

Lenguajes de marcas y sistemas de gestión de la información



UT03 – CSS

1 – Introducción – Sintaxis

De la semántica a la presentación

Ya hemos estudiado HTML, y hemos visto que:

- Es un lenguaje de marcado, que trabaja en modo texto
- Usa etiquetas (tags) que se añaden al contenido para estructurarlo
- Se centra en la semántica, en el significado y organización del contenido, no en el aspecto

Aunque se centra en el contenido, hemos podido ver que por defecto muestra ciertos elementos con cierto aspecto, por ejemplo:

- Encabezados (h1 – h6) en negrita y con tamaño en función de su nivel.
- Elemento address en cursiva (o itálica).

Vamos a aprender a definir el aspecto de los elementos HTML que no tienen estilo, y a modificar el aspecto por defecto de aquellos que ya lo tengan.

CSS

CSS es un lenguaje, no de programación, pero sigue siendo un lenguaje.

Es un lenguaje declarativo (no imperativo), que define (declara) el aspecto de los elementos de una página web.

También se puede utilizar CSS con otros tipos de documentos, no solo con HTML. También se puede usar con XML, por ejemplo.

Creado el año 1996, su estándar, como el del HTML, lo define y lo mantiene el W3C (World Wide Web consortium).



CSS

Se han publicado, desde 1996, tres especificaciones de CSS:

- CSS 1 – La primera versión oficial. Diciembre de 1996.
- CSS 2 – Mayo de 1998.
- CSS 2.1 – Junio de 2011.
- CSS 3 – No se puede dar una fecha específica, porque cambió la forma de especificar el estándar.

CSS había crecido mucho, y era poco operativo mantener todas las características en una sola especificación.

Así que se partió la especificación en múltiples especificaciones más pequeñas, y se han ido añadiendo otras.

CSS

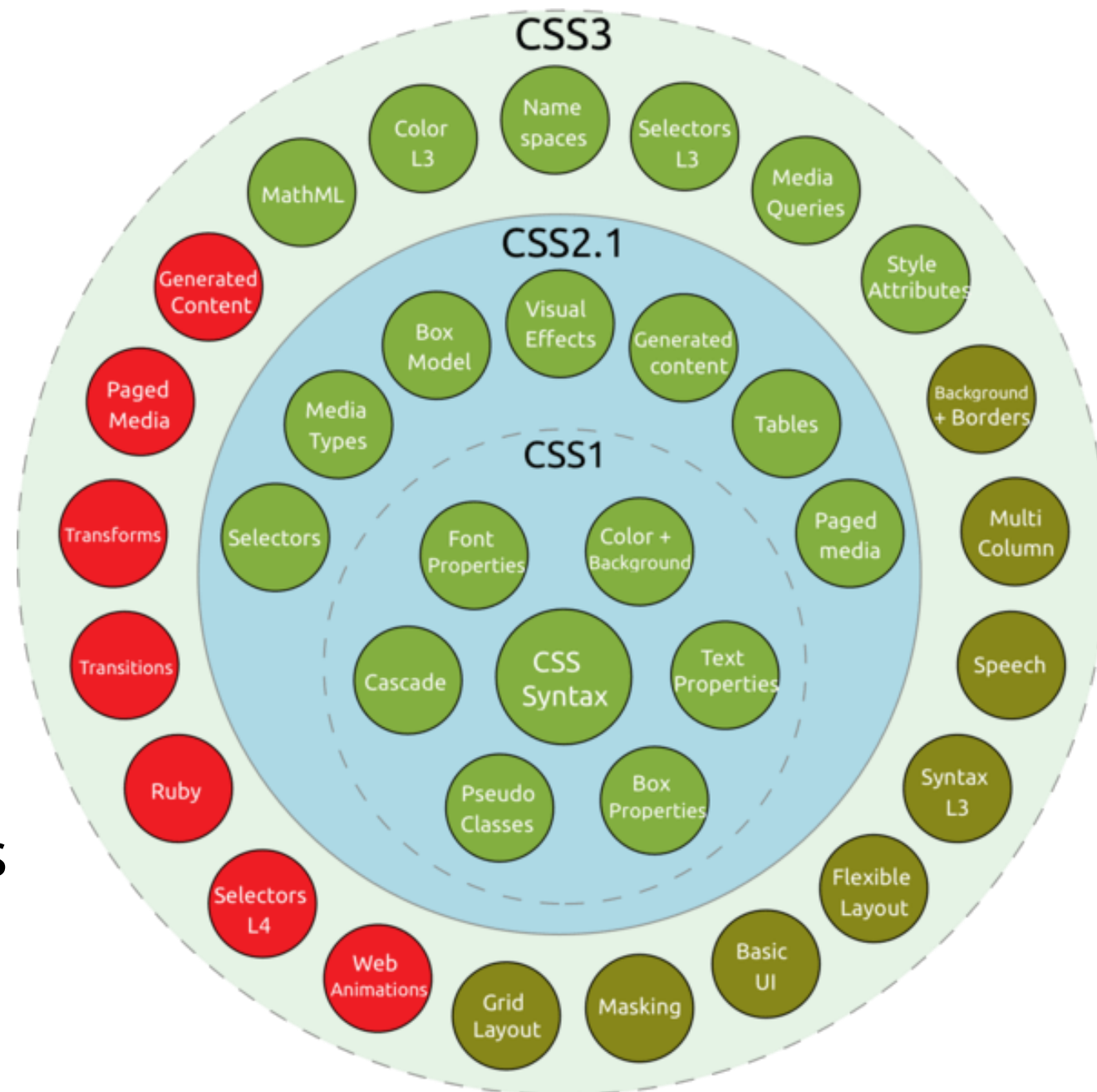
No habrá una especificación CSS 4.

Lo que ocurre es que cada una de las especificaciones "menores", denominadas "módulo", crece de forma independiente.

Por ejemplo, el módulo de color ya tiene propuesto un candidato a estándar para el nivel 4, y hay un borrador del nivel 5.

Para buscar distintos módulos y sus versiones (no solo CSS):

<https://www.w3.org/TR/>



CSS

CSS son las siglas de "Cascading Style Sheets", hojas de estilo en cascada.

Con CSS se crean hojas de estilo (style sheets) para definir la presentación de un documento HTML cuando se visualiza en un navegador.

Antes de la existencia y generalización de CSS, toda la presentación, el aspecto, se debía realizar usando ciertos atributos y elementos de HTML.

La mayoría de estos atributos y elementos están obsoletos o desaconsejados hoy día, y por eso no los hemos visto en la parte de HTML. Por ejemplo, el elemento "<center>", o los atributos "bgcolor", "align", "width" o "height".

CSS

La palabra "cascading" o "en cascada" se refiere a la forma en la que aplican los estilos sobre los elementos.

Los estilos para un elemento se aplican uno detrás de otro, de forma que sobre los cambios que introduce un estilo, se aplican los del siguiente, luego el siguiente, y así sucesivamente.

El orden en que se aplican los estilos se determina por cuatro factores:

- Posición y orden del estilo. El orden en que aparecen los estilos.
- Especificidad. Hay estilos que aplican a muy pocos elementos html, y otros a muchos o incluso a todos los elementos html de una página.
- Origen: si es un estilo en línea o viene de una hoja de estilos externa.
- Importancia: podemos elegir priorizar unos estilos frente a otros

Ventajas de usar CSS

Las principales ventajas de usar CSS frente a otros sistemas de formato de documentos son:

- Gestión centralizada de los estilos de un solo punto. Esto implica:
 - Ahorro de tiempo
 - Más fácil mantenimiento y detección de errores
- Carga más rápida de las páginas
- Más control sobre los estilos que con sólo HTML. Mucho más flexible.
- Mejor soporte por distintos dispositivos, incluyendo navegadores dentro de dispositivos móviles. Se puede comprobar el soporte de los navegadores a las distintas características en <https://caniuse.com>

Cosas que se pueden hacer con CSS

Con CSS se pueden hacer, entre otras, las siguientes cosas:

- Modificar el aspecto por defecto de los elementos HTML.
- Transformar elementos cambiando su tamaño (escalado), rotándolo, sesgándolo, ...
- Crear animaciones y efectos sin el uso de Java o JavaScript.
- Hacer que una web sea responsive, haciendo que se adapte a los móviles, o a otros dispositivos de mayor tamaño.
- Crear versiones para imprimir de una página.
- ...

Cómo usar CSS

Hay tres formas de utilizar CSS en un documento HTML:

Archivo CSS externo.

- La más habitual y recomendable.
- Se usa un elemento `<link rel="stylesheet">`, que enlaza a un fichero .css independiente. Debe ubicarse dentro de la cabecera (`<head>`).

Bloque de estilos en la página.

- Se usa un elemento `<style>`. Debe ubicarse en la cabecera.

Estilos en línea

- Se usa el atributo `style` en el elemento HTML.

Cómo usar CSS – Fichero externo

Se enlaza con un elemento link:

```
<link rel="stylesheet" href="estilos.css" />
```

Es la forma recomendada de utilizar CSS. Se recomienda poner estos elementos lo antes posible dentro de <head>, para que se apliquen antes de que se carguen scripts contenido.

Se puede enlazar más de una hoja de estilos. En este caso, el orden es importante. Los estilos se aplican en el orden en que se declaran, y los últimos en declararse pueden sobrescribir los que se declararon primero.

La principal ventaja de las hojas de estilo externas es que centralizamos el aspecto de la web, y con modificar un solo fichero podemos modificar el aspecto de múltiples páginas.

Cómo usar CSS – Bloque de estilos

Se declaran los estilos con un elemento `<style>`

```
<style>  
    div { background: red; color: white;}  
</style>
```

Puede servir para cosas puntuales, pero no es lo más recomendado.

Puede provocar una repetición de reglas de estilo en distintas páginas

Aumenta el peso (tamaño) de los documentos HTML.

También es más lenta la carga de la página, porque no se cachean.

`<style>` tiene que estar dentro de `<head>`, pero funcionará da igual donde lo pongamos. El problema de colocarlo fuera de `<head>` es que puede producirse un "flash" de contenido sin estilos, hasta que se cargan.

Cómo usar CSS – Estilos en línea

Se usa el atributo global style en los elementos html

```
<p style="color:red">Este texto será rojo</p>
```

El atributo style, global, se puede aplicar a cualquier elemento html.

