (flex-start, flex-end, center, strech, baseline)
  - **order**
  - **flex-grow**
  - **flex-shrink**

Presentación

- Establecer el **contenedor** Flexible o **Flex Container** para que todos los contenidos interiores, los hijos directos, se comporten flexiblemente.

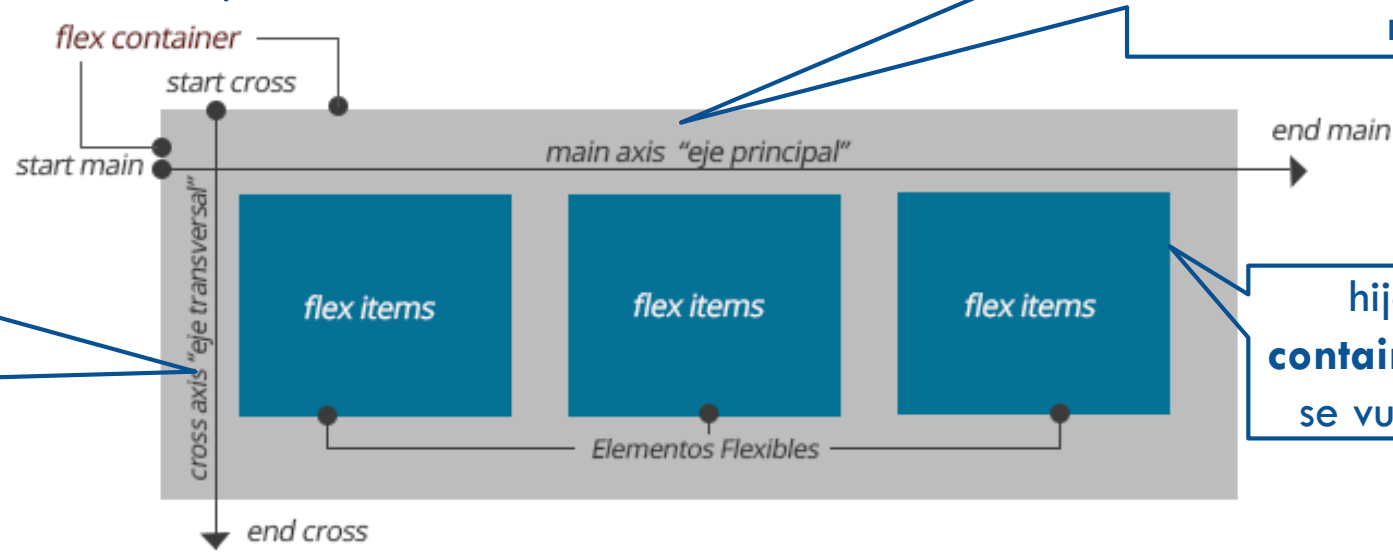
display: flex;

contenedor principal flexible

eje principal que atraviesa el elemento **flex container** horizontalmente de izquierda-derecha (**start main** a **end main**)

eje principal que atraviesa el elemento **flex container** verticalmente de arriba-abajo (**start cross** a **end cross**)

