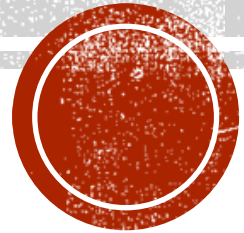
UT4
CSS



PROFESOR: JOSÉ JOAQUÍN MORENO GALLARDO

RAS: RA1-2-3

OBJETIVOS

- Analizar las diferentes opciones de la presentación de la información en documentos web
- Valorar la importancia de definir y aplicar guías de estilos.
- Seleccionar colores y otros elementos de diseño
- Crear y usar plantillas de diseño
- Reconocer la posibilidad de ,codificar las etiquetas HTML
- Definir estilos de forma directa
- Asociar estilos globales a hojas externas
- Identificar las propiedades de cada elemento
- Crear clases de estilos

INTRODUCCIÓN CSS

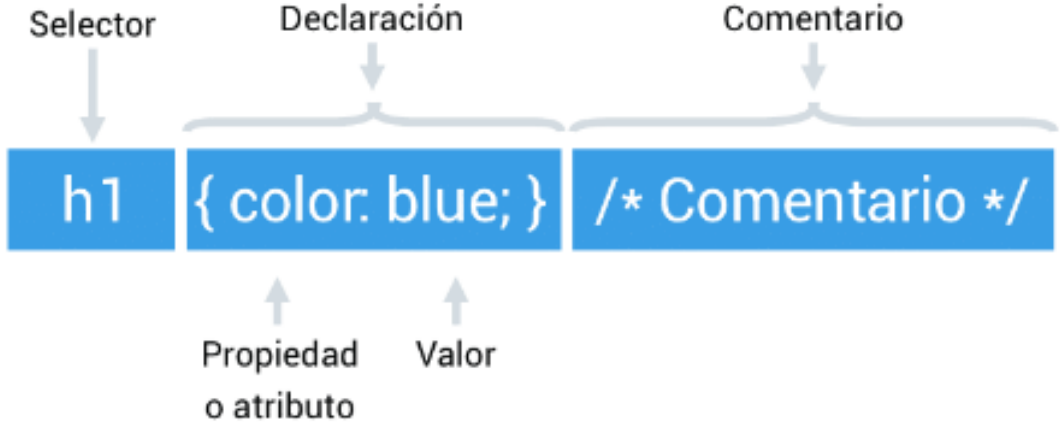
CSS (Cascading Style Sheets, Hojas de Estilo en Cascada en español) es un lenguaje de estilo utilizado para definir la presentación y el diseño de documentos HTML.

El lenguaje CSS está creado por el World Wide Web Consortium (W3C), la comunidad internacional que desarrolla estándares que aseguran el crecimiento futuro de la web y vela por conseguir webs disponibles para todo el mundo.

El lenguaje CSS se ha ido creando a lo largo del tiempo en varios niveles. Cada nivel de CSS se ha construido sobre el anterior, generalmente añadiendo funcionalidades nuevas.

CSS3 ya incluye más de 200 propiedades o atributos.

INTRODUCCIÓN CSS



Cada navegador ofrece un soporte CSS distinto. Por este motivo, es necesario comprobar que nuestras webs se visualizan correctamente en los diferentes navegadores, CSS Validator (jigsaw.w3.org/css-validator/) verifica la validez del código CSS y muestra errores y advertencias para corregir problemas.

```
selector  #id  .clase  [atributo]  :pseudoclase  ::pseudoelemento  {
    propiedad : valor ;
    propiedad : valor
}
```

INTRODUCCIÓN CSS

Los estilos en línea son declaraciones CSS que se integran en las etiquetas HTML mediante el atributo style.

Párrafo de color verde.

Otra manera muy simple de añadir estilo con CSS es utilizando la etiqueta <style> en la cabecera <head> del fichero HTML del sitio.

```
<html>
<head>
   <title>CSS incrustado en la cabecera</title>
   <style>p{color:green;}</style>
</head>
<body>
    <p>Párrafo de color verde.</p>
</body>
</html>
```

INTRODUCCIÓN CSS

Mediante hojas de estilo externas se consigue separar el archivo de estilos del fichero HTML. El archivo de estilos cuenta con la **extensión .css** y se referencia desde HTML mediante el elemento <link>.

```
<html>
<head>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <p>Párrafo de color verde.</p>
</body>
</html>
```

```
p { color: green; }
```

https://validator.w3.org/

SELECTORES CSS

Los selectores pueden apuntar a elementos específicos, clases, identificadores o incluso atributos de un elemento.

Selector universal
Sintaxis: * { atributo:valor; }
Ejemplo: * { color: grey; }/* El estilo se aplicará a todos los elementos de la página*/

Selector etiqueta
Sintaxis: etiqueta { atributo:valor }
Ejemplo: p {color: green;} /* El estilo se aplicará a todos los elementos <p>.*/

Selector clase
Sintaxis: .clase { atributo:valor }
Ejemplo: .blend otro-estilo{color: red;} /* El estilo se aplicará a cualquier elemento que tenga la clase .blend */. Un elemento puede tener múltiples clases: <div **class="blend otro-estilo**">

SELECTORES CSS

Selector identificador

Sintaxis: #id { atributo:valor }

Ejemplo: #cent {color: blue;} /* El estilo se aplicará al elemento que tenga el id #cent */

Selector descendiente

Sintaxis: selector1 selector2 selectorN {atributo: valor;} /* El estilo se aplica sobre el selector N */

Ejemplo: div p { color: black;}

Combinación de selectores

Sintaxis: selector1, selector2, selector3{atributo: valor;} /* El estilo se aplica sobre los selectores indicados */

Ejemplo: div, p { color: orange;}

SELECTORES CSS

Selector de hijos

Sintaxis: selector1 > selector2 {atributo: valor;} /* El estilo se aplica sobre el selector 2 */

Ejemplo: div > p { color: white;}

Selector adyacente

Sintaxis: selector1 + selector2{ atributo: valor; } /* El estilo se aplica al selector 2 */

Ejemplo: div + p { color: black;} /* El estilo se aplica a todos los párrafos que sean hermanos de un div */

Más adelante veremos los selectores por atributos.

Lista completa: https://www.w3schools.com/cssref/css_selectors.php

SELECTORES CSS

Selector	Descripción
*	Selecciona todos los elementos del DOM
etiqueta	Selecciona todas las etiquetas indicadas
.class	Selección de los elementos con la clase .class
#id	Selección del elemento con id #id
sel1 sel2	Selección de los selectores sel2 que se encuentren dentro de los selectores sel1 (pueden ser hijos de hijos)
.class1.class2	Selección de los elementos con las dos clases: class1 y class2
sel1.class1	Selección de todos los selectores sel1 con clase class1
sel1, sel2	Selección de todos los selectores separados por comas
sel1 > sel2	Selección de los selectores sel2 cuando son hijos de sel1 (no pueden ser hijos de hijos)
sel1 + sel2	Selección del selector sel2 cuando es hermano de sel1 (su elemento padre es el mismo)

UNIDADES DE MEDIDA

- Las unidades absolutas mantienen su aspecto y se visualizan siempre igual independientemente de las características del dispositivo: cm, mm, **px**, pc..
- Las unidades relativas se ajustan a cada tipo de dispositivo ya que dependen de la resolución de cada pantalla:**em,%,**rem,fr…
- Se recomienda **usar unidades relativas**, em o px siendo 16px el tamaño de la fuente que tiene por defecto los navegadores. Se tiene que 16px es igual a 1 em.