Se pueden poner varios estilos separados por punto y coma ";"

```
<p style="background:gray; color:red">Este texto será rojo</p>
```

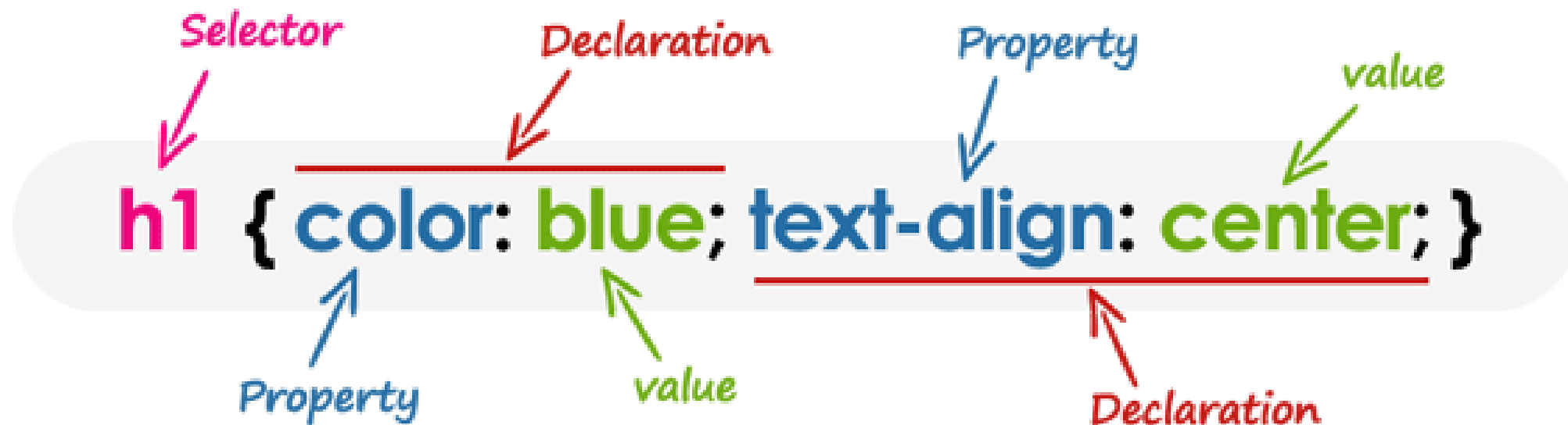
De nuevo, puede servir para cosas puntuales, pero no es lo más recomendado. Siempre se prefiere el uso de una hoja de estilos externa.

Los problemas potenciales son los mismos que con el elemento <style>, puede provocar una repetición de reglas de estilo en distintas páginas y aumenta el peso (tamaño) de los documentos HTML.

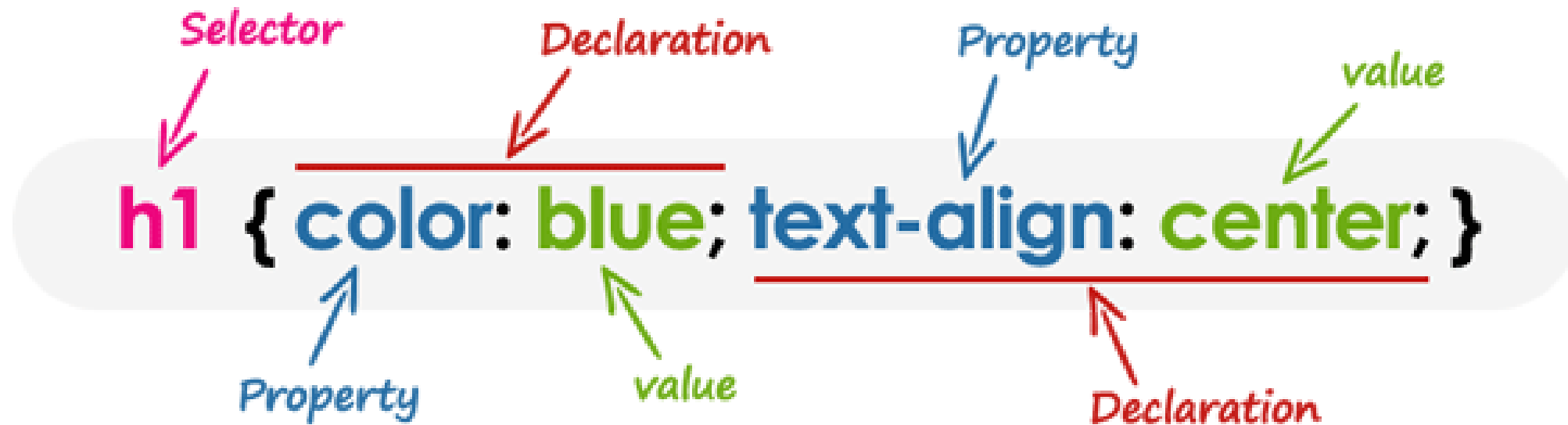
Sintaxis CSS

Una hoja de estilos está formada por reglas, que el navegador aplica a los diferentes elementos.

Una regla CSS tiene dos partes, un selector y una o más declaraciones. Si hay más de una declaración se separan por punto y coma.



Sintaxis CSS



El selector indica a que elemento o elementos del documento HTML se debe aplicar el estilo.

A continuación, se ponen llaves para agrupar todas las declaraciones asociadas al selector.

Cada una de las declaraciones tienen la forma “propiedad: valor”. Las declaraciones pueden estar en la misma línea o en varias.

Sintaxis CSS – Comentarios

Los comentarios en CSS comienzan con “/*” y se cierran con “*/”. Igual que los comentarios de varias líneas en Java.

```
/* Comentario CSS en una línea */  
p { font-weight: bold; }
```

```
/* Comentario CSS  
de varias líneas */  
p { font-weight: bold; }
```

No se pueden hacer comentarios de una línea al estilo Java (//). Si queremos añadir comentarios a una declaración, lo haremos con /**/

```
p {  
    /* Comentario a la declaración */  
    font-weight: bold;  
}
```


Sintaxis CSS - Ejemplos

```
/* Establece fuente para el cuerpo de la página */  
body { font-family: Arial, Helvetica, sans-serif; }
```

```
/* Todos los párrafos con fondo morado y texto azul claro */  
p {  
    color: aqua;  
    background-color: rebeccapurple;  
}
```

```
/* Elementos de lista (li) con borde, una separación del contenido  
respecto al borde y un margen que los separa del resto de elementos. */  
ul li {  
    border: 1px solid red;  
    padding: .1em;  
    margin: 2px 10px;
```

Sintaxis CSS – Reglas @ (at-rules)

Hay ciertas directivas CSS que empiezan con el símbolo @ (at-rules)

Estas reglas o directivas tienen múltiples y variadas utilidades, y veremos algunas más adelante. De momento, vamos a ver sólo @import.

@import permite importar reglas desde otra hoja de estilo.

```
@import url(reset.css);
```

```
@import url(colores.css);
```

```
@import url(https://cdn.jsdelivr.net/npm/bootstrap@5.3.2  
/dist/css/bootstrap.min.css)
```

Sintaxis CSS – Recomendaciones

Un fichero CSS puede estar formado por varios cientos de reglas.

A medida que un fichero CSS crece se hace más difícil de leer y mantener.

Al igual que en otros lenguajes, hay una serie de buenas prácticas que ayudan a leer y mantener el código CSS. Las dos más importantes:

- En cada regla CSS, escribe una declaración por línea.
- Mantén el documento indentado.

Esto, en VS Code es muy fácil de conseguir. Basta con que pidáis a VS Code que formatee el código con el menú contextual (botón derecho)

Se puede activar el formateo automático al guardar un documento. Se hace en File > Preferences > Settings y buscar "Format On Save". También puede ser útil activar "Format On Paste"

Sintaxis CSS – Convención para id y class

Los atributos id y class son muy importantes al definir reglas y selectores CSS (y JavaScript)

Aunque no es obligatorio, se recomienda usar la notación lower-kebab-case para el valor de los atributos id y class.

No recomendado:

```
<section class="newProducts">  
  <header id="newProductsHeader"><h2>Products</h2></header>  
</section>
```

Recomendado:

```
<section class="new-products">  
  <header id="new-products-header"><h2>Products</h2></header>  
</section>
```

