CSS

Cascading Style Sheets

CSS

- Lenguaje de hojas de estilos para controlar la apariencia de documentos HTML
- Estándar manejado por W3C
- La versión actual es CSS3 (o CSS nivel 3)

COMO AGREGAR CSS

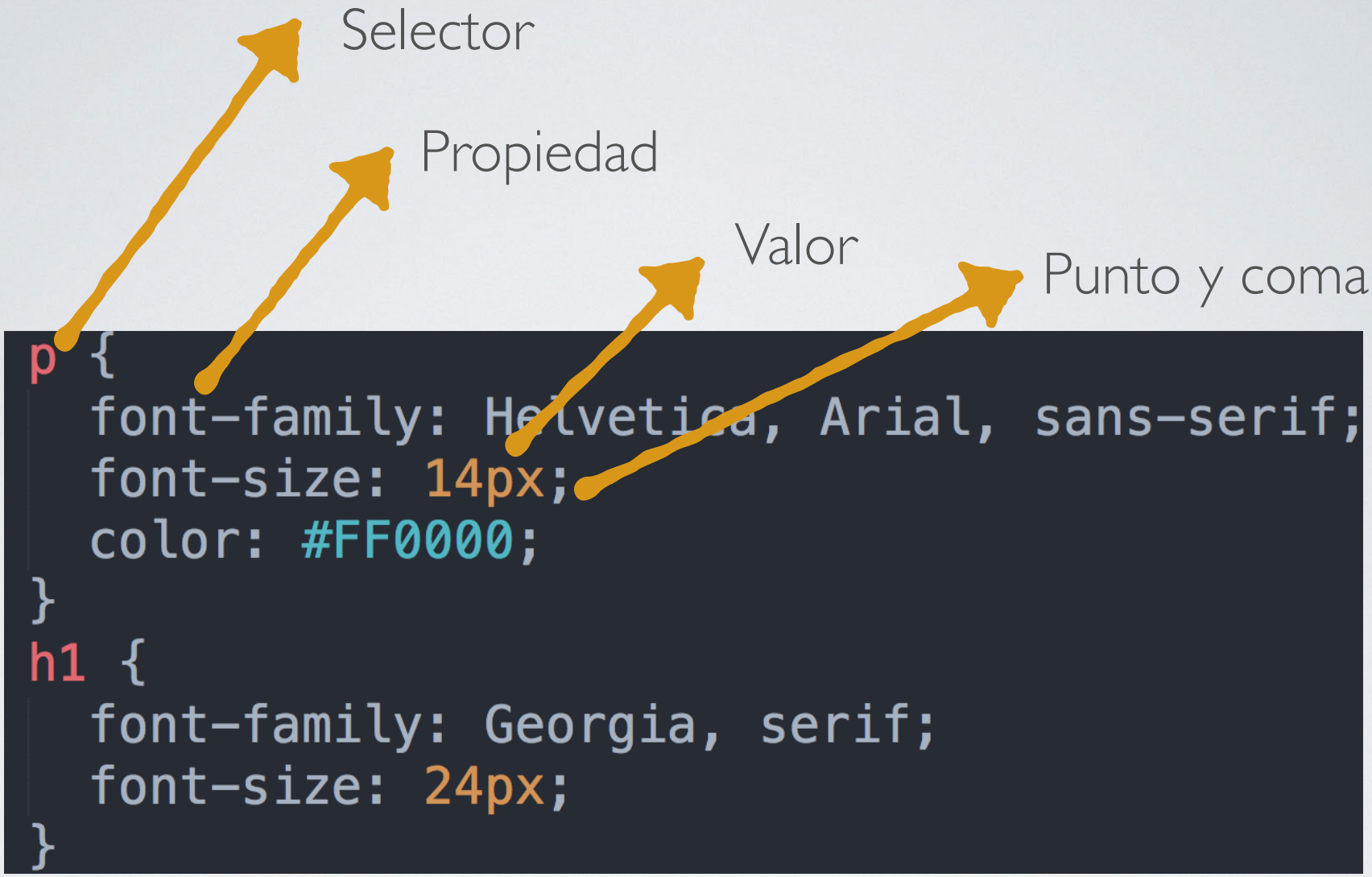
- Para agregar CSS a un documento HTML existen 3 formas:
  - Dentro del documento HTML, con la etiqueta
    `<style type="text/css">...</style>`
  - Usando un archivo externo
    `<link rel="stylesheet" type="text/css" href="estilos.css" media="screen" />`
  - En linea (Mala práctica!)
    `<p style="color:red">Texto rojo</p>`

<LINK />

- Se usa para enlazar recursos externos y debe estar dentro de la etiqueta `<head>`
- Los atributos más importantes son:
  - rel: Indica la relación entre el recurso externo y el documento HTML. Para CSS el valor es siempre "stylesheet"
  - type: Indica el tipo de recurso externo. Para CSS el valor es "text/css"
  - href: Indica la ubicación del recurso externo
  - media: Indica el medio sobre el que se aplicarán los estilos
- `<link rel="stylesheet" type="text/css" href="estilos.css" media="screen" />`

ESTRUCTURA CSS

- CSS tiene 3 elementos importantes: Selectores, propiedades y valores
  - Selector: Define el grupo de elementos a los que se les aplicarán los estilos
  - Propiedad: Aspecto del elemento que se modificará (color, tamaño, etc.)
  - Valor: Es el nuevo valor que se le asignará a la propiedad
- Cada linea de propiedades se debe terminar con un punto y coma (;)


Selector
Propiedad
Valor
Punto y coma
p {
font-family: Helvetica, Arial, sans-serif;
font-size: 14px;
color: #FF0000;
}
h1 {
font-family: Georgia, serif;
font-size: 24px;
}

COMENTARIOS

- En CSS se pueden agregar comentarios que son ignorados por el navegador
- Los comentarios se agregan entre los caracteres `/*` y `*/`
- `/* Este es un comentario CSS */`

SELECTORES

SELECTORES

- Los selectores son los que definen los elementos sobre los que se aplicarán los estilos
- Existen distintos tipos:
  - De etiqueta
  - Descendiente
  - De clase
  - De ID
  - Herencia

SELECTOR DE ETIQUETA

- Se seleccionan los elementos según la etiqueta HTML
- `p { color: red; }`
- `div { border: 5px solid yellow; }`
- `article, section { float: left; }`

SELECTOR DESCENDIENTE

- Permite seleccionar elementos que estén dentro de otros elementos (según etiqueta, clase, ID, etc.)
- Ejemplo: Para buscar todos los `<span>` dentro de un `<p>`
  - `p span { text-decoration: underline; }`