Pruebe los siguientes ejemplos:

```
<p style="font-size: 16px;">Párrafo de 16px</p>
<p style="font-size: 12px;">Párrafo de 12px</p>
<p style="font-size: 1em;">Párrafo de 1em</p>
<p style="font-size: 1.5em;">Párrafo de 1.5em</p>
<p style="font-size: 0.5in;">Párrafo de 0.5 pulgadas</p>
<p style="font-size: 8mm;">Párrafo de 8 milímetros</p>
<p style="font-size: 12pt;">Párrafo de 12 puntos</p>
```

UNIDADES DE MEDIDA

- Las unidades absolutas mantienen su aspecto y se visualizan siempre igual independientemente de las características del dispositivo: cm, mm, **px**, pc..
- Las unidades relativas se ajustan a cada tipo de dispositivo ya que dependen de la resolución de cada pantalla:**em,%,**rem,fr…
- Se recomienda **usar unidades relativas**, em o px siendo 16px el tamaño de la fuente que tiene por defecto los navegadores. Se tiene que 16px es igual a 1 em.

Pruebe los siguientes ejemplos:

```
<p style="font-size: 16px;">Párrafo de 16px</p>
<p style="font-size: 12px;">Párrafo de 12px</p>
<p style="font-size: 1em;">Párrafo de 1em</p>
<p style="font-size: 1.5em;">Párrafo de 1.5em</p>
<p style="font-size: 0.5in;">Párrafo de 0.5 pulgadas</p>
<p style="font-size: 8mm;">Párrafo de 8 milímetros</p>
<p style="font-size: 12pt;">Párrafo de 12 puntos</p>
```

COLORES Y FONDO EN CSS

Propiedad	Descripción	Valores
color	Color del texto	RGB | HSL | HEX | nombre del color | RGBA | HSLA
background-color	Color de fondo	RGB | HSL | HEX | nombre del color | RGBA | HSLA | transparent
background-image	Imagen de fondo	url(«…») | none
background-repeat	Repetición de la imagen de fondo	repeat | repeat-x | repeat-y | no-repeat
background-attachment	Desplazamiento de la imagen de fondo	scroll | fixed
background-position	Posición de la imagen de fondo	percentage | length | left | center | right
background-size	Tamaño de la imagen de fondo	auto | cover | contain | valor
Opacity	Transparencia de un elemento	[0 – 1] (0 → totalmente transparente)

COLORES Y FONDO EN CSS

```
<p class="a">Rojo con su nombre</p>
<p class="b">Rojo con Hexadecimal<p/>
<p class="c">Rojo con RGB</p>
<p class="d">Rojo con HSL</p>
<p class="e">Texto con opacidad 0.5</p>
<p class="f">Fondo con opacidad 0.5</p>
<p class="g">Fondo transparente</p>
```

```
.a { background-color: red; }
.b { background-color: #FF0000; }
.c { background-color: RGB(255,0,0); }
.d { background-color: HSL(0,100%,50%); }
.e { background-color: red; opacity: 0.5; }
.f { background-color: rgba(255, 0, 0, 0.5); }
.g { background-color: transparent; }
```

```
body { background-image: url("https://bit.ly/2slA7jo");}
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: repeat-x;}
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: repeat-y;}
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: no-repeat;}
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: repeat;}
body { background-image: url("https://bit.ly/2slA7jo"); background-size: 100px 100px;}
```

```
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: no-repeat;
background-attachment: fixed;}
body { background-image: url("https://bit.ly/2slA7jo"); background-repeat: no-repeat;
background-attachment: scroll;}
```

COLORES Y FONDO EN CSS

Posición en el Centro Superior: Este ejemplo coloca la imagen de fondo en el centro superior del elemento.
div { background-image: url("imagen-fondo.jpg"); **background-position**: center top; }
Posición en la Esquina Superior Derecha: En este caso, la imagen de fondo se coloca en la esquina superior derecha del elemento.
div { background-image: url("imagen-fondo.jpg"); background-position: right top; }
Posición en la Esquina Inferior Izquierda: Aquí, la imagen de fondo se encuentra en la esquina inferior izquierda del elemento.
div { background-image: url("imagen-fondo.jpg"); background-position: left bottom; }
Posición Personalizada en Píxeles: Puedes especificar coordenadas en píxeles para posicionar la imagen de fondo de manera precisa.
div { background-image: url("imagen-fondo.jpg"); background-position: 50px 30px; /* 50px desde la izquierda, 30px desde arriba */ }
Posición en el Centro del Contenedor: Para centrar la imagen de fondo tanto vertical como horizontalmente en el contenedor.
div { background-image: url("imagen-fondo.jpg"); background-position: center center; }
Posición en Porcentajes: Puedes utilizar porcentajes para posicionar la imagen de fondo en relación con el tamaño del elemento.
div { background-image: url("imagen-fondo.jpg"); background-position: 25% 75%; /* 25% desde la izquierda, 75% desde arriba */ }

COLORES Y FONDO EN CSS

La propiedad **background-size** se utiliza para controlar el tamaño de la imagen de fondo en relación con el elemento que la contiene.

/* La imagen se ajusta para cubrir completamente el contenedor */
background-size: cover;

/* La imagen se ajusta para caber completamente dentro del contenedor */
background-size: contain;

/* La imagen tiene un ancho fijo de 200px y se ajusta automáticamente en altura */
background-size: 200px auto;

/* La imagen tiene un ancho fijo del 50% del contenedor y una altura fija de 100px */
background-size: 50% 100px;

PROPIEDADES DEL TEXTO

Propiedad	Descripción	Valores
text-indent	Desplazamiento de la primera línea del texto	longitud | porcentaje
text-align	Alineamiento del texto	left | right | center | justify
text-decoration	Efectos de subrayado y tachado	none | underline | overline | line-through
letter-spacing	Espacio entre caracteres	normal | longitud
word-spacing	Espacio entre palabras	normal | longitud
text-transform	Transformación a mayúsculas / minúsculas	capitalize | uppercase | lowercase | none
line-height	Tamaño del espacio entre líneas (interlineado)	longitud | porcentaje
vertical-align	Alineación vertical	top | middle | bottom baseline | sub | super | valor