Lenguajes de marcas y sistemas de gestión de la información



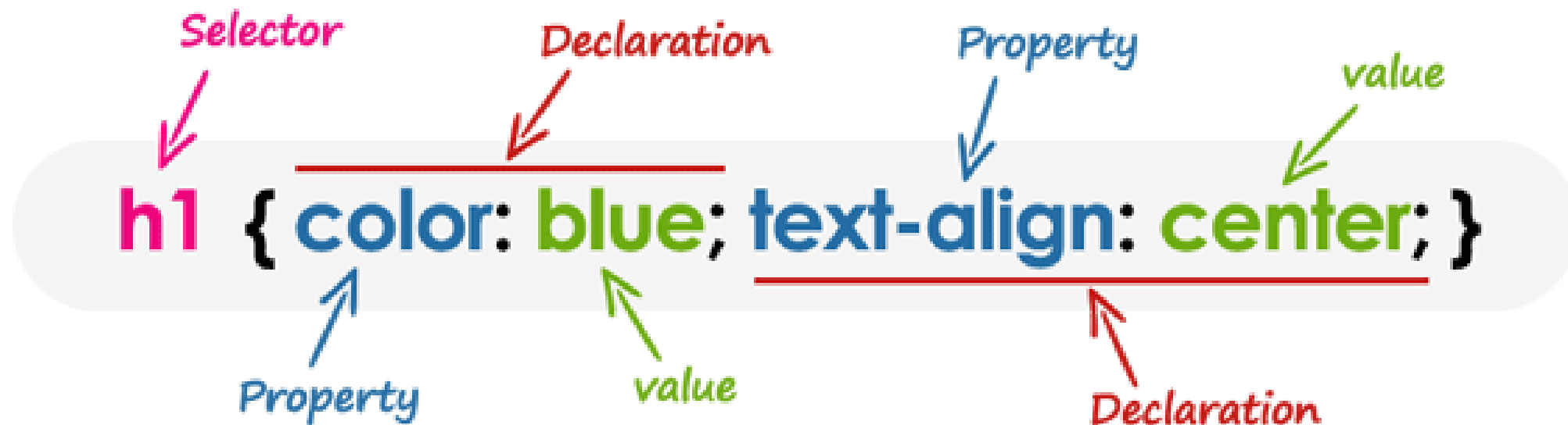
UT03 – CSS

2 – Selectores básicos – Primeras propiedades – div y span

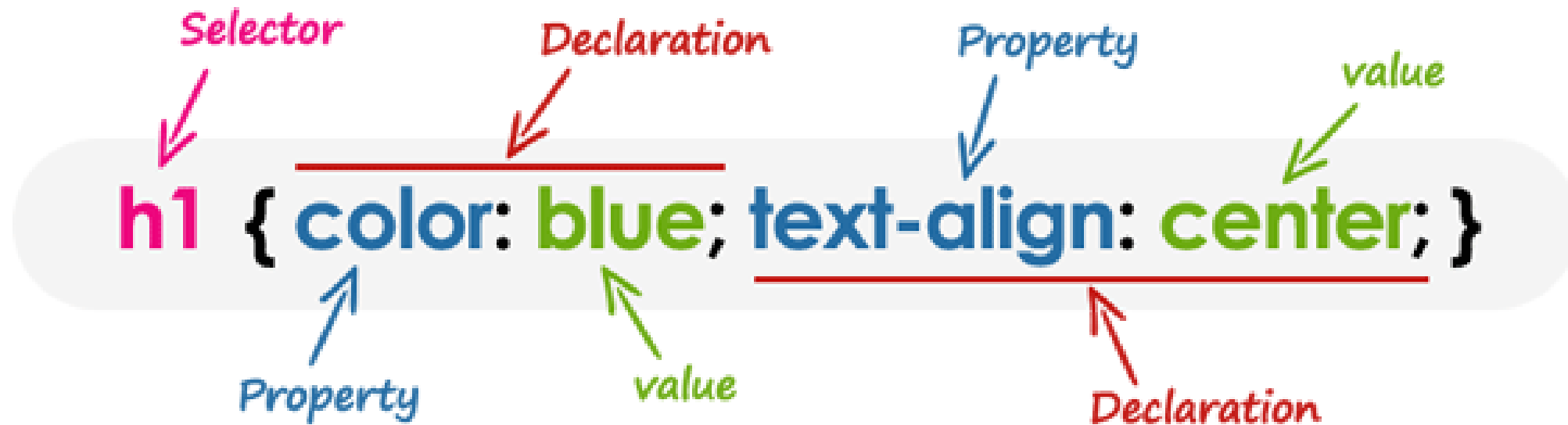
Sintaxis CSS

Una hoja de estilos está formada por reglas, que el navegador aplica a los diferentes elementos.

Una regla CSS tiene dos partes, un selector y una o más declaraciones. Si hay más de una declaración se separan por punto y coma.



Sintaxis CSS



El selector indica a que elemento o elementos del documento HTML se debe aplicar el estilo.

A continuación, se ponen llaves para agrupar todas las declaraciones asociadas al selector.

Cada una de las declaraciones tienen la forma “propiedad: valor”. Las declaraciones pueden estar en la misma línea o en varias.

Sintaxis CSS – Comentarios

Los comentarios en CSS comienzan con “/*” y se cierran con “*/”. Igual que los comentarios de varias líneas en Java.

```
/* Comentario CSS en una línea */  
p { font-weight: bold; }
```

```
/* Comentario CSS  
de varias líneas */  
p { font-weight: bold; }
```

No se pueden hacer comentarios de una línea al estilo Java (//). Si queremos añadir comentarios a una declaración, lo haremos con /**/

```
p {  
    /* Comentario a la declaración */  
    font-weight: bold;  
}
```


Selectores básicos – Universal

El selector universal se puede utilizar para que se apliquen estilos a todos los elementos.

Se utiliza el símbolo *

Ejemplo: esta regla elimina el margen y el padding (relleno) en todos los elementos. Además, cambia el espaciado entre líneas para aumentarlo un 50%.

```
* {  
    margin: 0;  
    padding: 0;  
    line-height: 150%;  
}
```

Selectores básicos – De elemento

Permite aplicar estilos a todas las ocurrencias de un elemento HTML, a todas las ocurrencias de las .

Se utiliza el nombre del elemento al que queremos aplicar el estilo.

Ejemplo: todos los párrafos con el texto en color gris.

```
p {  
    color: gray  
}
```

Ejemplo: todos los enlaces con el texto en tamaño de 30 píxeles

```
a {  
    font-size: 30px;  
}
```

Selectores básicos – Por Id

Permite especificar reglas para un elemento específico, usando su atributo "id".

id es uno de los atributos generales que cualquier elemento HTML puede tener. Y no puede haber dos elementos con el mismo id en una página.

Se usa la almohadilla (#) seguida del Id del elemento

Ejemplo: subrayado de olas en un elemento con id "enlace-google"

```
#enlace-google{  
    text-decoration: wavy;  
}
```

Ejemplo: ancho de 300 píxeles en un elemento con id "mas-ancho"

```
#mas-ancho{  
    width: 300px;  
}
```

Selectores básicos – Por clase CSS

El atributo class permite asignar a un elemento una o más clases CSS.

Ejemplo: párrafo con la clase CSS "destacado":

```
<p class="destacado">Lorem ipsum dolor sit amet</p>
```

Ejemplo: celda con las clases "dato" y "fondo-amarillo".

```
<td class="dato fondo-amarillo">3589</td>
```

En el atributo class se pueden escribir tantas clases como sean necesarias, y al elemento aplicado se le aplicarán todas las reglas definidas. Si hay reglas para varias clases, se aplicarán varias reglas.

A diferencia del id, que tiene que ser único, en un documento HTML puede haber múltiples elementos con el mismo class.

Selectores básicos – Por clase CSS

Se utiliza el punto (.) seguido del nombre de la clase CSS a la que deseamos aplicar la regla.

Según los ejemplos HTML anteriores:

Regla para la clase "destacado":

```
.destacado { font-weight: bold; }
```

Regla para las clases "dato" y "fondo-amarillo":

```
.dato { text-transform: uppercase; }
```

```
.fondo-amarillo { background-color: #ffff00; }
```

En este caso, a la celda con las dos clases CSS se le aplicarán las dos reglas, la de class "dato" y la de class "fondo-amarillo".

Selectores básicos – Elemento con clase CSS

Permite seleccionar los elementos de cierto tipo con una misma clase CSS. Se usa el nombre del elemento, seguido del punto (.) y la clase CSS.

Ejemplo: párrafo con la clase CSS "destacado":

```
<p class="destacado">Lorem ipsum dolor sit amet</p>
```

Ejemplo: celda con las clases "dato", "fondo-amarillo" y "destacado".

```
<td class="dato fondo-amarillo">3589</td>
```

Con .destacado seleccionaríamos el p y la celda. Si sólo queremos seleccionar el párrafo destacado usaríamos:

```
p.destacado { font-weight: bold; }
```

Selectores básicos – Por atributo

Permiten seleccionar los elementos que tengan cierto atributo, o que tengan un atributo con cierto valor.

Se usan los corchetes, para indicar el atributo que buscamos, y si añadimos un = dentro, podemos buscar un valor de atributo específico.

Se puede combinar con el nombre de elemento u otros selectores

Ejemplo: Todos los elementos con atributo disabled:

```
[disabled] { ... }
```

Ejemplo: los enlaces que tienen un atributo title

```
a[title] { ... }
```

Ejemplo: los enlaces que abren en ventana nueva (target="_blank")

```
a[target="_blank"] { ... }
```

Regla para varios selectores

Si tenemos que aplicar las mismas declaraciones a varios selectores, se pueden separar estos (los selectores) por comas.

Por ejemplo, poner letra de color rojo a los p, a los enlaces con target _blank, y a los li con class "destacado":

```
p,  
a[target="_blank"],  
li.destacado {  
    ...  
}
```

Aunque no es obligatorio, se recomienda poner cada selector en una línea independiente, para que sea más fácil de leer.

Combinando selectores

Además del ".", hay otras formas de combinar selectores, según lo que estemos buscando. Se usa un símbolo entre dos selectores para definir la relación de los elementos seleccionados.

Vamos a ver los siguientes: descendiente, hijo, hermanos, hermano adyacente. Para ello vamos a basarnos en este HTML:

```
<nav>
<ul>
  <li class="item-1"><a href="pagina1.html">Página 1</a></li>
  <li class="item-2"><a href="pagina2.html">Página 2</a></li>
  <li class="item-3"><a href="pagina3.html">Página 3</a></li>
  <li class="item-4"><a href="pagina4.html">Página 4</a></li>
</ul>
</nav>
```

Combinando selectores – Descendiente

Permite seleccionar elementos que son descendientes (no necesariamente hijos directos) de otros elementos.

Se usa el espacio para seleccionar combinar los selectores.

Según el HTML anterior, para seleccionar los enlaces dentro de la navegación:

```
nav a { color: green; }
```

O los li dentro de la navegación:

```
nav li { color: green; }
```

Aunque los enlaces o los elementos de lista no sean hijos directos de la navegación, se seleccionan porque son descendientes (nietos los li, y bisnietos los a).

Combinando selectores – Hijo

Permite seleccionar elementos que son descendientes (no necesariamente hijos directos) de otros elementos.

Se usa el símbolo ">" para combinar los selectores.

Según el HTML anterior, para seleccionar los enlaces dentro de la navegación:

```
nav > a { color: green; } /* No selecciona nada */
```

O los li dentro de UL:

```
ul > li { color: green; }
```

Este selector permite afinar más lo que queremos seleccionar que el selector descendiente.

Combinando selectores – Hermano

Permite seleccionar elementos que son hermanos, están al mismo nivel que un elemento, y a continuación de este.

Se usa el símbolo "~" para combinar los selectores.

Según el HTML anterior, para seleccionar los "li" hermanos del "li" con class "item-2":

```
li.item-2 ~ li
```

Selecciona TODOS los "li" que son hermanos del "li" con clase "item-2", y que están DESPUÉS que él. Los anteriores NO se seleccionan, aunque sean también sus hermanos, porque están al mismo nivel.

Importante, en este caso todos los elementos son li, pero pueden ser hermanos elementos con distinto tag.

Combinando selectores – Adyacente

Permite seleccionar elementos que son hermanos, están al mismo nivel que un elemento, pero justo al lado del elemento deseado, y a continuación de este.

Se usa el símbolo "+" para combinar los dos selectores.

Según el HTML anterior, para seleccionar el elemento adyacente al "li" con class "item-2", y posterior a él:

```
li.item-2 + li
```

Selecciona SOLO el siguiente "li" al "li" con class "item-2", es decir, el li con class "item-3". No item-1 ni item-4. Igual que ~, sólo selecciona el adyacente que está DESPUÉS que el primer selector. El anterior NO se selecciona, aunque sea también sus hermano.

Algunas propiedades básicas

Aunque iremos viendo las propiedades según su ámbito de aplicación, vamos a ver algunas que pueden ser útiles para ir practicando.