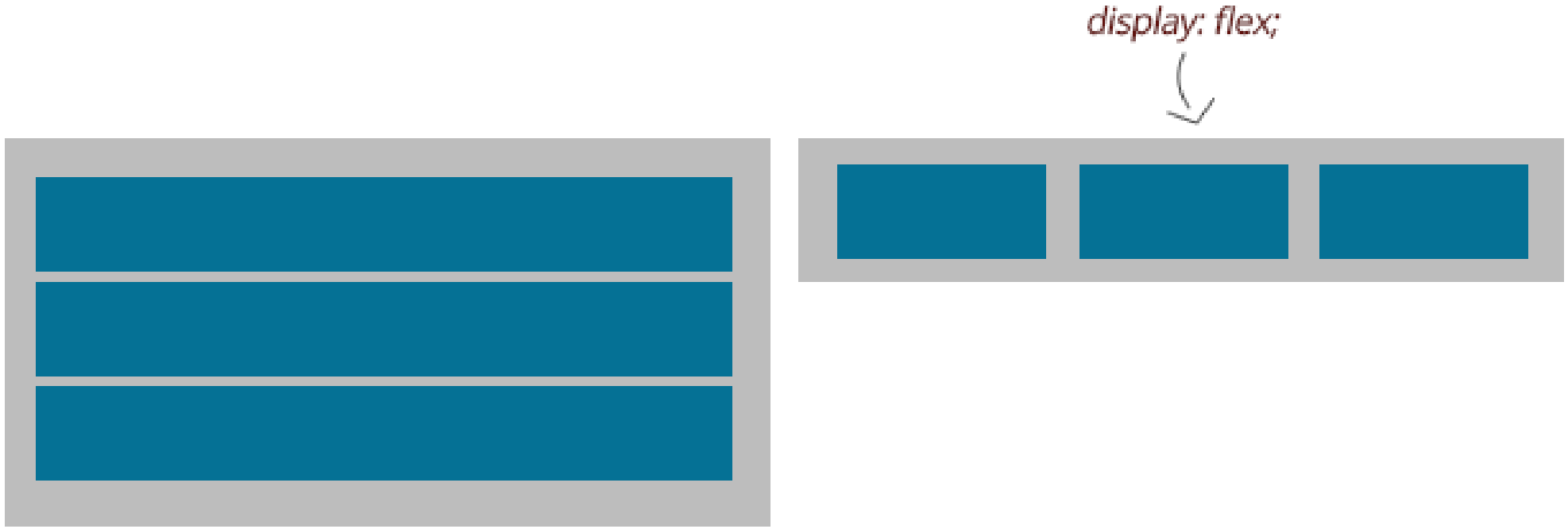
hijos directos de **flex container** y automáticamente se vuelven **items flexibles**



display: flex

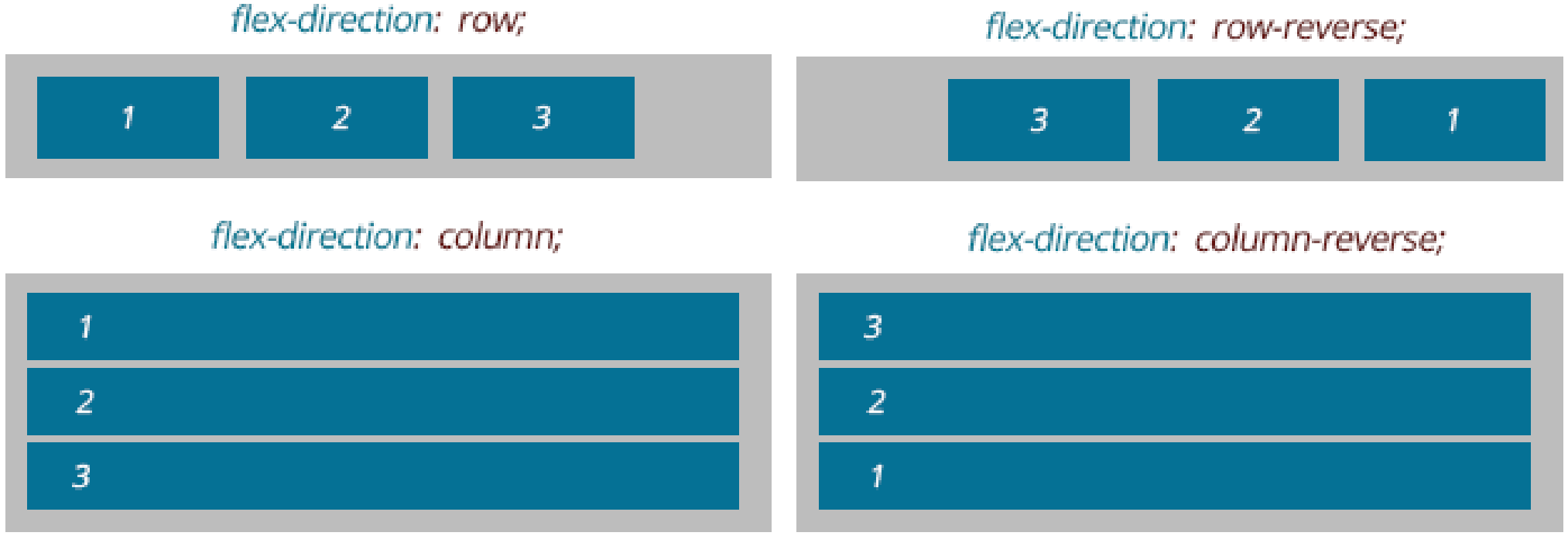
Presentación

- Uso de **display**
  - **flex** para aplicárselo a los elementos de nivel bloque
  - **inline-flex** para aplicárselo a los elementos de los de nivel inline.
  - **flex00.html y flex01.html**