PROPIEDADES DE LAS FUENTES

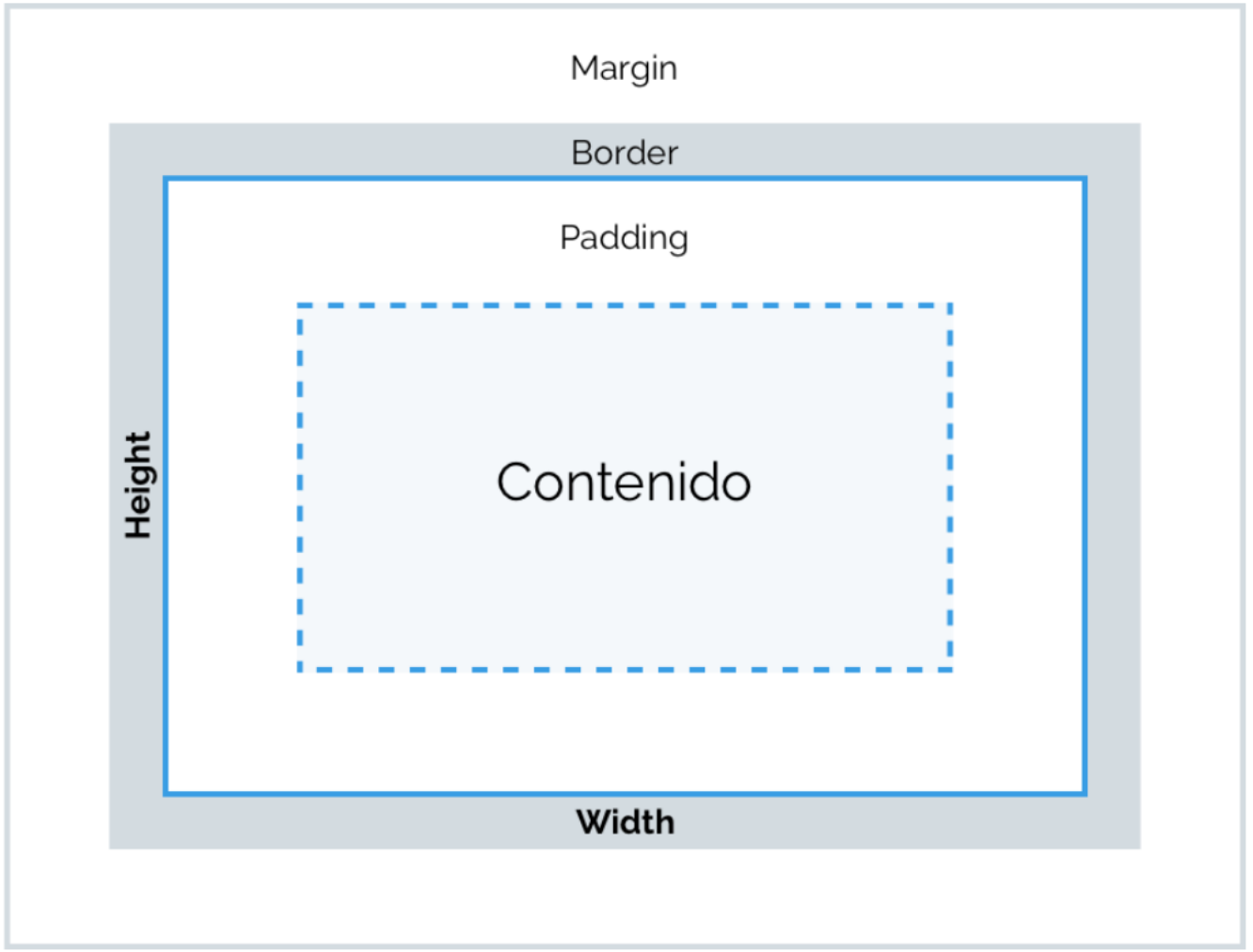
Propiedad	Descripción	Valores
font-family	Familias de fuentes	nombre-familia | *
font-style	Estilo de la fuente	normal | italic | oblique
font-variant	Convierte a mayúsculas manteniendo todas las letras en un tamaño inferior a la primera	normal | small-caps
font-weight	Anchura de los caracteres. Normal = 400, Negrita = 700	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
font-size	Tamaño de la fuente	xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | longitud | porcentaje

PROPIEDADES DE LAS LISTAS

Propiedad	Descripción	Valores
list-style-type	Estilo aplicable a los marcadores visuales o viñetas de las listas	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none
list-style-image	Imagen aplicable a las viñetas de las listas	url() | none
list-style-position	Posición de las viñetas dentro de la lista	inside | outside
list-style	Permite establecer varios estilos de la lista en una sola propiedad	list-style-type | list-style-position | list-style-image

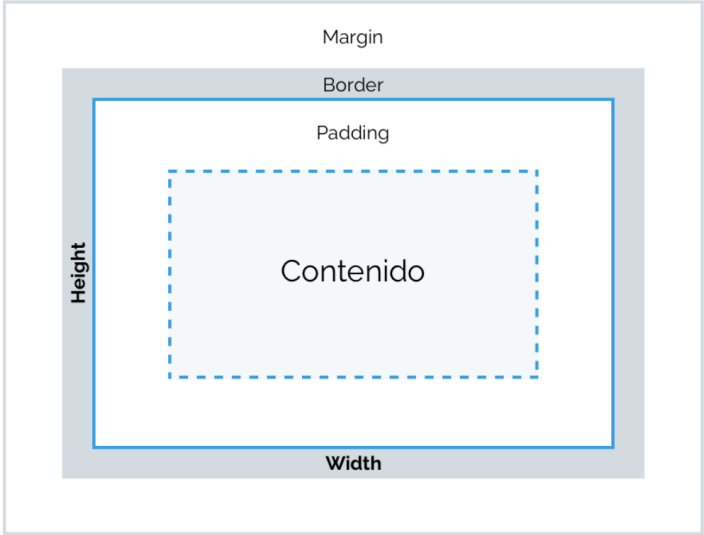
MODELOS DE CAJAS

Las propiedades más importantes de las cajas o contenedores son las siguientes: **margin** (margen externo), **border** (borde) y **padding** (margen interno).



PADDING O RELLENO

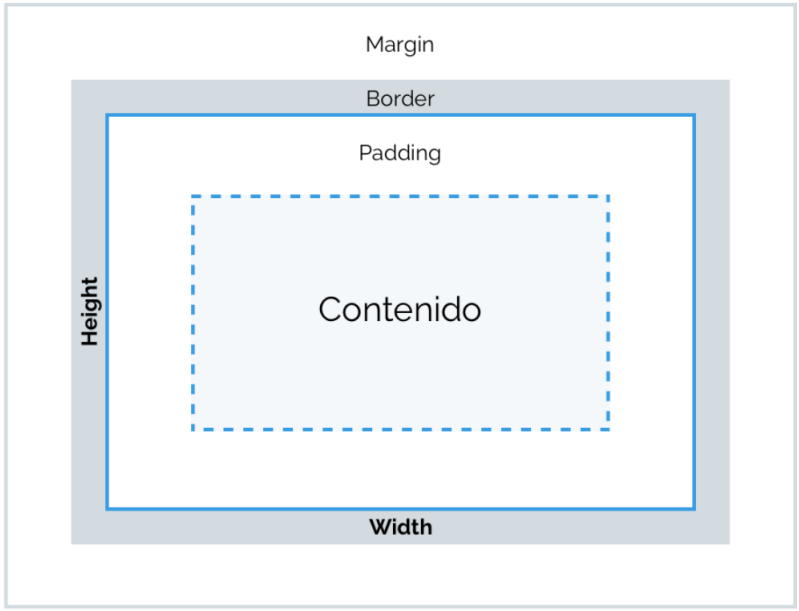
El padding es el margen interno de un elemento, también se le llama relleno y es la cantidad de espacio entre el borde y el contenido del elemento.