Para fuentes:

- Font-family: cambia la fuente del texto. Puede ser una o varias fuentes.
- Font-size: cambia el tamaño del texto.
- Font-weight: cambia el peso del texto (negrita/bold o ligera/light)

Para fondos y bordes:

- Background-color: cambia el color de fondo de un elemento.
- Border: establece el borde del elemento
- Width, height: ancho y alto de un elemento. No siempre aplican, veremos más adelante cuándo sí y cuando no.

Elementos sin carga semántica

Hasta ahora hemos escrito HTML que siempre era semántico. Cada elemento tiene un propósito y un significado.

Pero para maquetar, para dar estilos y distribuir el contenido, a veces hay que agrupar elementos dentro de un contenedor, o hay que diferenciar una parte de texto sin dar significado.

Para esto podemos usar:

- Elemento en bloque "div". Su nombre viene de "división". Sirve para agrupar elementos HTML que necesitamos juntos para maquetar.
- Elemento en línea . Su nombre viene del verbo inglés "span" (abarcarse, extenderse). Sirve para rodear una parte de texto u otros elementos en línea para agruparlos y poder dar estilos.

Lenguajes de marcas y sistemas de gestión de la información

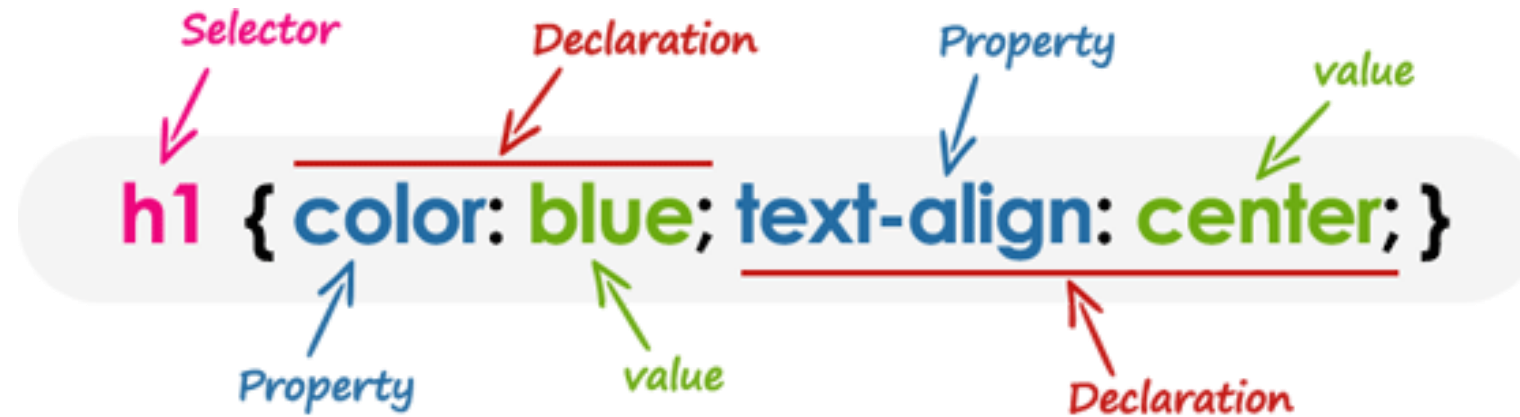


UT03 – CSS

3 – Variables y funciones

Sintaxis CSS

Reglas CSS: selector y una o más declaraciones



Selector indica a que elemento o elementos del documento HTML se debe aplicar el estilo.

Cada una de las declaraciones tienen la forma “propiedad: valor”. Las declaraciones pueden estar en la misma línea o en varias.

Las declaraciones indican qué estilos queremos aplicar a los elementos seleccionados.

Variables CSS (CSS Custom Properties)

A veces en CSS tenemos que repetir un valor numérico muchas veces.

Por ejemplo, imaginemos que queremos utilizar el ancho 200px (200 píxeles) en múltiples reglas.

Tendremos una hoja de estilos con el valor en muchos sitios. Si tenemos que cambiarlo, por ejemplo, por 225px, tendremos que reemplazarlo en múltiples reglas. Pero teniendo cuidado de no reemplazar otros 200px que se refieran a otra cosa.

O podemos usar variables CSS, que permiten:

- Definir un valor en un único punto, y reutilizarlo muchas veces.
- Cambiar el valor siempre que queramos sin miedo a efectos laterales.
- Tener un código CSS más semántico, más fácil de leer

Variables CSS – Definición

Para definir una variable CSS, tenemos que definirla:

- Dentro de una regla para el elemento que queremos que aplique. En muchas ocasiones este elemento será toda la página (html o :root).
- Con el nombre que queramos, pero siempre precedido de dos guiones (--)

Ejemplo: variables para toda la página que definen el ancho de los botones medianos, y el fondo de todos los botones:

```
html {  
    --btn-back-color: teal;  
    --btn-width-mediano: 200px;  
}
```

Los nombres de las variables los hemos elegido, no son propiedades CSS existentes, aunque pueda parecerlo.

Variables CSS – Uso

Para usar una variable CSS se utiliza la función "var".

Esta función recibe como parámetro el nombre de la variable CSS (incluidos los guiones --) y devuelve el valor asignado.

Ejemplos:

```
button {  
    background-color: var(--btn-back-color);  
}
```

```
.mediano {  
    width: var(--btn-width-mediano);  
}
```

El navegador usará los valores asignados previamente a las variables cuando tenga que calcular los estilos de los distintos elementos.

Variables CSS – Ámbito (scope)

Al definir la variable, se puede establecer su ámbito, aquellos elementos en los que tendrá valor.

Ejemplo: variables definidas para distintos ámbitos:

- Para los elementos de tipo blockquote

```
blockquote { --blockquote-right-margin: 1em; }
```

- Para los elementos con class "cita-autor"

```
.cita-autor { --margin-left: 10px; }
```

- Para cualquier elemento de la página (todo el html)

```
html { --back-color: teal; }
```

Variables CSS – Ámbito (scope) – Aplicación

Esto permite sobrescribir valores de variables. Por ejemplo, con este HTML y CSS, podemos tener valores distintos para una misma variable en cada una de las secciones de un documento HTML.

```
<section class="sect1">...</section>
<section class="sect2">...</section>
<section class="sect3">...</section>
```

```
section { --section-padding: 100px; }
.sect1 { --section-padding: 80px; }
.sect3 { --section-padding: 80px; }
```

La variable
"--section-padding" valdrá
100px para todas las
secciones, pero para la
sección 1 y 3 su valor será de
80px, por la cascada.

Veremos esto de la cascada
más adelante.

Funciones CSS

Una función CSS nos permiten realizar cálculos dentro de nuestro código CSS, como si de un lenguaje de programación se tratara.

Hay funciones para múltiples usos: matemáticas, control de tiempo en animaciones, para transformar elementos, para aplicar filtros gráficos a los elementos, para calcular colores, etc.

Ejemplos de alguna función matemática:

- `calc()`. Operaciones con dimensiones. Ejemplo: `calc(0.2rem + 50px)`
- `max()` y `min()`. Máximo o mínimo de dos valores.
- `mod()`. Módulo, resto de la división entera.
- `attr()`. Extrae el valor de un atributo del elemento en que se aplica la regla. Sólo se puede usar en ciertas situaciones que ya veremos.

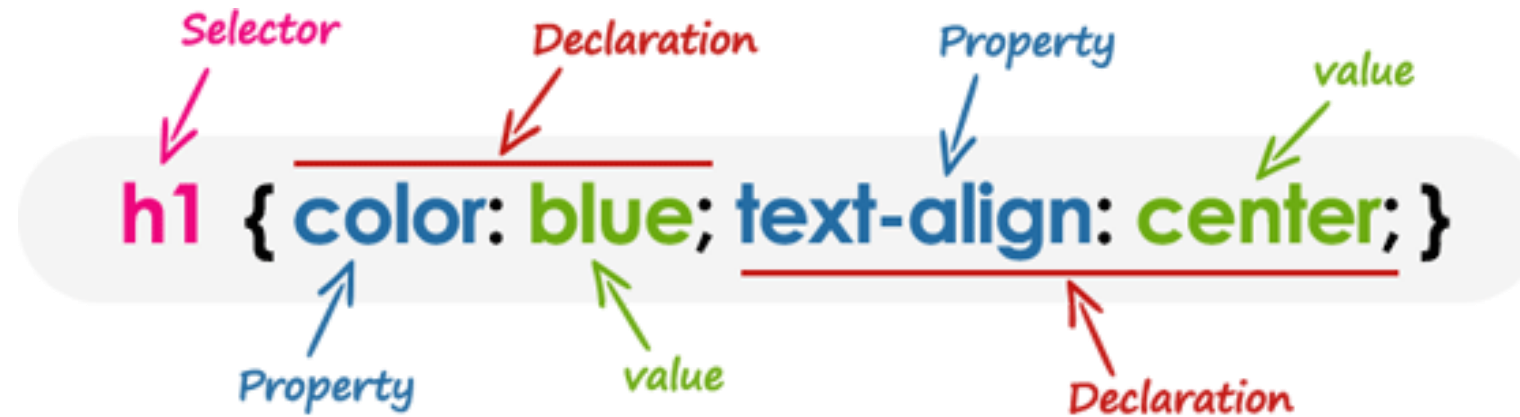
Lenguajes de marcas y sistemas de gestión de la información



UT03 – CSS 4 – Colores

Sintaxis CSS

Reglas CSS: selector y una o más declaraciones



Selector indica a que elemento o elementos del documento HTML se debe aplicar el estilo.

Cada una de las declaraciones tienen la forma “propiedad: valor”. Las declaraciones pueden estar en la misma línea o en varias.

Las declaraciones indican qué estilos queremos aplicar a los elementos seleccionados.

Propiedades más habituales

Las dos propiedades más habituales relacionadas con el color son:

- `color`: indica el color de texto del contenido de un elemento
- `background-color`: indica el color de fondo de un elemento.

Los colores de texto (`color`) se heredan, y los colores de fondo (`background-color`) no se heredan. Esto significa que, si ponemos un color y un color de fondo a un elemento, sus descendientes:

- Tendrán todos (salvo contadas excepciones) el mismo color de texto.
- No tendrán el mismo color de fondo. Esto es más difícil de ver porque si están dentro del elemento, parecerá que tienen ese color de fondo.

Forma de codificar el color

Los colores, en general, no específicamente en HTML y CSS, se pueden codificar de varias formas:

- RGB. Combinación rojo, verde y azul (Red, Green, Blue). Posiblemente la más utilizada en ordenadores (programación, css, ...)
- HLS. Tono, brillo y saturación (Hue, Brightness, Saturation).
<https://www.uifrommars.com/que-es-hsl/>
- HWB: Tono, blancura, oscuridad (Hue, Witeness, Blackness)
<https://dev.to/adrianbenavente/hwb-un-mordisco-del-futuro-de-css-46j8>
- CMYK. Combinación de cian, magenta, amarillo y negro (Cyan, Magenta, Yellow, black). Usada sobre todo en impresión en papel.
- Otros: LAB, LCH, Pantone, etc..