SELECTORES DE CLASE

- Si queremos aplicar estilos a un grupo de elementos específicos lo mejor es usar el atributo class de los elementos HTML
- Un selector de clase se define con un punto (.) seguido por el nombre de la clase
- `<p class="parrafo_grande">…</p>`
- `.parrafo_grande { font-size: 25pt; }`

SELECTOR DE ID

- En algunos casos queremos aplicar estilos a un elemento en específico
- En vez de usar una clase se puede usar el atributo id de una etiqueta HTML
  - Recordatorio: el atributo id debe ser **único** en el documento HTML
- Es similar al selector de clase, pero se usa un "gato" (#)

SELECTOR DE ID

- ```

    Contenido destacado

  ```

- ```
  #destacado {
    color: red;
    background-color: yellow;
  }
  ```

HERENCIA DE PROPIEDADES

- Algunas propiedades CSS se heredan automáticamente a sus descendientes
- `body { color: blue; }` cambia el color del texto de `<body>`, y al ser heredable también cambia el color de todos los descendientes

VALORES Y UNIDADES

VALORES Y UNIDADES

- Las medidas en CSS se usan para definir el alto, el ancho y margen de los elementos y para definir el tamaño de los textos
- Las medidas se componen de un valor y una unidad
- Las unidades pueden ser absolutas o relativas

UNIDADES ABSOLUTAS

- Son unidades que no dependen de un valor de referencia, están completamente definidas
- En general no se usan porque no se adaptan al tamaño de la pantalla del usuario
- Ejemplos: mm, cm, in, pt

UNIDADES RELATIVAS

- Las unidades relativas están definidas con respecto a otro valor
- Las unidades relativas más usadas son
  - em: relativa con respecto al tamaño de la letra del texto
  - ex: relativa con respecto a la altura de la letra "x". 1ex = 0.5 em aprox.
  - px: relativa con respecto a la resolución de pantalla

UNIDADES RELATIVAS

Unidad	Símbolo	Ejemplo
Porcentaje	%	#menu1 {width: 50%;}
Relativa al tamaño de letra	em	#menu1 {font-size: 2.65em;}
Relativo a la x minúscula	ex	#menu1 {font-size: 2.65ex;}
Pixel*	px	#menu1 {font-size: 24px;}

COLORES

COLORES

- Existen 4 formas de definir colores en CSS
  - Palabras clave
  - RGB decimal
  - RGB porcentual
  - RGB hexadecimal

PALABRAS CLAVE

- Existen 240 palabras claves que definen distintos colores
- Por ejemplo: red, yellow, blue, firebrick, indigo, mistyrose
- Lista completa: http://www.w3schools.com/cssref/css_colornames.asp

RGB DECIMAL/PORCENTUAL

- Los colores RGB se definen mediante la combinación de saturaciones de 3 colores: Rojo, verde y azul
- Estas saturaciones se pueden expresar de diversas maneras:
  - Porcentual: desde 0% a 100%
  - Decimal: desde 0 a 255
- En CSS se definen con rgb(R, G, B), donde cada parámetro es la saturación para el color indicado

RGB HEXADECIMAL

- También es posible definir colores RGB mediante un código hexadecimal
  - Hexadecimal: Sistema numérico de base 16, desde 0 a F
- La saturación de cada color se define con 2 símbolos por saturación (16*16 = 256) y se debe anteponer el símbolo "#"
  - Ejemplos: #FF0000 -> Rojo. #00FF00 -> Verde

EJEMPLOS RGB

Color	Decimal RGB	Percentage RGB	Hexadecimal
	255, 0, 0	100%, 0%, 0%	FF0000
	255, 127, 0	100%, 50%, 0%	FF7F00
	255, 255, 0	100%, 100%, 0%	FFFF00
	127, 255, 0	50%, 100%, 0%	7FFF00
	0, 255, 0	0%, 100%, 0%	00FF00
	0, 255, 127	0%, 100%, 50%	00FF7F
	0, 255, 255	0%, 100%, 100%	00FFFF
	0, 127, 255	0%, 50%, 100%	007FFF
	0, 0, 255	0%, 0%, 100%	0000FF
	127, 0, 255	50%, 0%, 100%	7F00FF
	255, 0, 255	100%, 0%, 100%	FF00FF
	255, 0, 127	100%, 0%, 50%	FF007F
	0, 0, 0	0%, 0%, 0%	000000
	128, 128, 128	50%, 50%, 50%	808080
	255, 255, 255	100%, 100%, 100%	FFFFFF
	0, 92, 92	0, 36%, 36%	005C5C
	178, 34, 34	70%, 13%, 13%	B22222

FUENTES

FAMILIA DE TIPOGRAFÍA

- Para definir la tipografía se usa la propiedad CSS `font-family`

- Permite definir un listado de tipografías por prioridad. Si no existe la primera intenta la segunda, etc.

- `font-family: Arial, Helvetica, sans-serif;`

TAMAÑO DE TIPOGRAFÍA

- Para definir el tamaño de usa la propiedad `font-size`

- Ejemplo:

```
p {
   font-size: 14px;
}
```

ESTILO DE TEXTO

- Se pueden definir estilos como itálica o "negrita" con CSS
- Se usa la propiedad `font-style` para itálica y `font-weight` para negrita
- ```
  .negrita {
    font-weight: bold;
  }
  ```
- ```
  .italica {
    font-style: italic;
  }
  ```

COLOR DEL TEXTO

- Para definir el color del texto se usa la propiedad CSS `color`
- El color se puede definir según todas las opciones indicadas anteriormente (RGB, hexadecimal, palabra clave, etc.)
- ```
  .rojo {
    color: red
  }
  ```
- ```
  a {
    color: #0000FF;
  }
  ```

ESPACIO ENTRE LETRAS / PALABRAS

- Para definir el espacio entre **letras** se usa la propiedad `letter-spacing`

- Para definir el espacio entre **palabras** se usa la propiedad `word-spacing`

- ```
  p {
    letter-spacing: 5px;
    word-spacing: 10px;
  }
  ```
  ```

ESPACIO ENTRE LINEAS

- Para definir el espacio entre dos líneas del mismo párrafo (“interlineado”) se usa la propiedad `line-height`

- Define el tamaño de la linea, no el interlineado en sí

  - ```
    line-height: 18px;
    font-size: 14px;
    /* interlineado de 4 px */
    ```
    ```

ALINEACIÓN DEL TEXTO

- El texto puede estar alineado a la izquierda, derecha, centrado o justificado
- Se usa la propiedad `text-align` con los posibles valores: `left`, `right`, `center`, `justify`
-
```
h1 {
   text-align: center;
}
```