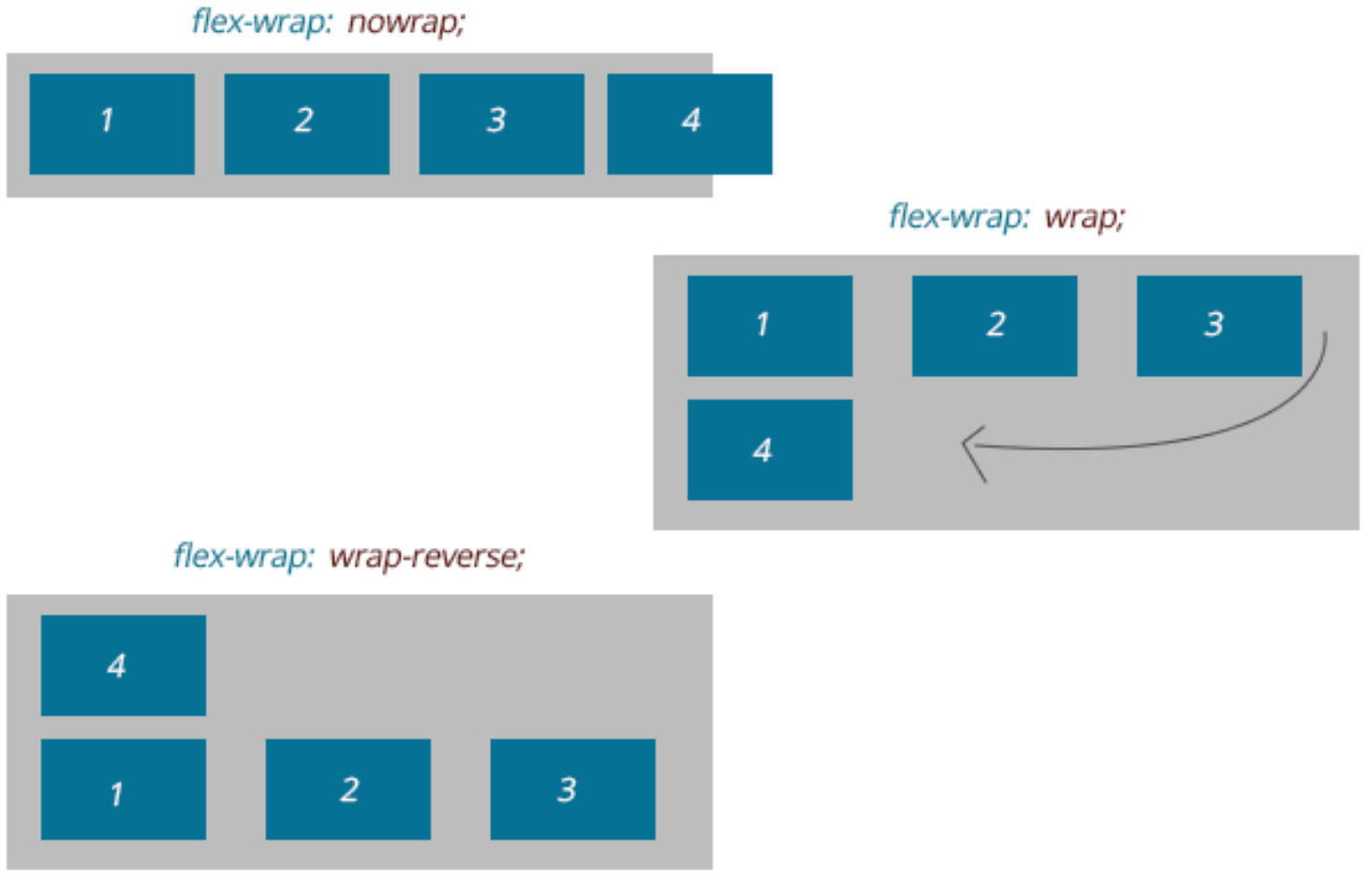
Presentación

- **flex-direction**
  - Establecer la **dirección de los flex items** en el eje principal dentro del **flex container.**
  - **flex02, flex03, flex04, flex05**