Colores en CSS

Hay varias formas de especificar un color en CSS:

- Colores por nombre. red, green , teal, slategray, etc. Muchos valores, pero irremediablemente limitados. No podemos usar cualquier color del espectro.
- Codificación RGB. Va precedida de la almohadilla (#):
 - Hexadecimal: #1CB2E4. Cada par de dígitos hexadecimales es un valor de 0 a 255 para el color. Las dos primeras R, las dos siguientes G, las dos últimas B.
 - Hexadecimal abreviado: #3B5. Cuando se repiten los dígitos hexadecimales en cada componente R, G y B. #3B5 = #33BB55

Colores en CSS

- RGB con canal alfa.
 - El canal alfa se añade a un color para indicar su índice de transparencia.
 - Por defecto, los colores son opacos, sin transparencia, con un alfa o alpha de valor 1 o 100%. Si ponemos un alfa de 0%, o simplemente 0, tendríamos un color transparente. Un alfa de 50% o 0.5 (con .) sería semitransparente.
 - En el formato hexadecimal, añadiríamos dos dígitos hexadecimales al código RGB, de modo que 00 sería transparente y FF (255) sería opaco. Ejemplo: #ff000080 sería el rojo a un 50% de transparencia.

Función rgb

Permite calcular un color en función de la cantidad de sus componentes.

Tiene cuatro parámetros:

- Componente R. Cantidad de rojo en el color.
- Componente G. Cantidad de verde en el color.
- Componente B. Cantidad de azul en el color.
- Canal alfa: nivel de transparencia. Es opcional. Si no aparece el color será opaco.

Puede recibir:

- Un valor de 0 a 255 o un porcentaje de 0% a 100% para los colores.
- Un valor de 0 a 1 o un porcentaje de 0% a 100% para el canal alfa.

Función rgb – Ejemplos

Algunos ejemplos de colores con la función rgb:

```
color: rgb(255 0 0);  
color: rgb(100% 0% 0%);  
background-color: rgb(230 150 45 / 20%);  
background-color: rgb(90% 59% 18% / 20%);  
background-color: rgb(90% 59% 18% 20%); /* barra opcional */
```

Esta es la notación moderna. Es muy habitual ver la notación clásica con comas:

```
color: rgb(255, 0, 0);  
color: rgb(100%, 0%, 0%);  
background-color: rgb(230, 150, 45, 20%);  
background-color: rgb(90%, 59%, 18%, 20%);
```

En la notación clásica no se puede usar la barra (/).

No todas las notaciones están soportadas en todos los navegadores.

Función rgba

La función rgba es una versión antigua de rgb, que se usaba para poder especificar un canal alfa.

Desde la actualización de la función RGB para que use canales alfa, no deberíamos usar rgba, pero la seguiremos encontrando en mucho código "legacy".

Colores especiales

Hay dos valores especiales para los colores:

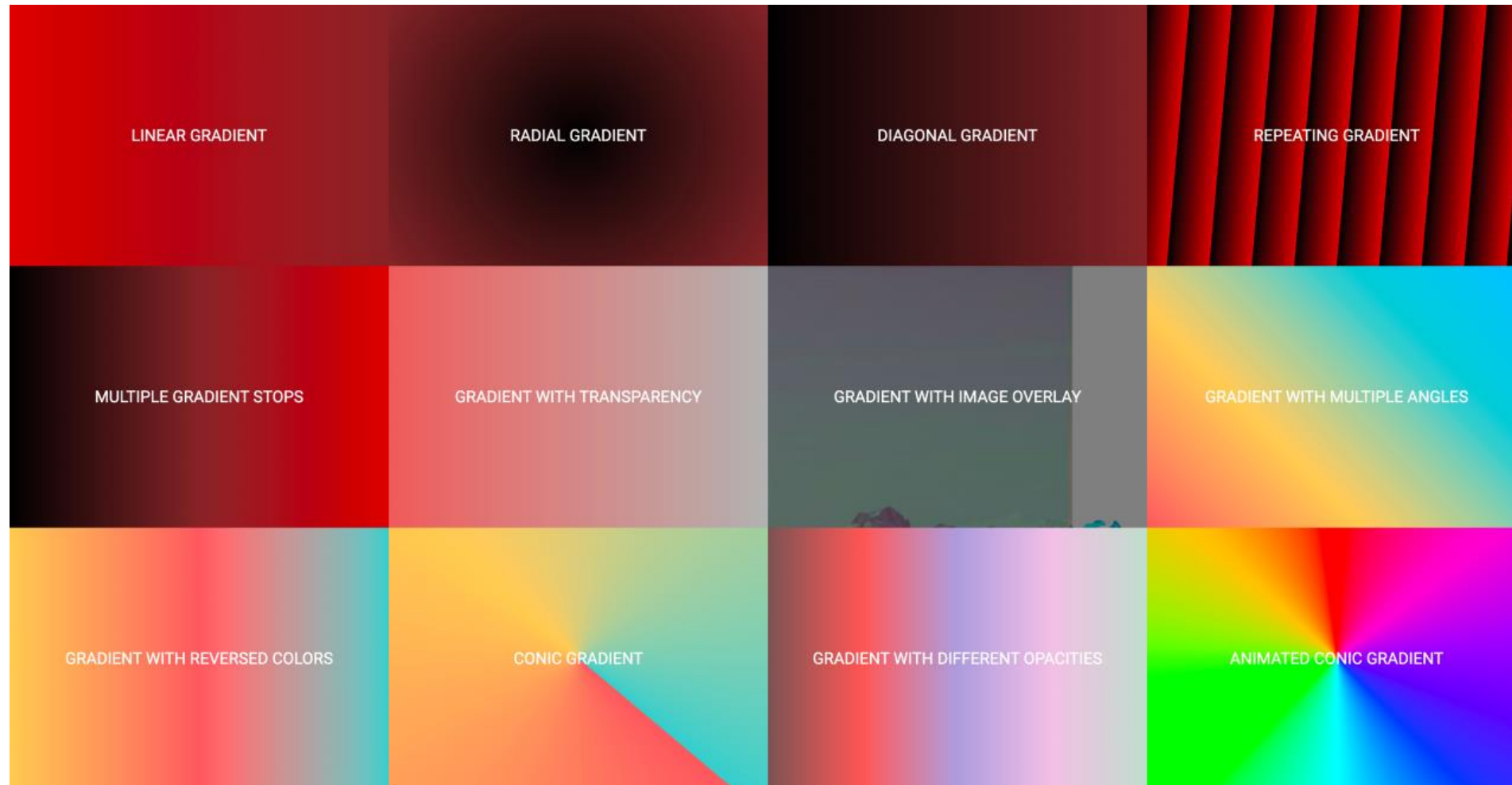
- transparent: transparente. Es el equivalente a una transparencia total, o canal alfa cero. Es el valor por defecto para los fondos (background-color)
- currentColor: sirve para referirse al color actual del texto. Sería la propiedad color del elemento o de un elemento ancestro.

También tenemos "constantes" para colores del sistema, como:

- linktext: color de enlaces no visitados
- field: color de fondo de un campo de texto <input>
- highlight: color de fondo de textos seleccionados
- ...

Gradientes (degradados)

Permiten crear colores no uniformes, que varían de un color a otro.



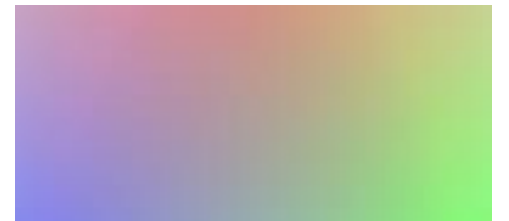
Fuente de la imagen:

<https://www.linkedin.com/pulse/12-examples-css-gradients-elevate-your-web-design-abhishek-garg-/>

Gradientes

Los gradientes se definen con funciones.

- Lineales: varían de un color a otro, creando bandas que progresan a lo largo de una dirección, siguiendo una línea recta. Se usa la función "linear-gradient".
- Radiales: El color inicial está en el centro, y progresa hacia el exterior, siguiendo los radios imaginarios de una circunferencia o elipse. Se usa "radial-gradient".
- Cónicos: la transición se produce rodeando un punto central. Se usa "conic-gradient". El ejemplo combina varios radiales con distintos alfa.



De cada uno de estos tipos hay variantes para poder repetir el degradado, y hay algunos otros más avanzados. Sólo veremos el lineal.

Gradientes lineales

Importante: no funcionan con la propiedad "background-color". Se debe usar "**background-image**" o el atajo "**background**".

Se definen con la función linear-gradient.

Esta función está sobrecargada, es decir, tiene varias versiones que permiten personalizar la forma en la que se hace el degradado.

En su forma más básica admite dos colores, y la transición se hace del primero al segundo, y de arriba hacia abajo:

```
background: linear-gradient(red, yellow);
```

Pero se pueden especificar más de dos colores, cuánto ocupa cada color, degradar a transparencia, superponer colores con canal alfa, hacer patrones repetitivos, etc. Ver los ejemplos en el repositorio en GitHub (UT03).