DECORACIÓN DEL TEXTO

- Para decorar el texto se usa `text-decoration`
- Permite agregar lineas en el texto, posibles valores:
  - `underline`: Subrayado
  - `overline`: Linea sobre el texto
  - `line-through`: Linea sobre el texto (tarjado)

TRANSFORMACIÓN DEL TEXTO

- Para transformar el texto se usa la propiedad `text-transform`

- Los posibles valores son:

  - `uppercase`: Transforma el texto a mayúsculas

  - `lowercase`: Transforma el texto a minúsculas

  - `capitalize`: Deja con mayúscula la primera letra de cada palabra

FUENTES EXTERNAS

- Antes solo era posible usar fuentes instaladas en los computadores
- ¿Que fuentes tendrán mis usuarios?
- Se creo la propiedad `@font-face` -> Carga fuentes externas (no instaladas en el computador)
- ```
  @font-face {
    font-family: miFuente;
    src: url(miFuente.woff);
  }
  p { font-family: “miFuente”; }
  ```
  ```

GOOGLE FONTS

- En https://www.google.com/fonts existen MUCHAS fuentes para usar
- Tutorial de uso: https://coderhouse.gitbooks.io/css/content/css-8.3-googlefonts.html
- Ejemplo:
  - Cargar fuente:
    ```
    <link href='https://fonts.googleapis.com/css?family=Open
    +Sans' rel='stylesheet' type='text/css'>
    ```
  - Usar fuente:
    ```
    p { font-family: 'Open Sans', sans-serif; }
    ```

IONICONS

- Tambien se pueden usar fuentes “no-convencionales”
- Ionicons es una fuente de iconos -> http://ionicons.com/
- Ejemplo:
  - Cargar fuente:
    ```
    <link href='http://code.ionicframework.com/ionicons/2.0.1/
    css/ionicons.min.css' rel='stylesheet' type=‘text/css’>
    ```
  - Usar fuente:
    ```
    <span class=“ion-home”></span>
    ```

FONDOS Y BORDES

COLOR DE FONDO

- Con la propiedad `background-color` se define un color de fondo
- El valor puede ser un color en cualquiera de sus formas (palabra clave, RGB decimal, hexadecimal, porcentual, etc.)
-
```
.fondo_rojo {
  background-color: red;
}
```

IMAGEN DE FONDO

- La propiedad `background-image` permite una imagen de fondo
- Puede tener url relativa (./images/fondo.jpg) o absoluta (http://blah.com/fondo.jpg)
- La url se define con `url()`
- ```
  #fondo {
    background-image: url(./images/fondo.jpg);
  }
  ```
  ```

REPETICIÓN DE FONDO

- Por omisión el fondo se repite como mosaico
- Para manejar esto se usa la propiedad `background-repeat`
- Los posibles valores son:
  - `no-repeat`: La imagen de fondo no se repite
  - `repeat-x`: La imagen se repite horizontalmente
  - `repeat-y:` La imagen se repite verticalmente
- `body { background-image: url("imagenes/fondo.jpg"); background-repeat: repeat-x; }`

POSICIÓN DE FONDO

- Una imagen de fondo se puede posicionar con la propiedad `background-position`
- Se pueden usar las palabras clave `[top, center, bottom]` y `[left, center, right]`
- También se puede indicar una posición en específico (por ej, en pixeles)
-
```
background-position: right top;
background-position: center center;
background-position: 10px 20px;
```

BORDE

- Para el borde de un elemento se pueden definir:
  - Grosor: propiedad `border-width`

    ```
    border-width: 3px;
    ```

  - Color: propiedad `border-color`

    ```
    border-color: red;
    ```

  - Estilo: propiedad `border-style`, con posibles valores: none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

    ```
    border-style: dashed;
    ```

- Se puede hacer en general con la propiedad `border`

  ```
  border: 3px dashed red;
  ```