Propiedad	Descripción	Valores
padding-top padding-right padding-bottom padding-left	Tamaño del relleno superior, derecho, inferior e izquierdo	longitud | porcentaje
padding	Tamaño del relleno	longitud | porcentaje {1,4}

MARGIN

El margin es el margen externo de un elemento, fuera de cualquier borde definido.



Propiedad	Descripción	Valores
margin-top margin-right margin-bottom margin-left	Tamaño del margen superior, derecho, inferior e izquierdo	longitud | porcentaje | auto
margin	Ancho de los márgenes	longitud | porcentaje | auto {1,4}

BORDER

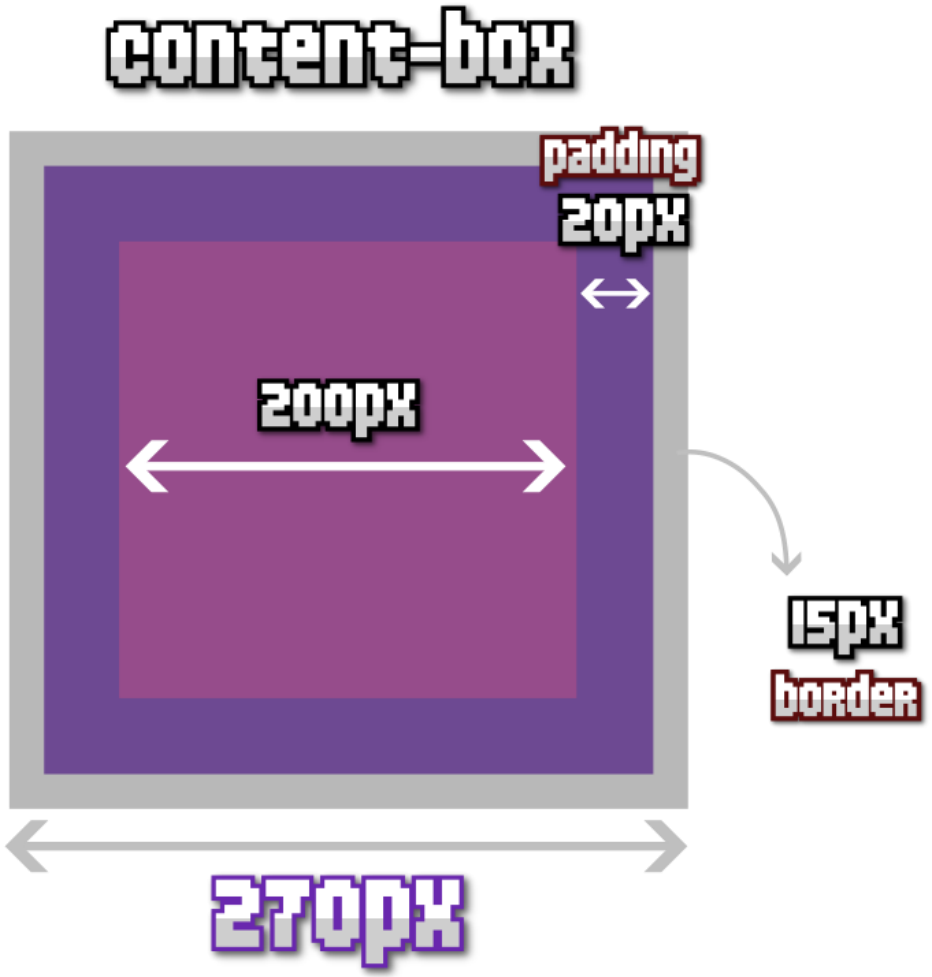
La propiedad border en CSS permite especificar el estilo, el ancho y el color de los bordes de un elemento. Puedes usar diferentes valores para crear distintos tipos de bordes, como líneas lisas, de puntos, redondeados, etc.

Propiedad	Descripción	Valores
border-top-width border-right-width border-bottom-width border-left-width	Anchura del borde superior, derecho, inferior o izquierdo	thin | medium | thick | longitud
border-width	Anchura del borde (reducida)	thin | medium | thick | longitud {1,4}
border-top-color border-right-color border-bottom-color border-left-color	Color del borde superior, derecho, inferior e izquierdo	color | transparent
border-color	Color del borde (reducida)	color | transparent {1,4}
border-top-style border-right-style border-bottom-style border-left-style	Estilo del borde superior, derecho, inferior e izquierdo	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
border-style	Estilo del borde (reducida)	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset {1,4}
border-top border-right border-bottom border-left	Ancho, estilo y color para el borde superior, derecho, inferior e izquierdo	border-top-width | border-top-style | border-top-color
border	Ancho, estilo y color para los bordes (reducida)	border-width | border-style | border-color
border-radius	Curvatura del borde	longitud | porcentaje {1,4}

WIDTH, HEIGHT, MIN-WIDTH, MAX-HEIGHT Y MIN-HEIGHT

Nombre propiedad	Descripción	Valores
width	Establece el ancho del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
height	Establece el alto del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
max-width	Establece el ancho máximo que puede tener un elemento	Unidad de longitud (px, em, %, etc.), none, initial, inherit
min-width	Establece el ancho mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit
max-heigt	Establecer el alto máximo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit
min-height	Establecer el alto mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit