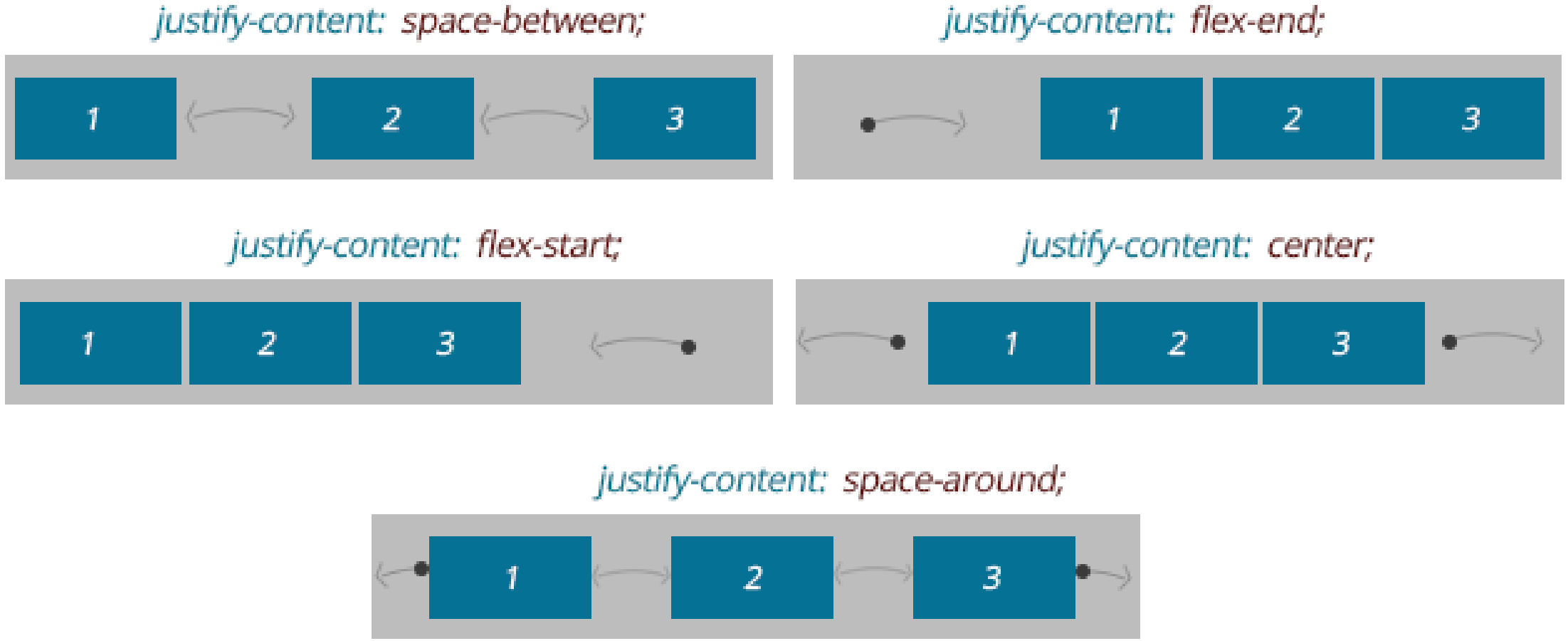
Presentación

- **flex-wrap**
  - Controla si el contenido flexible desborda el contenedor o se ajusta al tamaño de su **flex container**.
  - **flex06, flex07, flex08**



Presentación

- **justify-content**
  - Permite controlar la alineación de los elementos flexibles en el main axis o eje principal.



Presentación

- **align-items**
  - Esta propiedad junto a sus valores nos va a permitir alinear los flex items en el cross axis, o eje transversal. Se aplica al contenedor

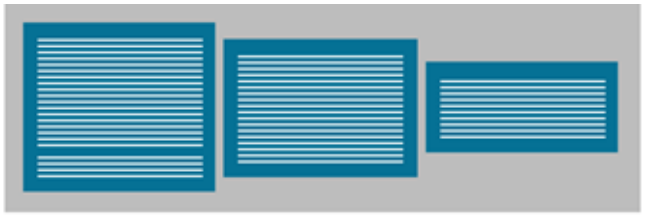
Flex items en el principio/final del eje transversal, en el start/end

align-items: flex-start;

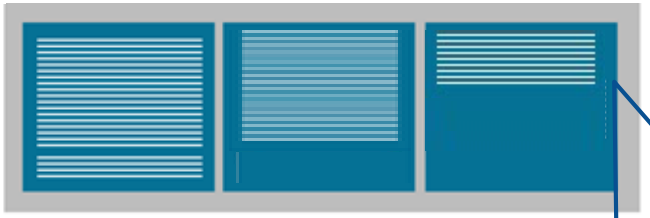
align-items: flex-end;