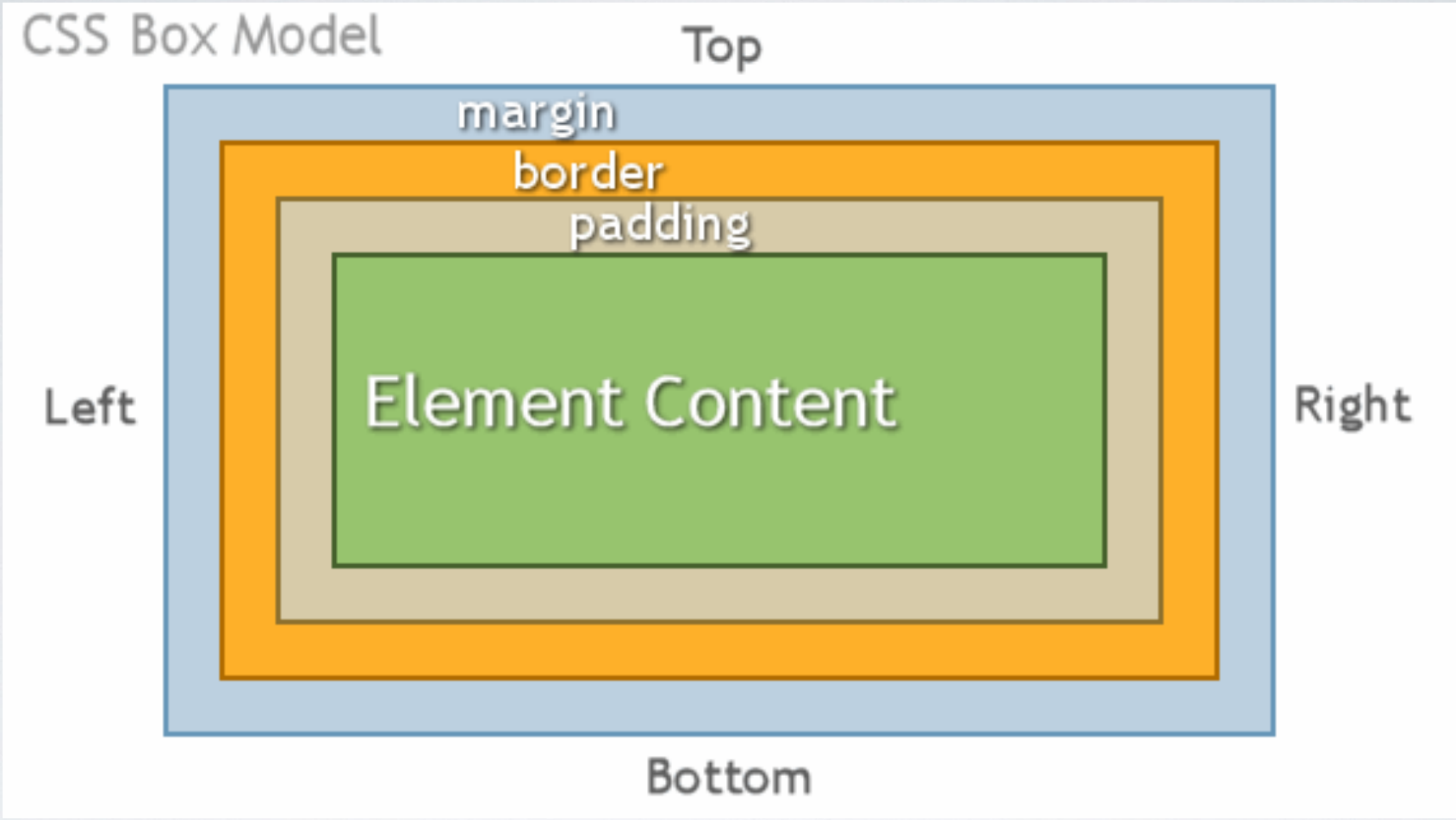
BORDE

- Los bordes también se pueden definir individualmente (`top`, `right`, `bottom`, `left`)
- `border-width: border-top-width, border-right-width,…`
- `border-color: border-top-color, border-right-color, …`
- `border-style: border-top-style, border-right-style, …`
- `border: border-top, border-right, border-bottom, …`

MODELO DE CAJAS

MODELO DE CAJAS

- En HTML cada elemento está contenido en una caja
- Con CSS podemos cambiar todas sus características
- Una caja tiene 4 elementos, de adentro hacia afuera:
  - Contenido (content)
  - Relleno (padding)
  - Borde (border)
  - Margen (margin)



MODELO DE CAJAS

ALTO Y ANCHO

- Para definir el alto y ancho de los elementos se usan las propiedades `height` y `width`
- Pueden tener valores con "unidades" (px, em, etc.) o porcentajes con respecto al padre
- Se pueden definir minimos y máximos con `min-width`, `max-width`, `min-height` y `max-height`

ALTO Y ANCHO

- ```
  div {
    min-width: 50px;
    max-width: 150px;
  }
  ```

- ```
  div {
    height: 300px;
  }
  ```

MÁRGENES

- Los márgenes se pueden definir de manera individual con `margin-top`, `margin-right`, `margin-bottom`, `margin-left`
- También se puede user `margin` como "shortcut"
  - `margin: 10px; /* top = right = bottom = left = 10px */`
  - `margin: 10px 5px; /* T = B = 10px | R = L = 5px */`
  - `margin: 10px 5px 2px; /* T = 10px | R = L = 5px | B = 2px */`
  - `margin: 10px 5px 2px 1px; /* T = 10px | R = 5px | B = 2px | L = 1px */`

RELLENO

- Se define el relleno con la propiedad `padding`
- Al igual que con los márgenes existen:
  - `padding-top`
  - `padding-right`
  - `padding-bottom`
  - `padding-left`
  - `padding /* shortcut */`

TIPOS DE BLOQUES

- Existen diversos tipos de bloques, los 3 más usados son:
  - block
  - inline
  - inline-block
- En CSS se define con la propiedad `display`

BLOCK

- Elemento de "bloque"
- Usa todo el ancho del contenedor padre
- Se pueden modificar sus dimensiones
- Ej: `<p>`

INLINE

- Elemento de linea
- Usa el tamaño justo para el contenido
- No se pueden modificar sus dimensiones
- Ej: `<span>`

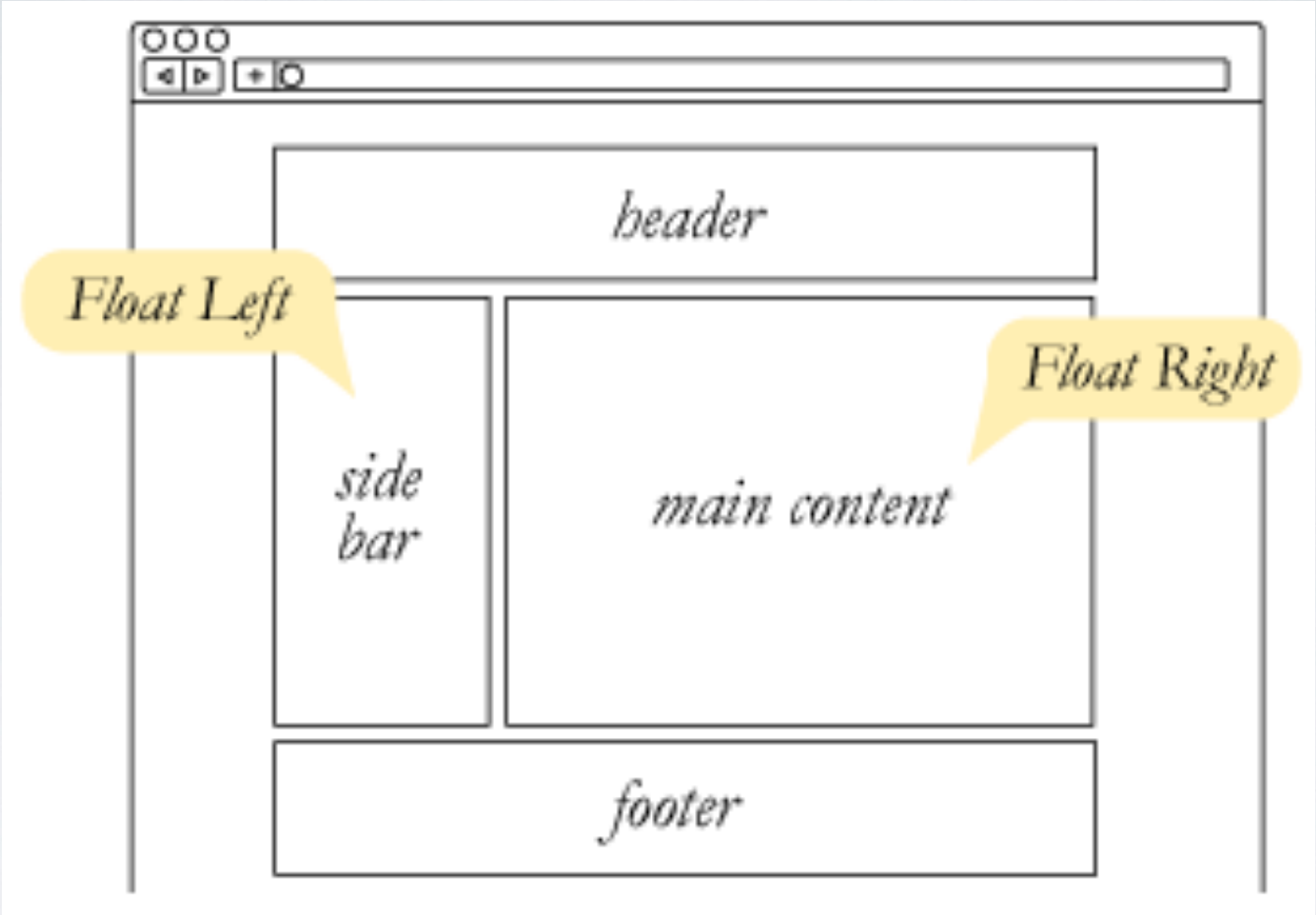
INLINE-BLOCK

- Internamente se comporta como elemento de bloque
  - Se pueden modificar sus dimensiones
- Externamente se comporta como elemento de linea