BOX-SIZING

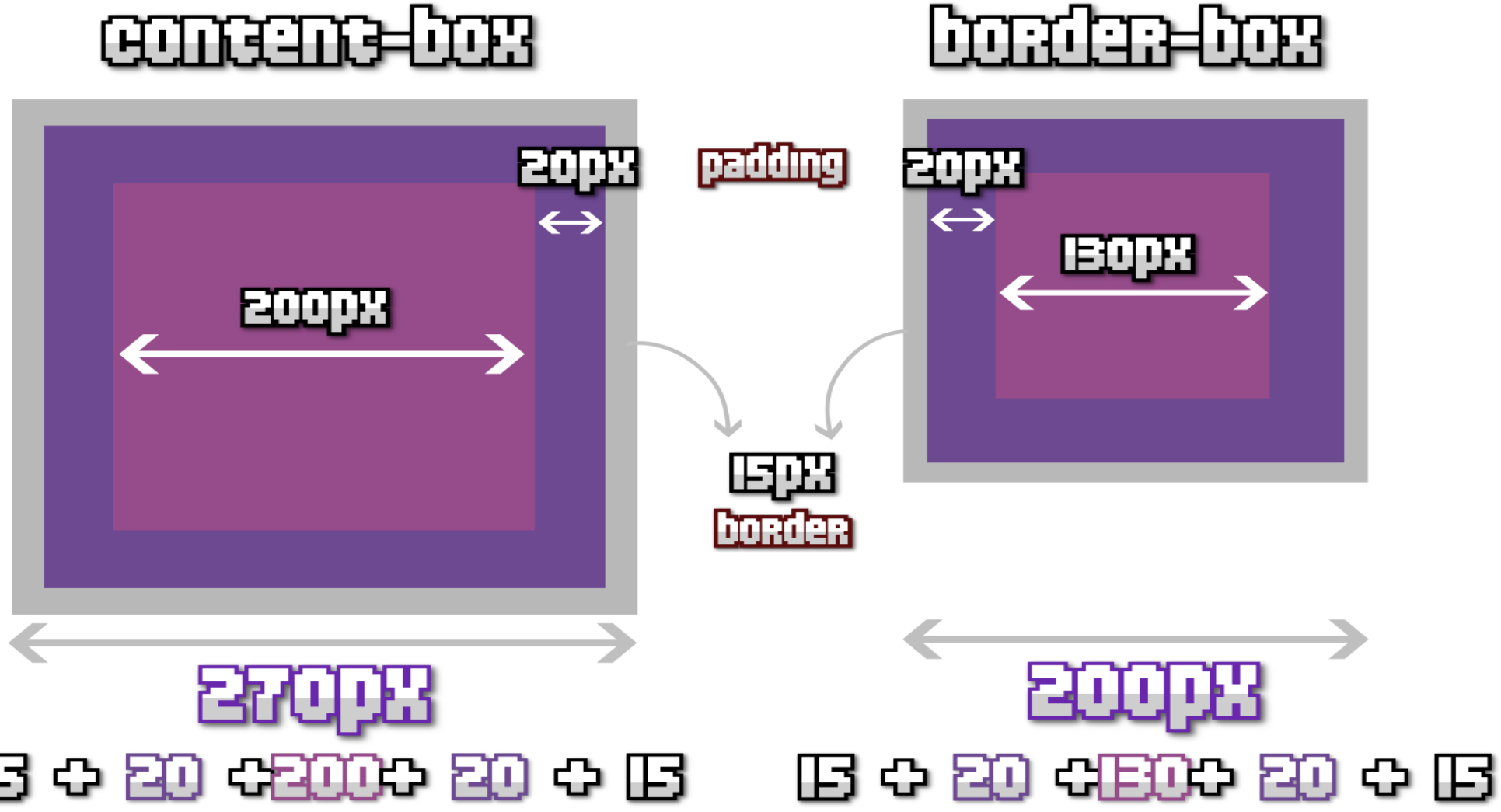


Por defecto, el modelo de cajas de CSS sigue un esquema donde al dar un tamaño a un elemento, concretamente se le da tamaño a su contenido. Sin embargo, si además le añadimos un borde (border) y/o un relleno (padding), se sumará al tamaño del contenido.

Esto puede resultar algo poco intuitivo, ya que un elemento al que le demos un tamaño width: 200px, y que además tenga un border: 15px y un padding: 20px, resultaría en realidad un elemento de tamaño total de 270px.

BOX-SIZING

Box-sizing permite alterar esta versión del modelo de cajas, usando las propiedades border-box.



ANIMACIONES CSS

CSS define tres tipos de degradados:

- Degradados lineales (hacia abajo/arriba/izquierda/derecha/diagonalmente)
- Gradientes radiales (definidos por su centro)
- Gradientes cónicos (rotados alrededor de un punto central)

```
background-image: linear-gradient(red, yellow);
background-image: linear-gradient(to right, red , yellow);
background-image:linear-gradient(to right, red, orange, yellow, green, blue, indigo,
violet);
background-image: radial-gradient(red, yellow, green);
background-image: radial-gradient(circle, red, yellow, green);
background-image: conic-gradient(red, yellow, green, blue, black);
background-image: conic-gradient(from 90deg, red, yellow, green);
```

ANIMACIONES CSS

Con CSS puedes agregar **sombra** al texto y a los elementos:

- text-shadow
- box-shadow

```
text-shadow: 2px 2px 5px red;
box-shadow: 10px 10px lightblue;
box-shadow: 10px 10px 5px lightblue;/*Efecto desenfoque*/
```

Podemos controlar el **desbordamiento** de texto CSS , ajuste de línea, reglas de salto de línea y modos de escritura usando text-overflow,, text-justify, word-wrap, Word-break y writing-mode.

Las **transiciones** CSS le permiten cambiar los valores de las propiedades sin problemas, durante un periodo de tiempo.

```
transition: width 2s; transition: width 2s, height 4s;
transition-property:width;transition-duration:2s;transition-timing-function:linear;
transition-delay: 1s;
```

ANIMACIONES CSS

CSS permite la animación de elementos HTML sin utilizar JavaScript.

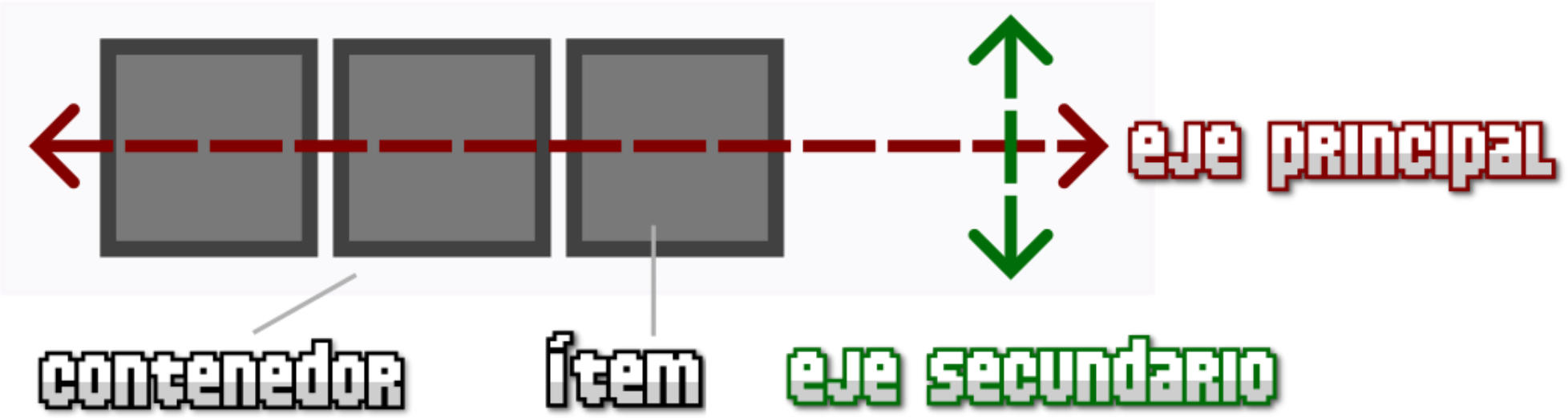
- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: ejemplo;
  animation-duration: 4s;
}

@keyframes ejemplo {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

FLEX CSS

Para empezar a utilizar flex lo primero que debemos hacer es conocer algunos de los elementos básicos de este esquema, que son los siguientes:



Contenedor: Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles.