Flex items en el centro del eje transversal.

align-items: center;

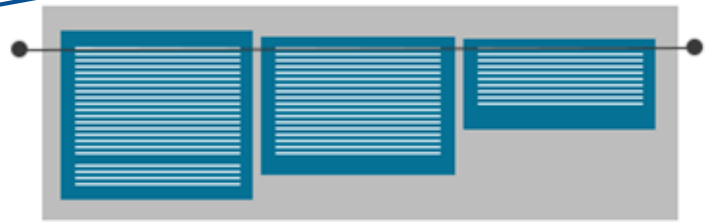


align-items: stretch;



flex items se alinean teniendo como base la primera línea del texto de cada ítem. Modifica el tamaño de la fuente de un div para ver la diferencia con flex-start

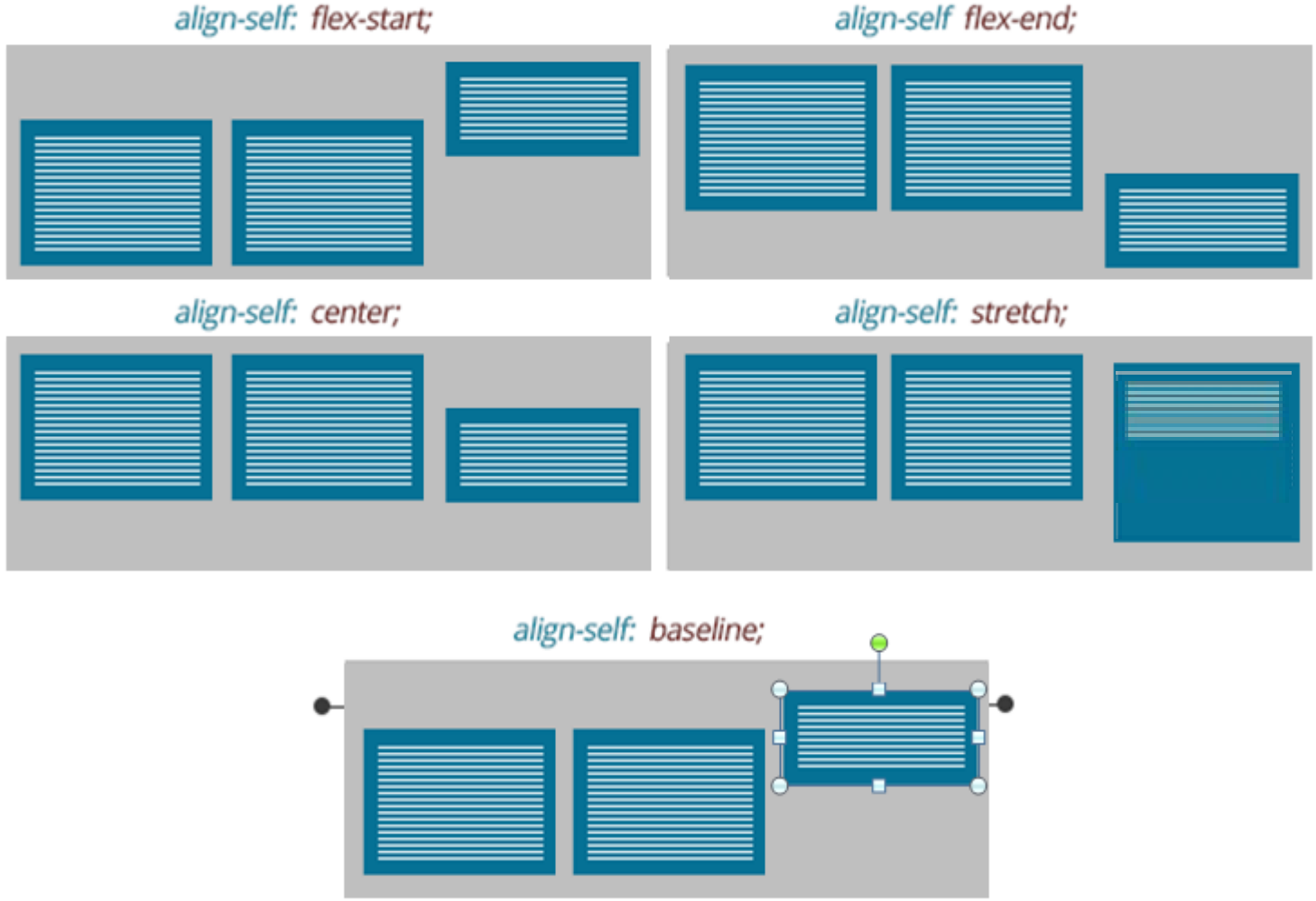
align-items: baseline;



flex items se alinean y les iguala su altura llevándolos desde start hasta end del cross axis o eje transversal.