ELEMENTOS FLOTANTES

ELEMENTOS FLOTANTES

- Los elementos se pueden hacer flotar dentro del layout
- Se usa la propiedad `float`
  - `left`: Hace flotar un elemento a la izquierda
  - `right`: Hace flotar un elemento a la derecha

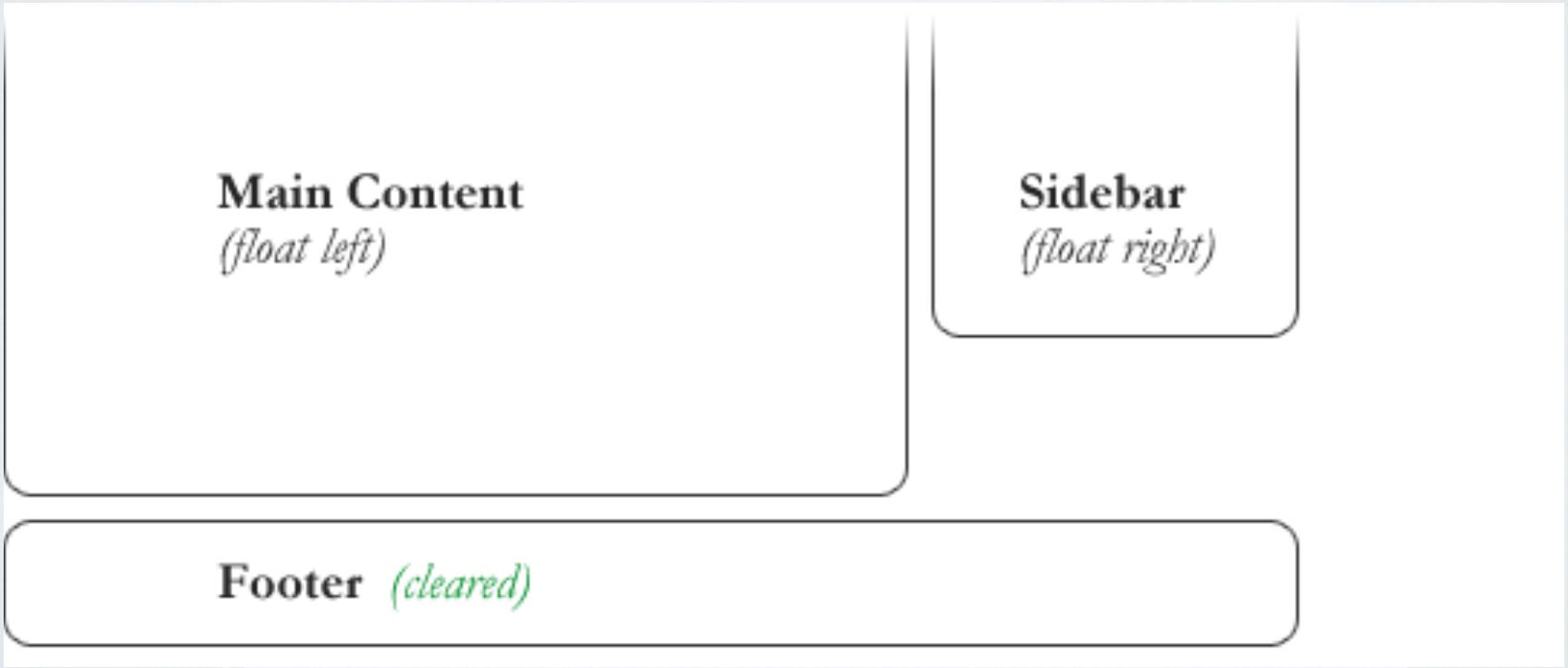


ELEMENTOS FLOTANTES

LIMPIAR ELEMENTOS FLOTANTES

- Al tener elementos flotando se puede deformar el layout
- Para esto se pueden "limpiar" los elementos
- "No seguir hasta que se terminen los elementos flotantes"
- Se usa la propiedad `clear` con valores: `left`, `right`, `both`


Main Content
(float left)
Sidebar
(float right)
Footer (not cleared!)


Main Content
(float left)
Sidebar
(float right)
Footer (cleared)

POSICIONAMIENTO

POSICIONAMIENTO

- Un elemento puede posicionarse de distintas formas. Se usa la propiedad `position`
  - `static`: Posicionamiento "normal"
  - `relative`: Se asigna una posición con respecto a su posición "normal"
  - `fixed`: Se asigna una posición con respecto a la ventana
  - `absolute`: Se asigna una posición con respecto al ancestros más "cercano" con un position que no sea "static"

POSICIONAMIENTO

- Las posiciones relativas se asignan con las propiedades `left`, `right`, `top`, `bottom`
- Esto se usa para todos los posicionamiento menos `static`