Eje principal: Los contenedores flexibles tendrán una orientación principal específica.

Eje secundario: perpendicular a la principal.

Ítem: Cada uno de los hijos que tendrá el contenedor en su interior.

FLEX CSS

En definitiva, es un elemento de HTML que tiene sus hijos configurados para usar flexbox, debiendo incluir dicho elemento la propiedad display:flex. Partimos del siguiente supuesto:

```
<div class="container"> <!-- Flex container -->
  <div class="item item-1">1</div> <!-- Flex items -->
  <div class="item item-2">2</div>
  <div class="item item-3">3</div>
</div>
```

```
.container {
  display: flex;
  flex-direction: column;
  background: steelblue;
}

.item {
  background: grey;
}
```

Modalidades

Display: flex se comporta como un elemento en bloque

Display: inline-flex, se comporta como un elemento en linea

Direccion de los ejes

Flex-direction: row,columna,row-reverse,columna-reverse

FLEX CSS

Multilínea

En general, flex se suele utilizar para estructuras de una sola dimensión, es decir, contenedores que sólo van en una dirección. Sin embargo, existe una propiedad denominada flex-wrap donde tenemos las opciones de nowrap,wrap y wrap-reverse. Agregaremos 6 ítems para ver sus resultados.

Para la dirección de los ejes tenemos una propiedad atajo llamada flex-flow: row wrap;

```
<div class="container"> <!-- Flex container -->
 <div class="item item-1">1</div> <!-- Flex items -->
 <div class="item item-2">2</div>
 <div class="item item-3">3</div>
 <div class="item item-4">4</div>
 <div class="item item-5">5</div>
 <div class="item item-6">6</div>
</div>
```

```
.container {
  display: flex;
  flex-wrap: wrap;
  background: steelblue;
  width: 200px;
}
.item {
  background: grey;
  width: 50%;
}
```

FLEX CSS

Huecos-gap

Existen dos propiedades recientes de flex-box que son row-gap y columna-gap, que como la propia propiedad indica establece el tamaño de un hueco entre los ítems y el elemento padre si necesidad de usar padding ni margin en los elementos hijos.

Row-gap se usa para flex-direction columna, columna-gap para flex-direction row. Se pueden usar ambas siempre que tengamos la opción de flex-wrap como wrap y en este caso si tendríamos múltiples columnas y filas.

Existe un atajo que es gap, donde podemos indicar el espacio o hueco para las filas y columnas.

```
.container {
  /* 2 parámetros: <row> <column> */
  gap: 4px 8px;
  /* Equivalente a */
  row-gap: 4px;
  column-gap: 8px;

  /* 1 parámetro: usa el mismo para ambos */
  gap: 4px;
  /* Equivalente a */
  row-gap: 4px;
  column-gap: 4px;
}
```

FLEX CSS

Alineación

Existen tres propiedades que nos permiten alinear los ítems dentro de un flex-box:

Propiedad	Valor	Eje
Justify-content	Start|end|center|space-between|space-around|space-evenly	**1**
Align-ítems	Start|end|center|strech|baseline	**2**
Align-content	Start|end|center|space-between|space-around|space-evenly|strech	**2**

De todas estas, debemos centrarnos en las dos primeras, ya que la tercers sól tiene sentido si tenemos un flex multilínea.

justify-content: Se utiliza para alinear los ítems del eje principal.

align-items: Usada para alinear los ítems del eje secundario.

align-content: Se utiliza para alinear el contenido del eje secundario entre líneas.

FLEX CSS

Justify-content

Sirve para colocar los ítems de un contenedor mediante una disposición concreta a lo largo del eje principal.

Valor	Descripcion
Start	Agrupa los elementos al inicio del eje principal
End	Agrupa los elementos al final del eje principal
Center	Agrupa los ítems al centro del eje principal
Space-between	Distribuye los ítems dejan espacio entre ellos
Space-around	Idem al anterior pero el espacio lo deja alrededor de ellos
Space-evenly	Idem al anterior pero el espacio es igual entre todos ellos

FLEX CSS

Align-items

Se encarga de alinear los ítems en el eje secundario del contenedor. Hay que tener cuidado de **no confundir align-items con align-content**, puesto que el segundo actúa sobre cada una de las líneas de un **contenedor multilinea** (no tiene efecto si no usamos flex-wrap), mientras que **align-items** lo hace sobre **la única línea** que tiene un contenedor flex sin wrap.

Valor	Descripcion
Start	Agrupa los elementos al inicio del eje secundario
End	Agrupa los elementos al final del eje secundario
Center	Agrupa los ítems al centro del eje secundario
Stretch	Distribuye los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor
baseline	Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.

FLEX CSS

Align-content

La propiedad align-content, es un caso particular de align-items. Nos servirá cuando estemos tratando con un contenedor flex multilinea creado mediante flex-wrap.

Valor	Descripcion
Start	Agrupa los elementos al inicio del eje principal
End	Agrupa los elementos al final del eje principal
Center	Agrupa los ítems al centro del eje principal
Space-between	Distribuye los ítems dejan espacio entre ellos desde el inicio hasta el final
Space-around	Idem al anterior pero el espacio lo deja alrededor de ellos
Space-evenly	Idem al anterior pero el espacio es igual entre todos ellos

FLEX CSS

Align-content

Un ejemplo con un contenedor de 200 píxels de alto con ítems de 50px de alto y un flex-wrap establecido para tener contenedores multilinea, observemos como podemos utilizar la propiedad align-content.

```
.container {
  background: #CCC;
  display: flex;
  width: 200px;
  height: 200px;
  flex-wrap: wrap;
  align-content: end;
}
```

```
.item {
  background: #777;
  width: 50%;
  height: 50px;
}
```

FLEX CSS

Atajos en las Alineaciones

Existe una propiedad de atajo con la que se pueden establecer los valores de las propiedades align-content y justify-content de una sola vez. Dicha propiedad es place-content y funciona de la siguiente forma:

```
.container {
  display: flex;
  /* 2 parámetros */
  place-content: start end;
  /* Equivalente a... */
  align-content: start;
  justify-content: end;
  /* 1 parámetro */
  place-content: start;
  /* Equivalente a... */
  align-content: start;
  justify-content: start;
}
```

FLEX CSS

Orden de los Elementos

Para terminar, con flex, y quizás una de las propiedades más interesantes, es **order**. Se trata de una propiedad mediante la cual podemos modificar y establecer un orden de los elementos mediante números: order:1,order:2, en cada item.

```
<div id="main">
<article>Articulo</article>
 <nav>Navegación</nav>
 <aside>Aside</aside>
</div>
```

```
#main {  display: flex;  text-align: center;}
#main > article {  flex: 1;  order: 2;}
#main > nav {  width: 200px;  order: 1;}
#main > aside {  width: 200px;  order: 3;}
```