Presentación

- **align-self**
  - Esta propiedad junto a sus valores nos va a permitir alinear un flex items en el cross axis, o eje transversal. Se aplica al ítem (elemento).



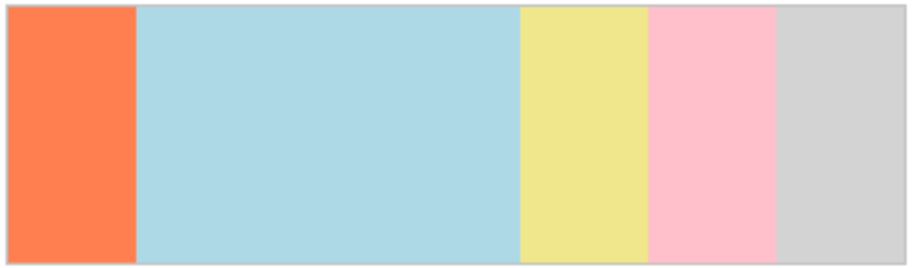
Presentación

- **order**
  - Establece el orden en el que aparecen los elementos en una caja flexible
  - Por defecto es **order:0;**

```
.uno {
    background-color: #bacee3;
    order: 3;
    -webkit-order: 3;
}
.dos{
    background-color: #b6ebb3;
    order: 2;
    -webkit-order: 2;
}
.tres {
    background-color: #7ebdbb;
    order: 1;
    -webkit-order: 1;
}
```

Presentación

- **flex-grow**
  - Determina el número de veces que será mayor respecto de los elementos que componen el elemento flexible



Note: Internet Explorer 10 and earlier versions do not suppo

```
<style>
#main {
    width: 350px;
    height: 100px;
    border: 1px solid #c3c3c3;
    display: -webkit-flex; /* Safari */
    display: flex;
}

/* Safari 6.1+ */
#main div:nth-of-type(1) {-webkit-flex-grow: 1;}
#main div:nth-of-type(2) {-webkit-flex-grow: 3;}
#main div:nth-of-type(3) {-webkit-flex-grow: 1;}
#main div:nth-of-type(4) {-webkit-flex-grow: 1;}
#main div:nth-of-type(5) {-webkit-flex-grow: 1;}

/* Standard syntax */
#main div:nth-of-type(1) {flex-grow: 1;}
#main div:nth-of-type(2) {flex-grow: 3;}
#main div:nth-of-type(3) {flex-grow: 1;}
#main div:nth-of-type(4) {flex-grow: 1;}
#main div:nth-of-type(5) {flex-grow: 1;}
</style>
```

```
<body>

<div id="main">
  <div style="background-color:coral;"></div>
  <div style="background-color:lightblue;"></div>
  <div style="background-color:khaki;"></div>
  <div style="background-color:pink;"></div>
  <div style="background-color:lightgrey;"></div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier versions do not support the
flex-grow property.</p>

<p><b>Note:</b> Safari 6.1 (and newer) supports an alternative, the -webkit-
flex-grow property.</p>
```

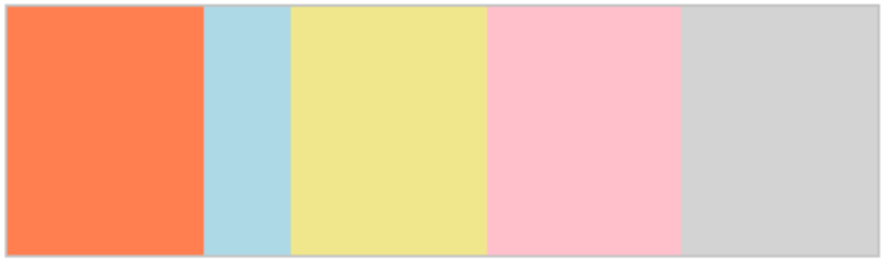
Presentación

- **flex-shrink**
  - Determina el factor de reducción, el número de veces que se reducirá el tamaño en relación a los demás cuando hay espacio negativo en el contenedor. (el contenedor es más pequeño de los anchos combinados de los elementos que hay en su interior). Por defecto es ‘1’.