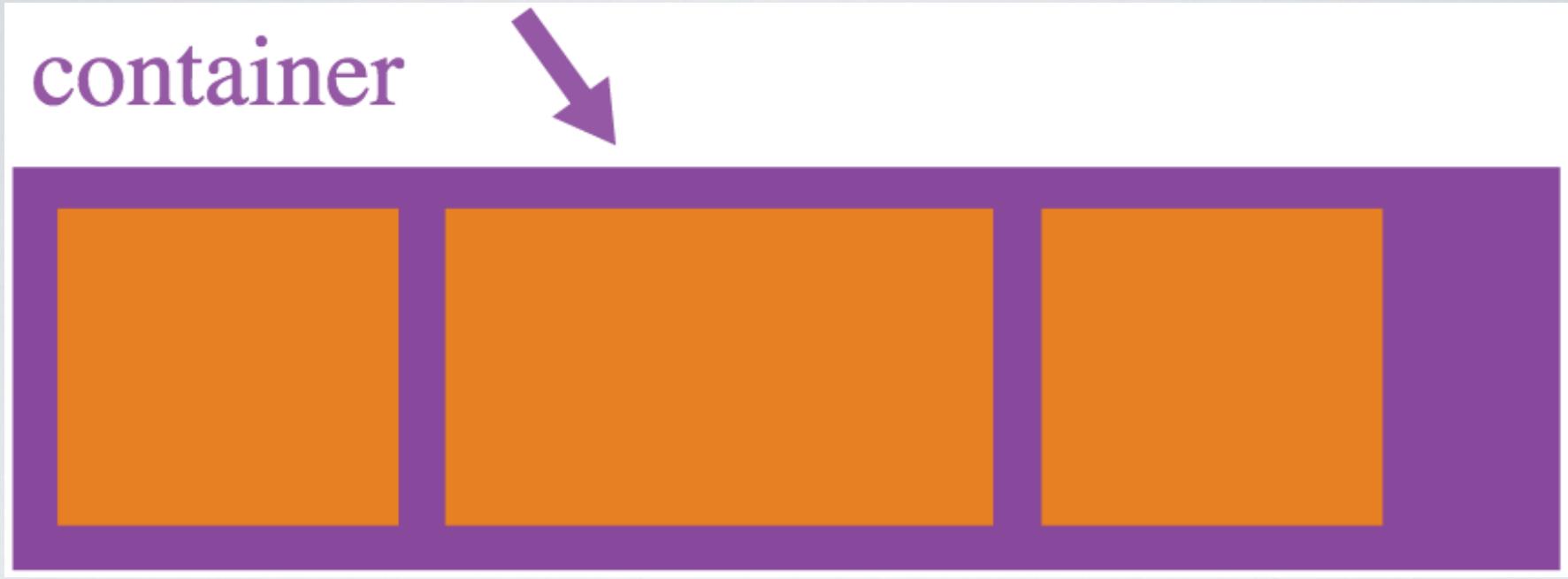
SUPERPOSICIÓN

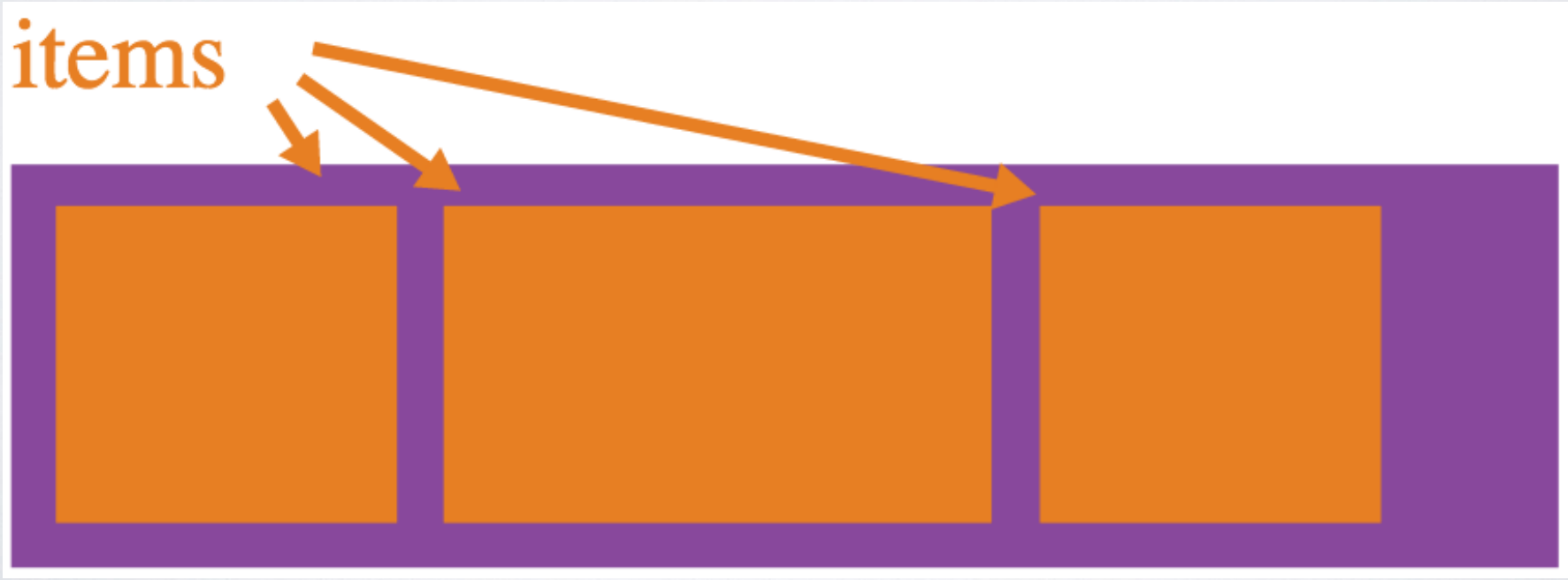
- Si dos o más elementos se superponen se use la propiedad `z-index` para "ordenar"
- `z-index` puede ser positivo o negativo
  - El elemento con `z-index` mayor queda encima
  - Dos elementos con igual `z-index` -> queda encima el último agregado al HTML

FLEXBOX

FLEXBOX

- Permite llenar un contenedor con items de forma flexible
- Usa todo el espacio disponible para el contenido
- Se compone de un contenedor e ítems


container

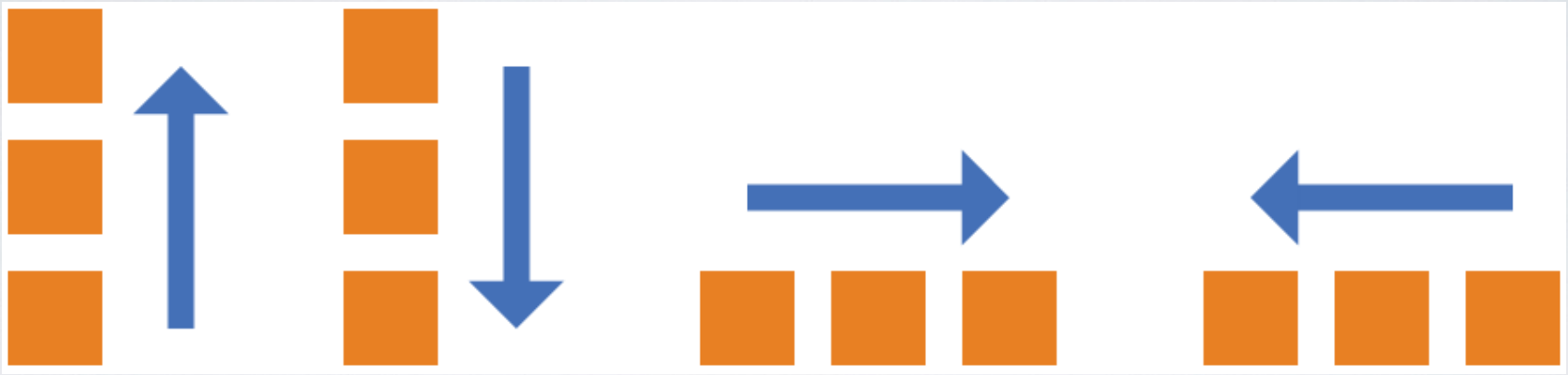

items

CONTENEDOR

- Para definir un contenedor se debe usar la regla `display: flex;`
- Con esto cada uno de los hijos del contenedor se convierte en un ítem