FLEX CSS

Los Elementos

Las siguientes propiedades nos permiten dotar de flexibilidad a los elementos hijos de un contenedor flex:

propiedad	Valor	Descripción
Flex-basis	Size|content	Tamaño base de los elementos antes de aplicar una variación sobre los mismos
Flex-grow	0|number	Indica el crecimiento de un elemento, siempre que no ocupe el 100% del padre
Flex-shrink	1|number	Factor de decrecimiento siempre que ocupe el 100% del padre.
Flex	Ninguno basis|grow|shrink	

```
.item {
  /* flex: <flex-grow> <flex-shrink> <flex-basis> */
  flex: 1 3 35%;
  /* Equivalente a... */   flex-grow: 1;   flex-shrink: 3;   flex-basis: 35%; }
```

FLEX CSS

Animación usando transform

```
<p>Mueve el ratón sobre los elementos:</p>
<div class="container">
  <div class="item">Elemento 1</div>
  <div class="item">Elemento 2</div>
  <div class="item">Elemento 3</div>
</div>
```


```
.container {
  display: flex;
  justify-content: center;
  align-items: center; height: 150px;
  gap: 5px;
  background: grey;
}

.item {
  flex: 0 1 15%;
  padding: 20px;
  background: black;
  color: white;
  transition: all 0.5s;
}

.item:hover {
  flex: 1 1 10%;
  background: indigo;
}
```

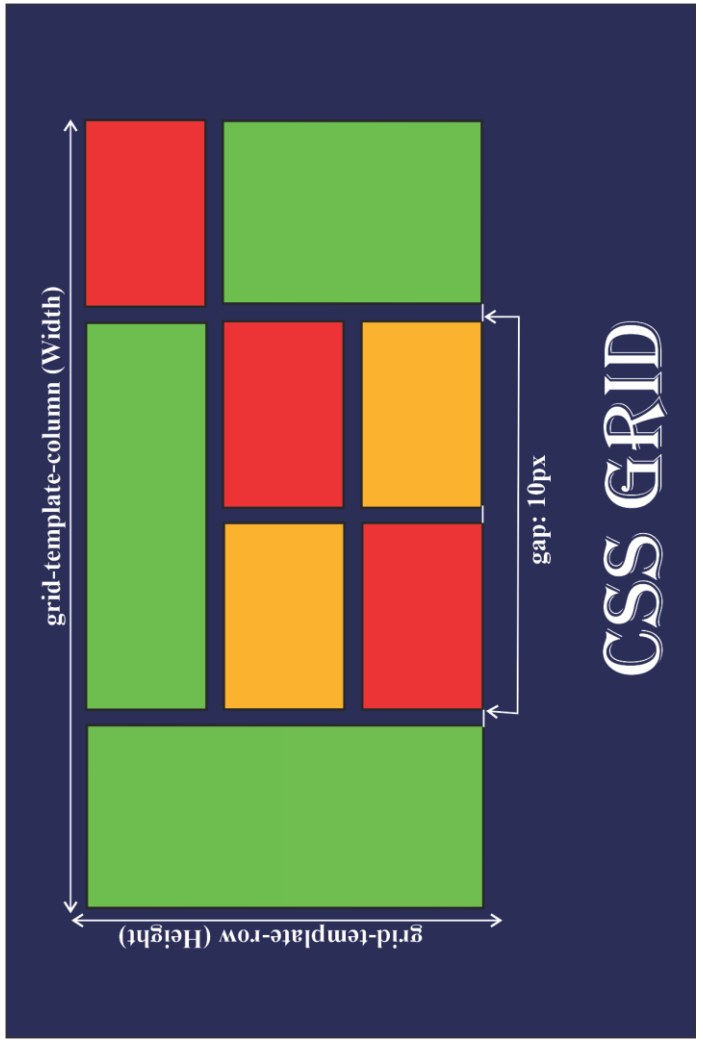
GRID CSS

Uno de los procesos más problemáticos y frustrantes de CSS, sobre todo para cuando empecé en su momento, era el de colocar y distribuir los elementos a lo largo de una página, mortal de necesidad.

Mecanismos como posicionamiento, floats o elementos en bloque o en línea, no eran suficientes o eran muy complejos para crear un layout.

El sistema de elementos flexibles Flex es una gran mejora, sin embargo, está orientado a estructuras de una sola dimensión, y puede ser laborioso crear estructuras más complejas.

Con el paso del tiempo, muchos frameworks CSS y librerías comenzaron a utilizar un **sistema basado en un grid** donde se definía una cuadrícula, a la que era posible darle tamaño, posición o colocación, cambiando el nombre de sus clases.