Presentación

```
<style>
#main {
    width: 350px;
    height: 100px;
    border: 1px solid #c3c3c3;
    display: -webkit-flex; /* Safari */
    display: flex;
}

#main div {
    -webkit-flex-grow: 1; /* Safari 6.1+ */
    -webkit-flex-shrink: 1; /* Safari 6.1+ */
    -webkit-flex-basis: 100px; /* Safari 6.1+ */
    flex-grow: 1;
    flex-shrink: 1;
    flex-basis: 100px;
}

#main div:nth-of-type(2) {
    -webkit-flex-shrink: 3; /* Safari 6.1+ */
    flex-shrink: 3;
}
</style>
```

```
<div id="main">
  <div style="background-color:coral;"></div>
  <div style="background-color:lightblue;"></div>
  <div style="background-color:khaki;"></div>
  <div style="background-color:pink;"></div>
  <div style="background-color:lightgrey;"></div>
</div>

<p><b>Note:</b> Internet Explorer 10 and earlier versions do not support the flex-shrink property.</p>

<p><b>Note:</b> Safari 6.1 (and newer) supports an alternative, the -webkit-flex-shrink property.</p>
```



Note: Internet Explorer 10 and earlier versions do not support the flex-shrink property.

Note: Safari 6.1 (and newer) supports an alternative, the -webkit-flex-shrink property.

Presentación

- **flex-basis**
  - Toma el mismo valor que la propiedad ‘width’ y establece el tamaño inicial del elemento antes de distribuir el espacio libre de acuerdo con los ratios de flex-grow o flex-shrink. Cuando se omite, su valor es ‘main-size’ (anteriormente, ‘auto’).

Presentación

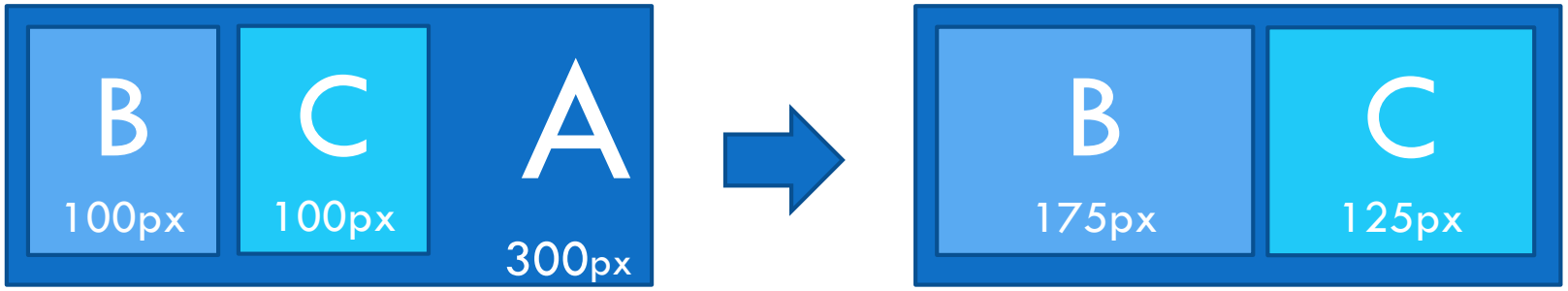
- **flex**
  - Establece cómo crece o decrece un elemento flexible dentro del contenedor en relación a los demás.

flex: flex-grow flex-shrink flex-basis;

```
A { display: flex;}
B { flex: 3 1 100px;}
C { flex: 1 2 100px;}
```