DIRECCIÓN

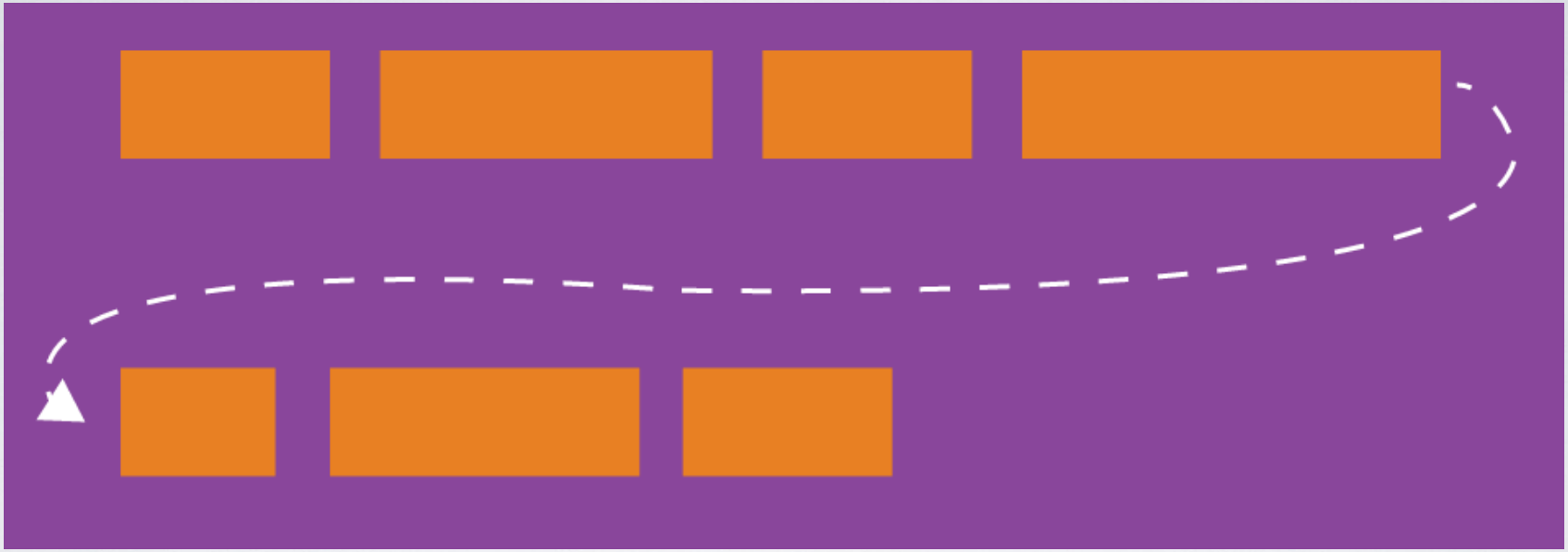
- Dado que el contenido se "ordena" automáticamente uno puede definir la dirección con `flex-direction`
  - `row` (por omisión): De izquierda a derecha
  - `row-reverse`: De derecha a izquierda
  - `column`: De arriba a abajo
  - `column-reverse`: De abajo a arriba



FLEX-DIRECTION

MULTIPLES FILAS/COLUMNAS

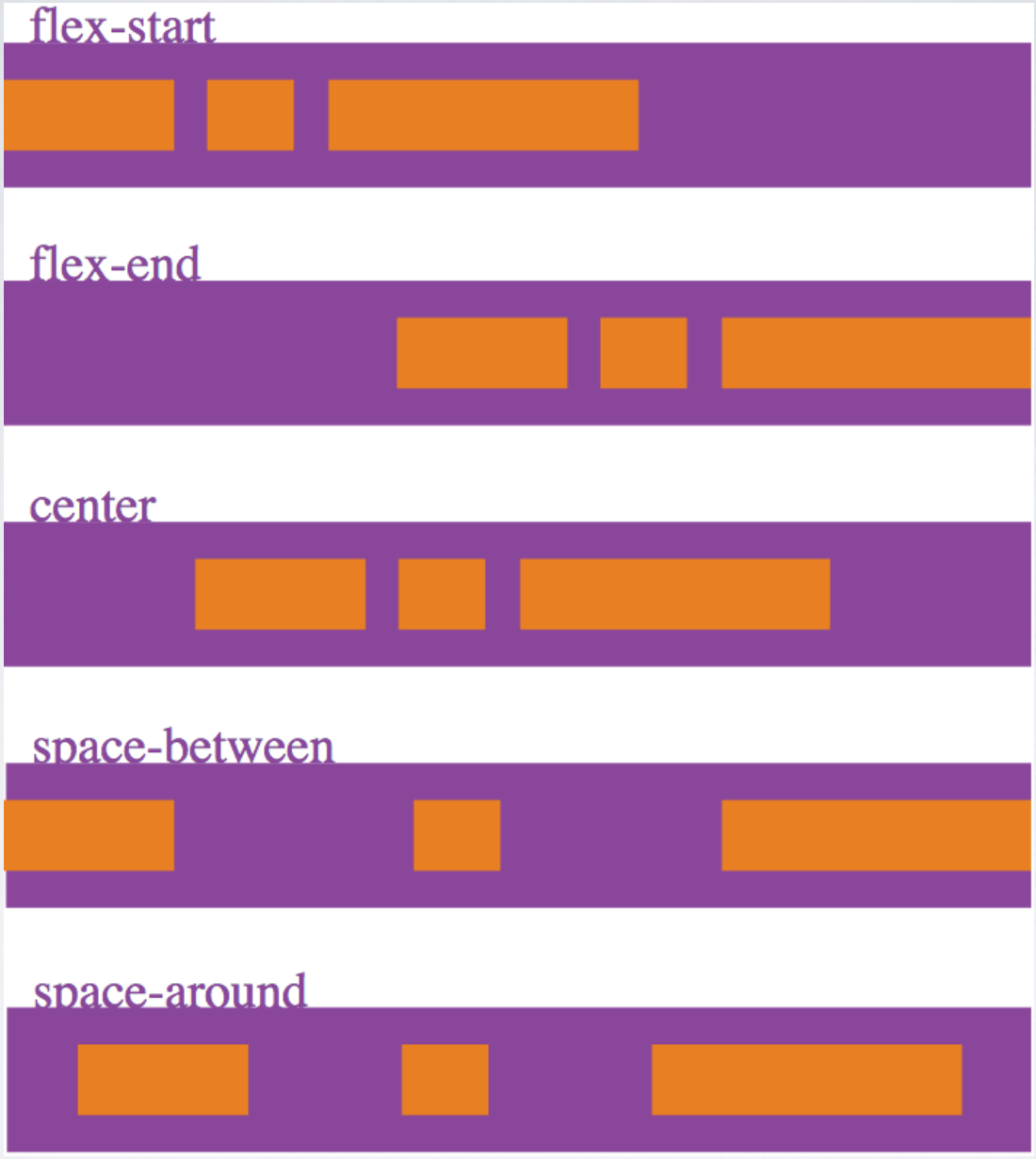
- Por omisión flexbox intentará mostrar todo en una fila o columna
- Esto se puede modificar con `flex-wrap`
  - `no-wrap` (por omisión): Deja todo en una línea
  - `wrap`: Flexbox en multilinea (de izquierda a derecha)
  - `wrap-reverse`: Flexbox multilinea (de derecha a izquierda)