Lenguajes de marcas y sistemas de gestión de la información



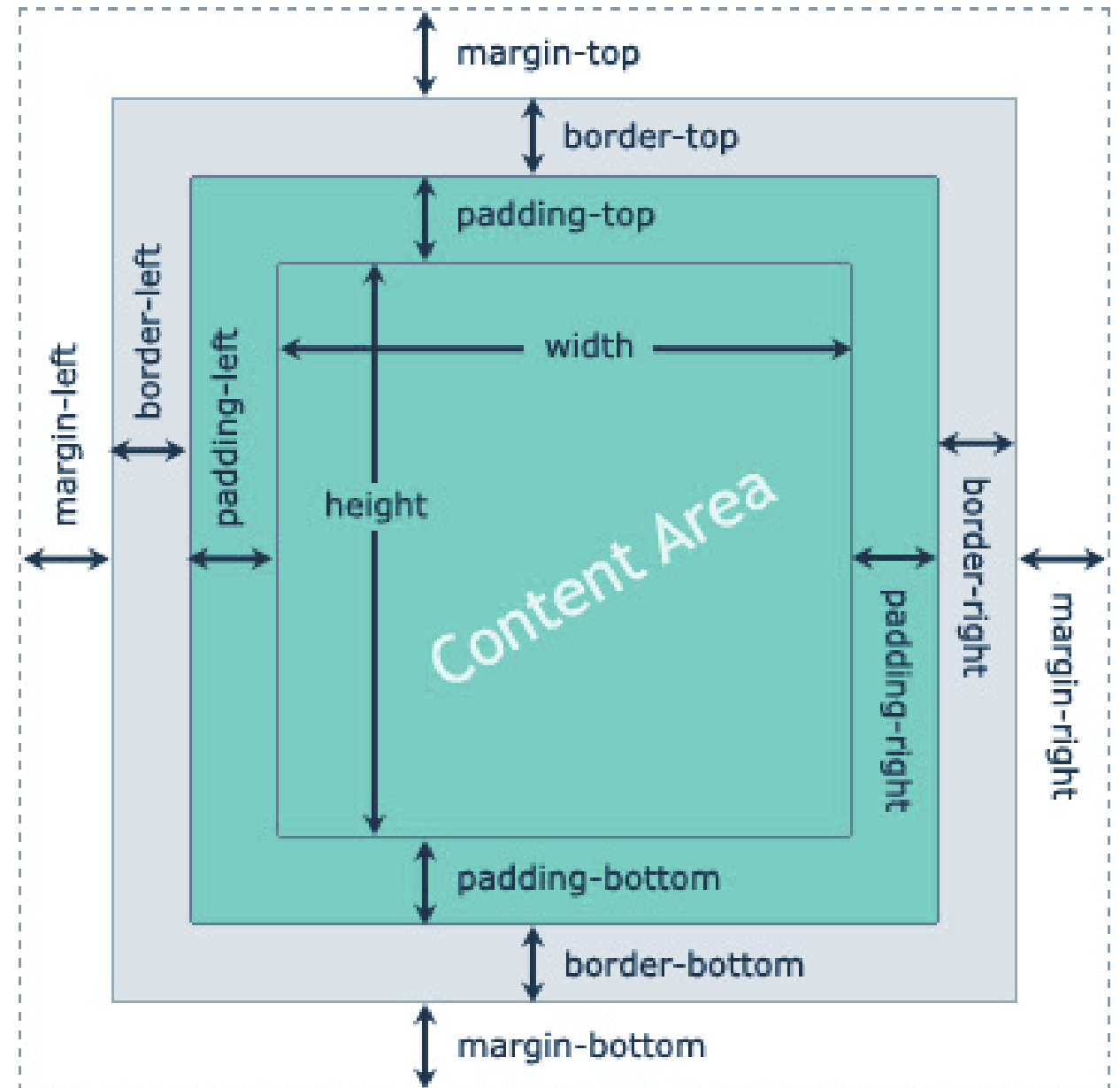
UT03 – CSS

5 – Modelo de caja - Valores y unidades

Modelo de caja en CSS – Partes de la caja

El modelo de caja interpreta los elementos como una caja, un rectángulo, formado por diferentes partes, que desde el interior al exterior de la caja son:

- Contenido (content).
- Relleno (padding).
- Borde (border).
- Margin (margen).



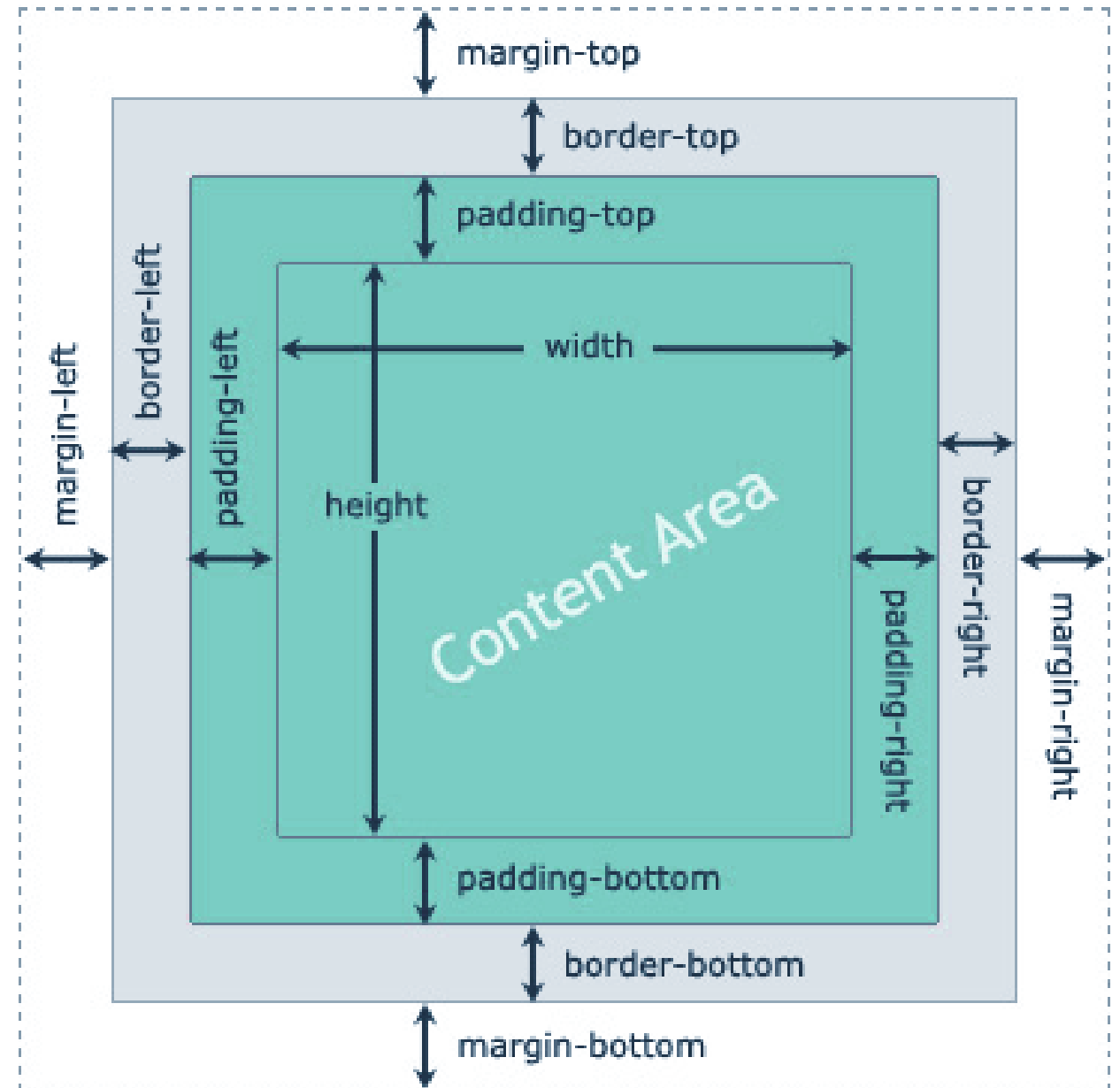
Modelo de caja en CSS – Partes de la caja

Contenido

- Definido por el contenido o los descendientes del elemento. Puede ser simplemente un texto o pueden ser múltiples elementos.

Padding

- Espacio entre el contenido y el borde del elemento.
- Transparente. Si se ha definido un fondo para el elemento, este fondo se verá bajo el padding.



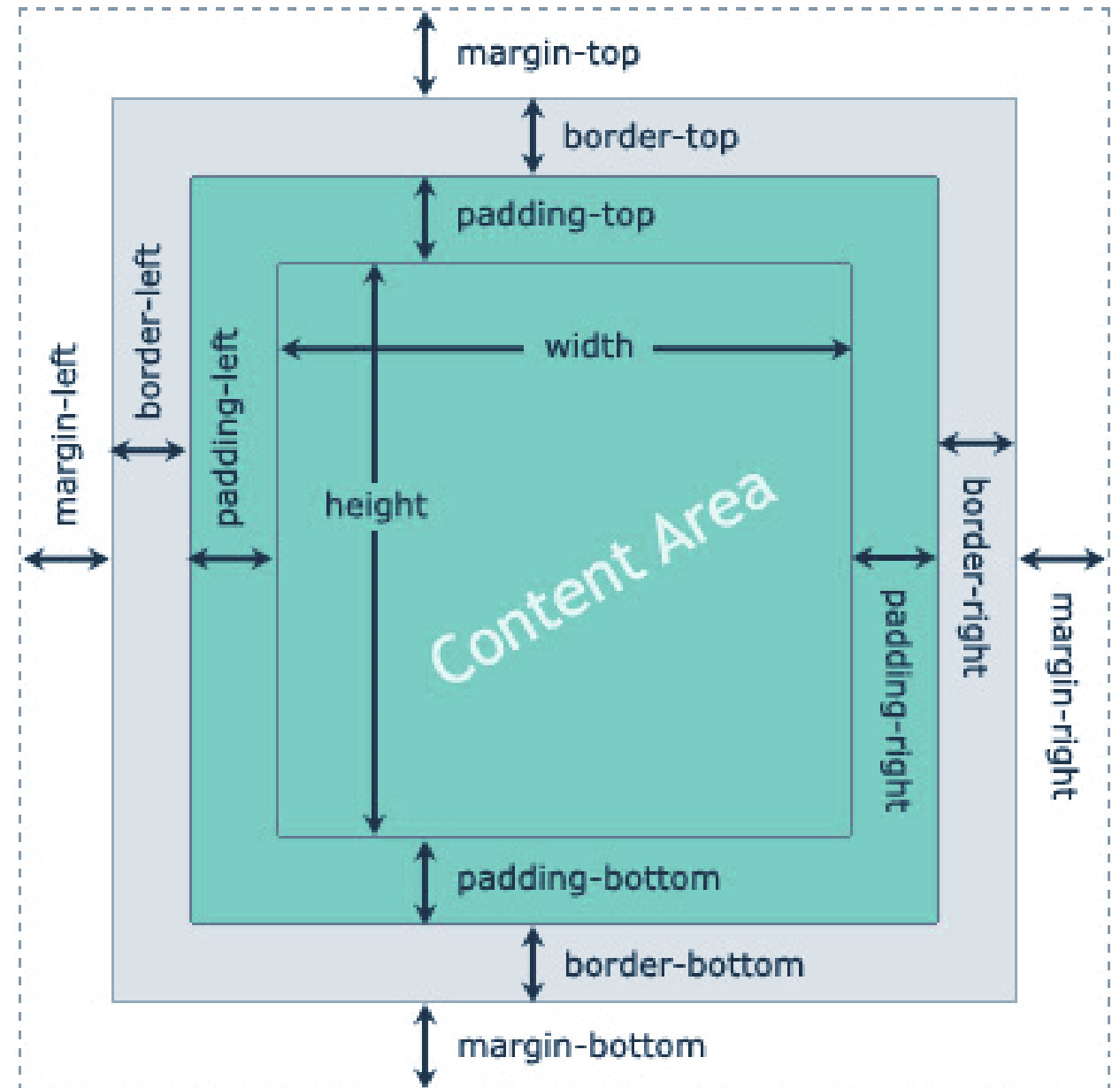
Modelo de caja en CSS – Partes de la caja

Border:

- Espacio entre el límite exterior del padding y el margen del elemento.