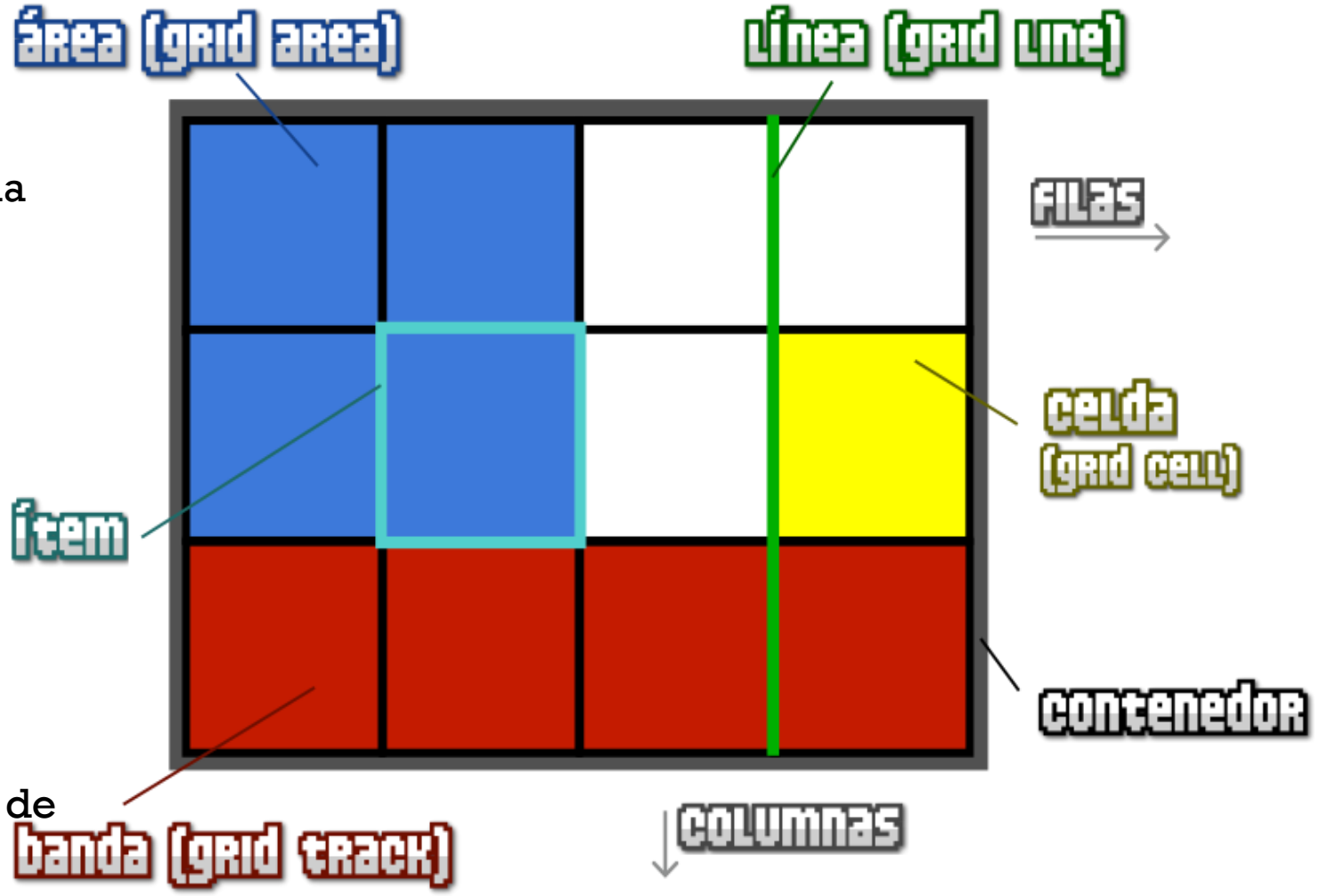
GRID CSS

Contenedor: El elemento padre contenedor que definirá la cuadrícula o rejilla.

Ítem: Cada uno de los hijos que contiene la cuadrícula (elemento contenedor).

Celda (grid cell): Cada uno de los cuadritos (unidad mínima) de la cuadrícula.

Area (grid area): Región o conjunto de celdas de la cuadrícula.



Banda (grid track): Banda horizontal o vertical de celdas de la cuadrícula.

Línea (grid line): Separador horizontal o vertical de las celdas de la cuadrícula.

GRID CSS

Trabajemos con el siguiente ejemplo:

```
<div class="grid"> <!-- contenedor -->
  <div class="item item-1">Item 1</div> <!-- cada uno de los ítems del grid -->
  <div class="item item-2">Item 2</div>
  <div class="item item-3">Item 3</div>
  <div class="item item-4">Item 4</div>
</div>
```

Para activar la cuadrícula grid hay que utilizar sobre el elemento contenedor la propiedad **display** y especificar uno de los dos valores que queramos utilizar: **grid (forma de bloque) o inline-grid (los ítems en línea).** Este valor influye en como se comportará la cuadrícula con el contenido exterior. El primero la cuadrícula aparece encima/debajo del contenido exterior (en bloque) y el segundo la cuadrícula aparece a la izquierda/derecha (en línea).

GRID CSS

En Grid CSS, la forma principal de definir una cuadrícula es indicar el tamaño de sus filas y sus columnas de forma explícita. Para ello, sólo tenemos que usar las propiedades CSS **grid-template-columns y grid-template-rows:**

```
.grid {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
}
```

Esto significa que, a priori, tendríamos una cuadricula o grid de 4 celdas en total.

Pueden ir cambiando el grid-template cada valor para ver como se comporta. Sería curioso hacer un programita en JS que vayáse cambiando estas propiedades.

GRID CSS

En el ejemplo anterior he utilizado píxels como unidades de las celdas de la cuadrícula, sin embargo, también podemos utilizar otras unidades (o incluso combinarlas): porcentajes, la palabra clave auto (que obtiene el tamaño restante) o la unidad especial de **grid fr (fracción restante),** es la más usada:

```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```

Dos columnas: Mismo tamaño de ancho para cada una.

Dos filas: La primera fila ocupará el doble (2fr) que la segunda fila (1fr).

GRID CSS

En algunos casos, en las propiedades **grid-template-columns y grid-template-rows** podemos necesitar indicar las mismas cantidades varias veces, por ello es mejor usar la función **repeat()** para indicar repetición de valores, señalando el número de veces que se repiten y el tamaño en cuestión.

```
.grid {
  display: grid;
  grid-template-columns: 100px repeat(4, 50px) 200px;
  grid-template-rows: repeat(2, 1fr 2fr);
  /* Equivalente a... */
  grid-template-columns: 100px 50px 50px 50px 50px 200px;
  grid-template-rows: 1fr 2fr 1fr 2fr;
}
```

GRID CSS

La función **minmax()** se puede utilizar como valor para definir rangos flexibles de celda. Si establecemos un rango, por ejemplo, grid-template-column: minmax(200px, 500px), estaremos indicando que la celda de columna indicada, **tendrá un tamaño de 500px, salvo que redimensionemos la ventana del navegador y la hagamos más pequeña, en cuyo caso, el tamaño de la celda podría ir disminuyendo hasta 200px**, medida en la cuál se quedaría como mínimo.

```
.container {
  display: grid;
  grid-template-columns: repeat(2, minmax(400px, 600px));
  grid-template-rows: repeat(2, 1fr);
  gap: 5px;
}
.item {
  background: black;
  color: white;
  padding: 1em;
}
```

GRID CSS

En la función repeat() es posible utilizar las palabras claves **auto-fill o auto-fit** para indicar al navegador que queremos que rellene o ajuste el contenedor grid con múltiples elementos hijos dependiendo del tamaño del viewport.

```
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  background: grey;
  gap: 10px;
}
.item {
  background: blue;
  color: #fff;
  font-size: 2rem;
}
```

la propiedad grid-template, es un atajo para grid-template-columns y en grid-template-rows.

GRID CSS

```
.grid {
  column-gap: 100px;
  row-gap: 10px;
}
.grid {
  /* gap: <row-gap> <column-gap> */
  gap: 20px 80px;
  /* Equivalente a... */
  row-gap: 20px;
  column-gap: 80px;
  gap: 40px;
  /* Equivalente a... */
  row-gap: 40px;
  column-gap: 40px;
}
```

OJO: Antiguamente, las propiedades column-gap, row-gap y gap eran conocidas como grid-column-gap, grid-row-gap y grid-gap, por lo que aún puede que encuentres información antigua que las use.

Veamos un ejemplo de su uso.

GRID CSS

Propiedades de alineación

Para **colocar y ajustar** nuestra cuadrícula grid o ajustar los ítems usamos las siguientes propiedades:

justify-items: Alinea los elementos (hijos) en **horizontal** (eje principal) dentro de cada celda.

align-items: Alinea los elementos (hijos) en **vertical** (eje principal) dentro de cada celda.

justify-content: Alinea el contenido (la cuadricula) en horizontal en el contenedor padre.

align-content: Alinea el contenido (la cuadricula) en vertical (eje secundario) en el contenedor padre.

En el caso de que queramos que **uno de los ítems hijos** tenga una distribución diferente al resto, podemos aplicar justify-self (eje principal) o align-self (eje secundario).

Si vamos a crear estructuras grid donde utilicemos los pares de propiedades justify-ítems//align-items por un lado, justify-content//align-content por otro, e incluso justify-self //align-self por otro, podemos usar los **atajos place-ítems (align-justify ítems), place-content(align-justify content) o bien place-self. Veamos un ejemplo.**