FLEX-WRAP

ALINEAR ÍTEMS HORIZONTAL

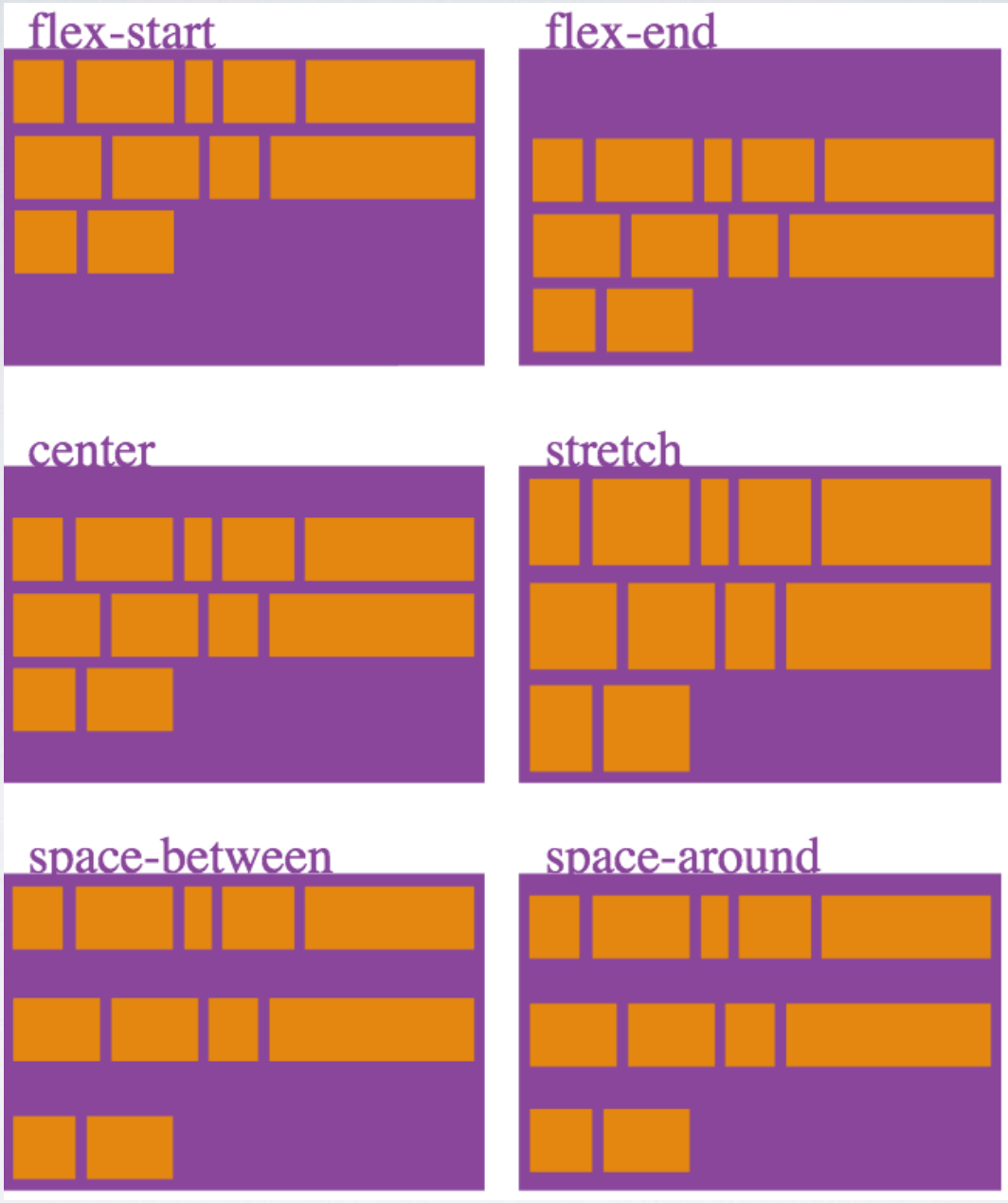
- Para definir el alineamiento horizontal de los ítems se usa `justify-content`
  - `flex-start` (por omisión): ítems agrupados al inicio
  - `flex-end`: ítems agrupados al final de la linea
  - `center`: ítems agrupados al centro
  - `space-between`: distribuye el espacio libre entre los ítems
  - `space-around`: distribuye el espacio libre entre y alrededor de los ítems



JUSTIFY-CONTENT

ALINEAR ÍTEMS VERTICAL

- Para el alineamiento vertical se usa `align-content`
  - `flex-start`: ítems agrupados arriba
  - `flex-end:` ítems agrupados abajo
  - `center`: ítems agrupados al centro
  - `space-between`: espacio libre se distribuye entre los ítems
  - `space-around`: espacio libre se distribuye entre y alrededor de los ítems
  - `stretch` (por omisión): ítems se agrandan para llenar el espacio disponible



ALIGN-CONTENT

RECURSOS

- https://coderhouse.gitbooks.io/css/content/
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/