Margin:

- Separa el elemento los elementos adyacentes.
- Es exterior y transparente.
- El fondo del elemento, si lo tiene, no se ve bajo el margen.



Propiedades del modelo de caja

Para el contenido:

- width y height: ancho y alto. No se aplican a los elementos en línea. Es una dimensión, un tamaño.
- min-width y min-height: ancho y alto mínimo. Evitan que las propiedades width y height sean menores que cierto valor.
- max-width y max-height: ancho y alto máximo. Evitan que las propiedades width y height sean mayores que cierto valor.

Propiedades del modelo de caja

Para el padding:

- padding-top, padding-right, padding-bottom, padding-left. También una dimensión, un tamaño.
- Atajo: padding. El atajo puede ser:
 - 4 valores: { padding arriba derecha abajo izquierda }
 - 1 valor: { padding todos } El mismo para los cuatro lados.
 - 2 valores: { padding arriba-abajo derecha-izquierda }
 - 3 valores: { padding arriba izquierda-derecha abajo }

Propiedades del modelo de caja

Para el borde:

- Ancho del borde:
 - `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`. Es una dimensión, un tamaño.
 - Atajo `border-width`. Funciona con el mismo sistema de uno, dos, tres o cuatro valores.
- Estilo del borde:
 - `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`. Pueden ser `none`, `solid`, `dotted`, `dashed`, etc.
 - Atajo `border-style`. Funciona con el mismo sistema de uno, dos, tres o cuatro valores.

Propiedades del modelo de caja

Para el borde:

- Color del borde:
 - border-top-color, border-right-color, border-bottom-color, border-left-color. Es un color (nombre, hexadecimal, función...)
 - Atajo border-color. Funciona con el mismo sistema de uno, dos, tres o cuatro valores.
- Atajo border.
 - Es un atajo para border-width, border-style y border-color cuando queremos aplicar el mismo valor a todos los bordes del elemento.
 - El orden de los valores no importa. Como son dimensiones diferentes (color, estilo y dimensiones) se identifica qué estamos indicando, aunque no estén ordenadas.

Propiedades del modelo de caja

Para el margin:

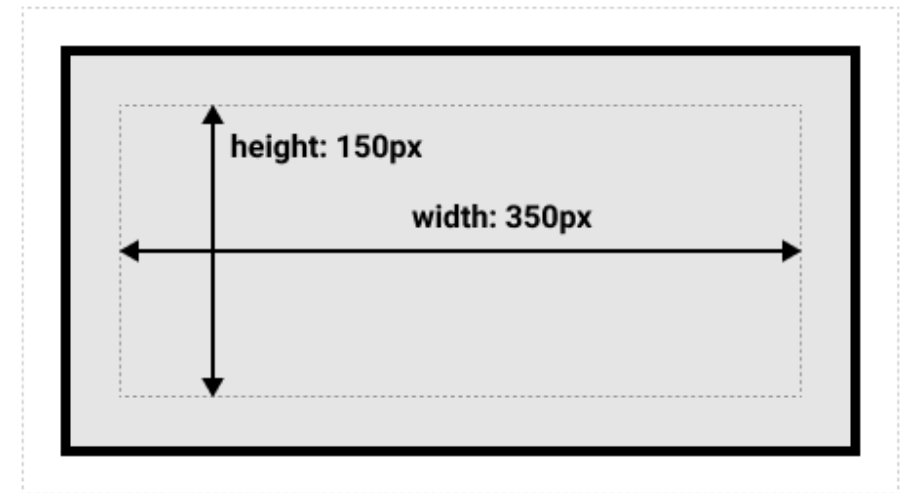
- margin-top, margin-right, margin-bottom, margin-left
- Atajo: margin. El atajo puede ser:
 - 4 valores: {margin arriba derecha abajo izquierda }
 - 1 valor: {margin todos } El mismo para los cuatro lados.
 - 2 valores: {margin arriba-abajo derecha-izquierda}
 - 3 valores: {margin arriba izquierda-derecha abajo }

Modelo de caja clásico

En el modelo de caja clásico, cuando damos ancho o alto (propiedades width o height) estamos dando el tamaño al contenido.

El elemento ocupará, además del ancho y alto indicado, el espacio indicado para padding, border y margin.

```
.caja-clasica {  
  width: 350px;  
  height: 150px;  
  margin: 10px;  
  padding: 25px;  
  border: 5px solid black;  
}
```



Alto total = $150 + (25 * 2) + (5 * 2)$, y se separará 10 de su entorno.

Ancho total = $350 + (25 * 2) + (5 * 2)$, y se separará 10 de su entorno.

Modelo de caja alternativo

El modelo de caja clásico puede ser problemático en algunos escenarios, porque es complicado saber cuánto ocupará realmente un elemento.

Para estas situaciones se puede activar el modelo de caja alternativo, con la propiedad `box-sizing`. Por defecto es `content-box` (clásico), pero se puede cambiar a alternativo con `border-box`.

En este modelo, el ancho y alto (`width` y `height`) se aplican a toda la caja, sin incluir el margen, que no se considera parte de la caja.

En este caso el ancho o alto del contenido se calcula restando de la caja las dimensiones de borde y `padding`.

Modelo de caja alternativo

Ejemplo:

```
.caja-alternativa {  
    box-sizing: border-box;  
    width: 350px;  
    height: 150px;  
    margin: 10px;  
    padding: 25px;  
    border: 5px solid black;  
}
```



Alto del contenido = $150 - (25 * 2) - (5 * 2)$, y se separará 10 de su entorno.

Ancho del contenido = $350 - (25 * 2) - (5 * 2)$, y se separará 10 de su entorno.

Caja en elementos en bloque

En los elementos en bloque en CSS:

- Su propiedad display es "block".
- La caja fuerza un salto de línea al final.
- La caja ocupa todo el ancho disponible, llegando al 100% si es posible.
- Se respetan las propiedades width y height
- El relleno, margen y borde mantienen alejados a otros elementos.

Ejemplos de estos elementos son <p>, <h1>, <div>, , ...

Cualquier elemento no en bloque se puede convertir a elemento en bloque cambiando su display a "block".

Caja en elementos en línea

En los elementos en línea en CSS:

- Su propiedad display es "inline".
- La caja no fuerza ningún salto de línea. Los elementos en línea se suceden en la misma línea uno tras otro.
- La caja ocupa sólo el ancho necesario para su contenido.
- No se aplican las propiedades width y height.
- El relleno, margen y borde se respetan, pero no mantienen alejados a otros elementos. Se produce cierto solapamiento.

Ejemplos de estos elementos son <a>, , , , ...

Cualquier elemento no en línea se puede convertir a elemento en línea cambiando su display a "inline".

A medio camino entre bloque y en línea

Hay una forma de hacer una mezcla entre en bloque y en línea.

- Su propiedad display es "inline-block".
- No fuerza salto de línea. Como un elemento en línea.
- Ocupa sólo el ancho necesario para su contenido, salvo si se indica width y/o height. En este caso se respeta, como en bloque.
- El relleno, margen y borde mantienen alejados a otros elementos, como en bloque

No hay elementos con este comportamiento. Hay que forzarlo.

Útil para dar márgenes a elementos en línea en el texto.

Para otras aplicaciones (menús en una sola línea, por ejemplo), se ha visto sobrepasado por flex y grid, que veremos más adelante.

Colapso de márgenes

Cuando se trabaja con márgenes verticales (superior e inferior), hay que tener en cuenta que estos márgenes no se acumulan.

Es decir, si tengo dos elementos en bloque (dos p, por ejemplo) seguidos, el primero tiene un margen inferior de 20px y el segundo un margen superior de 50px, la separación no será de 70px.

En estos casos el margen aplicado será el mayor de los dos, así que serán sólo 50px.

Este comportamiento se denomina colapso de márgenes, y se puede producir en otras circunstancias, como que se colapse el margen de un elemento y el de su contenedor.

Más información en [Entendiendo el colapso de márgenes – MDN](#).

Valores CSS

Todas las propiedades CSS tienen una serie de valores admitidos.

En la referencia CSS en MDN hay siempre una sección para cada propiedad en la que se muestran los valores admitidos.

Por ejemplo, si miramos la propiedad "padding-left" en MDN, encontramos una sección "Values", en la que podemos encontrar que esta propiedad admite "<length>" y "<percentage>".

Estos valores que aparecen entre símbolos <> indican que puede tomar un valor más o menos arbitrario, pero siguiendo normas. Una longitud (length) puede expresarse de varias formas (px, pt, em, etc.). Un porcentaje es un número seguido del símbolo %.

Otro ejemplo: si miramos text-align vemos que admite sólo valores concretos: start, end, left, right, center, justify, justify-all y match-parent.

Valores CSS – Dimensiones

Hay ciertos valores CSS que identifican dimensiones. Estas dimensiones pueden ser (entre otras):

- Longitud. Por ejemplo, el alto o ancho de un elemento, el relleno (padding) o el margen, se establecen con longitud.
- Ángulos. En CSS los ángulos se miden en sentido de las agujas del reloj (a diferencia del estándar matemático, que mide en sentido contrario a las agujas del reloj)
- Tiempo. Usado en transiciones y animaciones.

Hay otros valores CSS que se refieren a color o posición, pero vamos a centrarnos de momento en dimensiones, porque es la mejor forma de entender el sistema de unidades de CSS.

Valores CSS – Valores universales

Hay ciertos valores que se pueden aplicar a cualquier propiedad CSS:

- **initial:** devuelve la propiedad a su valor por defecto.
- **inherit:** calcula el valor de la propiedad en función del elemento padre. No hace nada si la propiedad no es heredable.
- **unset:** actúa como initial o inherit, dependiendo de la propiedad:
 - Si es heredable, será inherit
 - Si no lo es, será initial
- **revert:** similar a unset, pero si no es heredable volverá al valor establecido por los estilos de usuario o del navegador, si los hay.
- **revert-layer:** elimina un nivel de la cascada.

Los más utilizados son initial e inherit.

Dimensiones en CSS y sus unidades

Una dimensión es un número con unas unidades asociadas. Por ejemplo: 30deg, 100ms, 50px, 2em, etc.