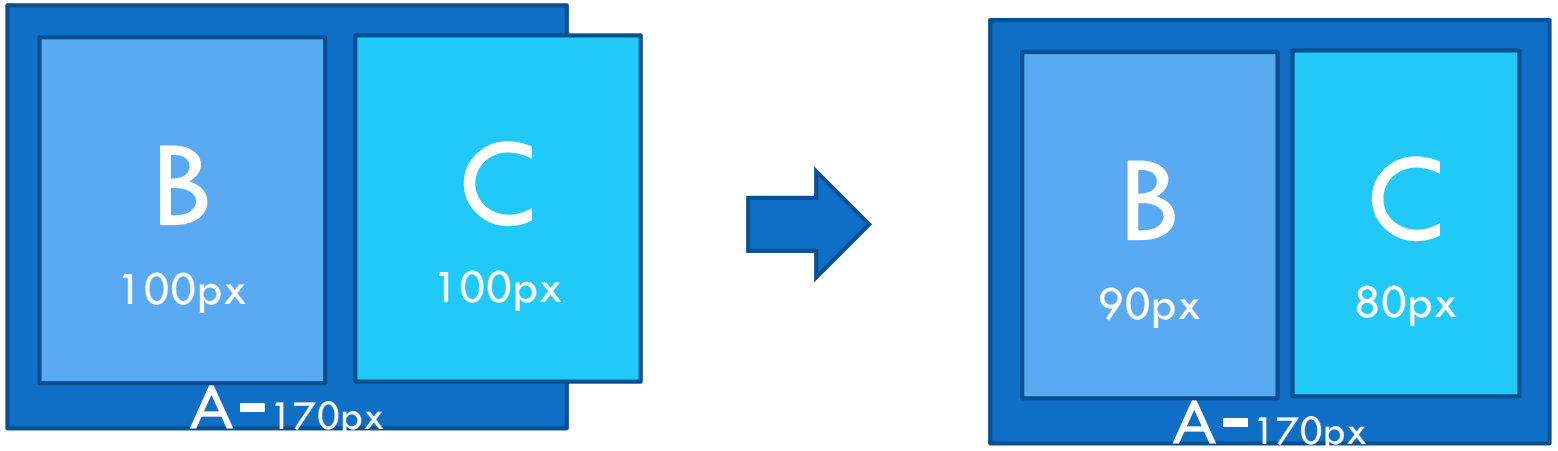
A es un contenedor flexible de 300px de ancho
B y C son elementos contenidos en A.
¿Cuánto ocupan B y C?

Presentación



- Se tienen que repartir los 100px que quedan.
- Como sobra espacio en A se aplica flex-grow, el primer parámetro, en proporción 3 a 1.
- B = 100px (espacio establecido en flex) + 75px (del espacio que resta por llenar)
- C = 100px (espacio establecido en flex) + 25px (del espacio que resta por llenar)

Presentación



- ¿Y si A tiene 170px de ancho?
  - Se aplica la propiedad **flex-shrink** (segundo valor de flex) en proporción 1 a 2.
  - Se resta a 100px (**flex-basis**, el tercer valor de C y D) la proporción de espacio que faltaría en A (30px)
  - B = 100px – 10px = 90px
  - C = 100px – 20px = 80px

Visibilidad

- **visibility**:
  - Especifica si un elemento está o no visible.
  - Valores
    - →**visible**: el elemento es visible. Valor por defecto.
    - →**hidden**: elemento no es visible y se mantiene el espacio que ocupa
    - →**collapse**: en el caso de las tablas, compacta la fila o la columna.
  - **Ejemplo41.css**. Mantener el título principal no visible, pero dejando libre el espacio que ocupa.
  - **Ejemplo41b.css**. Tomando como referencia el ejercicio dónde se distribuía la información de los personajes en una tabla (ejemplo25.html), especificar los títulos como collapse.

Visibilidad

- **opacity**
  - Establece el grado de opacidad de un elemento.
  - Valores
    - →En %, siendo 0 transparente y 100 totalmente opaco el valor 0.5 equivale al 50%.
    - →En IE 8 se usa mediante **filter: alpha(opacity=N);** siendo N el grado de opacidad.
  - El valor de la opacidad es heredado por todos los elementos que lo contienen.
  - El cuarto valor de **rgba**, que indica opacidad, no es heredado por los elementos que lo contienen. **rgba(0, 191, 255, 0.3)**
  - **Ejemplo42.css**: listado con opacidad del 50%

Texto

```
    body {
      background
    }

    div {
      width: 250px;
      height: 250px;
      background-color: rgb(10, 200, 0);
      font-size: 20px;
    }
   </style>
</head>
<body>
  <div> Texto </div>
</body>
</html>
```

Texto

```
    body {
      background
    }

    div {
      width: 250px;
      height: 250px;
      background-color: rgba(10, 200, 0, 0.3);
      font-size: 20px;
    }
   </style>
</head>
<body>
  <div> Texto </div>
</body>
</html>
```

Texto

```
    body {
      background
    }

    div {
      width: 250px;
      height: 250px;
      background-color: rgb(10, 200, 0);
      opacity: 0.3;
      font-size: 20px;
    }
   </style>
</head>
<body>
  <div> Texto </div>
</body>
</html>
```