Las unidades son case-insensitive, podemos usar tanto 20px como 20PX.

Las unidades se escriben sin espacio entre el número y la unidad.

En CSS se usan dimensiones para especificar:

- Longitudes o distancias
- Ángulos
- Tiempo y frecuencia
- Resolución
- Flex. Similar a un porcentaje, pero es una fracción de cierta parte del elemento. Lo veremos cuando veamos flex y grid.

Unidades CSS de distancia

Las unidades de longitud o distancia (o tamaño) pueden ser:

- Absolutas: definen un valor absoluto, que no depende de otros elementos en el documento.
- Relativas: definen un valor relativo al tamaño de otra cosa. Hay diferentes unidades que toman como referencia distintos elementos del documento u otros elementos, como la parte visible de la página.
- Relativas al contenedor de una media query. Utilizadas junto a las media queries para hacer diseños responsive.

Unidades absolutas

Son unidades con un tamaño fijo, relativo al cm o a la pulgada.

- Píxeles (px): 1 píxel de la pantalla. $1\text{px} = 1/96$ pulgadas.
- Puntos (pt): $1\text{pt} = 1/72$ pulgadas.
- Picas (pc): $1\text{pc} = 1/6$ pulgadas.
- Pulgadas (in): $1\text{in} = 2.54\text{ cm} = 96\text{px}$.
- Cuarto de milímetro (q): $1\text{q} = 0.25\text{ mm} = 1/40\text{ cm}$
- Milímetro (mm): $1\text{mm} = 1/10\text{ cm}$
- Centímetro (cm): $1\text{cm} = 10\text{mm} = 96\text{px}/2.54$

Son útiles sobre todo para definir distancias en medios de tamaño fijo, como papel. Un A4 es siempre de las mismas dimensiones, pero una pantalla de móvil, tablet o PC puede tener tamaños variados.

Unidades relativas

Son unidades con un tamaño relativo a otros elementos del documento.
Pueden ser:

- Relativas al tamaño de fuente.
- Relativas a otra distancia.
- Relativas al viewport. El viewport es la porción de documento que se está visualizando.

Unidades relativas a la fuente

Toman como referencia la fuente de un elemento del documento.

- em. Relativa al tamaño de la fuente (font-size) del elemento.
 - 1em es el 100% del tamaño de fuente.
 - 0.5em es el 50% y 2em es el doble del tamaño de fuente.
 - Muy usado en line-height, font-size, margin-top y margin-bottom.
- rem. Igual que em, pero relativa al tamaño de fuente de la raíz del documento. Es la mejor para crear interfaces escalables.
- ex. Relativa a la altura de la x minúscula de la fuente del elemento. En fuentes que tienen la x minúscula, es muy habitual que $1\text{ex} = 0.5\text{em}$.
- cap. Relativa al alto de las mayúsculas de la fuente del elemento.

Unidades relativas al viewport

El viewport es la región del documento que está siendo visualizada en ese instante.

Es la parte del documento visible en la pantalla, es decir, se excluyen barras de herramientas del navegador, o el resto de la pantalla si el navegador no está maximizado.

Si el documento se está visualizando en pantalla completa, modo en que no se ven las herramientas del navegador ni otras aplicaciones, coincidirá con el tamaño de la pantalla.

En HTML 5 hay una directiva viewport que podemos poner en un tag meta en la cabecera para dar indicaciones al documento de cómo se tiene que visualizar. Suele funcionar sólo en dispositivos móviles. Los navegadores de escritorio.

Unidades relativas al viewport

- vw. Relativa al ancho del viewport. $1\text{vw} = 1\%$ del ancho del viewport.
- vh. Relativa al alto del viewport. $1\text{vh} = 1\%$ del alto del viewport.
- vi. Relativa al eje "inline" del elemento raíz. El eje inline es el eje en sentido de escritura. Esta unidad es una unidad lógica que usa para dar soporte a escrituras verticales y horizontales. En nuestro estándar "left-to-right" vi es lo mismo que vw.
- vb. Relativa al eje "block" del elemento raíz. El eje block es el perpendicular al sentido de escritura. En nuestro estándar "left-to-right" vi es lo mismo que vh.
- vmin y vmax. Relativas a la menor dimensión (alto/ancho) del viewport. Igual que las otras, 1vmin será el 1% del menor de los dos valores.

Unidades relativas a la altura de línea

Toman como referencia la altura de línea (line-height) de un elemento del documento.

- lh. Relativa a la altura de línea del elemento.
- rlh. Relativa a la altura de línea del elemento raíz del documento.

Otras - Ángulos y tiempo / frecuencia

Ángulos:

- deg. Grados sexagesimales. Hay 360 grados en un círculo completo.
- grad. Gradianes. Hay 400 gradianes en un círculo completo.
- rad. Radianes. Hay 2π radianes en un círculo completo.
- turn. Vueltas o revoluciones. Hay una vuelta en un círculo completo

Tiempo

- s y ms. Segundo y milisegundo. $1s = 1000ms$.

Frecuencia

- hz y khz. Hercio y Kilohercio. $1Hz$ 1 vez / segundo. $1khz = 1000hz$.

Valores CSS – Color – Image

Un valor de color (<color>) puede ser:

- Palabras clave de colores: red, green, transparent, etc.
- Valores hexadecimales completos o abreviados, con o sin canal alfa.
- Valores de color (con o sin canal alfa) a partir de funciones, como rgb(), rgba(), hls(), hlsa(), etc.

Un valor de imagen (<image>) puede ser:

- Un archivo de imagen al que se accede a través de la función url()
 - URL absoluta: url(https://sitioweb.com/imagen.jpg).
 - URL relativa a la ubicación del CSS: url(imagen.png)
- Un gradiente (degradado)

Valores CSS – Otros

Otros valores CSS que iremos viendo más adelante son:

- Posición (<position>): top, left, right, bottom, center. También puede ser una combinación de varios o combinarse con porcentajes u otras dimensiones.
- Cadenas de texto (<string>): usadas en ocasiones para indicar contenido antes o después de los elementos.

Lenguajes de marcas y sistemas de gestión de la información



UT03 – CSS

6 – Pseudoclases y pseudoelementos

Pseudoclases

Es un tipo de selector que identifica los elementos que están en un estado específico. Podemos dividirlos en:

- De interacción. Relacionadas con las acciones del usuario.
- De ubicación. Relacionadas con enlaces.
- De estructura
- De formularios. Relacionadas con uso y validación de formularios
- De idioma. Relacionadas con el idioma establecido en los elementos.
- De estado. Relacionadas con el estado de la ventana o de elementos modales.

Sólo veremos algunas pseudoclases. Referencia completa en MDN.

Pseudoclasses – Sintaxis

Las pseudoclasses comienzan con dos puntos (:)

Ejemplo de pseudoclasses:

- :hover
- :any-link
- :root

La mayoría pueden añadirse a un selector:

Ejemplos:

- a:hover
- p.clase-de-parrafo span:first-of-type

Pseudoclases – De interacción

Se pueden usar en cualquier elemento, pero es muy habitual usarlas en aquellos con los que interactúa el usuario.

- `:hover` – El usuario pasa el ratón sobre dicho elemento.
- `:active` – El usuario se encuentra pulsando dicho elemento.
- `:focus` – El elemento cuando tiene el foco.
- `:focus-within` – Uno de los hijos del elemento tiene el foco.
- `:focus-visible` – El elemento tiene el foco (se cumple `:focus`) y el elemento debe destacarse (esto lo determina el navegador)

Pseudoclases – De ubicación (enlaces)

Permiten cambiar el estado de enlaces u otros hipervínculos (area).

- :any-link – Cualquier elemento que es un enlace (incluye area).
- :link – El enlace no se ha visitado aún.
- :visited – El enlace se ha visitado anteriormente.
- :target – Selecciona un enlace cuya ancla ("id") coincide con el ancla de la página actual (la parte detrás de # en la URL).

Pseudoclases – De estructura

Permiten seleccionar elementos en función de su posición o estructura.

- `:root` – Aplica estilo al elemento raíz (padre) del documento. Similar a `html`, pero más específica que `html`.
- `:first-child` – Primer elemento hijo (de cualquier tipo).
- `:last-child` – Último elemento hijo (de cualquier tipo).
- `:nth-child(n)` – Elemento hijo número n (de cualquier tipo).
- `:nth-last-child(n)` – Elemento hijo número n empezando desde el final (de cualquier tipo).

Pseudoclases – De estructura

Permiten seleccionar elementos en función de su posición o estructura.

- :first-of-type – Primer elemento hijo (cierto tipo de elemento).
- :last-of-type – Último elemento hijo (cierto tipo de elemento).
- :nth-of-type(n) – Elemento hijo número n (cierto tipo de elemento).
- :nth-last-child(n) – Elemento hijo número n empezando desde el final (cierto tipo de elemento).
- :only-child – Elemento que es hijo único (de cualquier tipo).
- :only-of-type – Elemento que es hijo único (cierto tipo de elemento).
- :empty Elemento vacío (sin hijos, ni texto).

Pseudoclases – De formularios (estado)

Permiten trabajar con los estilos de campos de formularios, en función de su estado.

- `:disabled` – El campo está deshabilitado.
- `:enabled` – El campo no está deshabilitado.
- `:read-only` – El campo es de solo lectura.
- `:read-write` – El campo es editable, no es read-only.
- `:placeholder-shown` – Se está mostrando el atributo placeholder.
- `:default` – El valor del campo tiene el valor por defecto.
- `:checked` – El campo está marcado. Para radio y checkbox.

Pseudoclases – De formularios (validación)

- :required – El campo es obligatorio (tiene atributo required).
- :optional – El campo es opcional – Por defecto son opcionales todos.
- :valid – El campo es válido, ha pasado la validación.
- :invalid – El campo no es válido, no ha pasado la validación.
- :user-valid – Igual que valid, pero requiere que el usuario haya interactuado con el campo antes de hacer la validación.
- :user-invalid – Igual que user-invalid, pero para campos no válidos.
- :in-range – El valor del campo está entre los valores indicados en los atributos min y max.
- :out-of-range – El valor del campo no está entre min y max.

Pseudoelementos

Permiten aplicar estilos a elementos "virtuales".

Los pseudoelementos no existen como tales en el documento HTML, no están marcados con etiquetas, sino que se "crean" dinámicamente.

Hay dos tipos:

- Pseudoelementos que identifica partes ya existentes del contenido, como primera línea de un texto, primera letra, errores ortográficos, etc.
- Pseudoelementos que permiten añadir contenido antes o después de otro elemento, sin necesidad de usar JavaScript.

Los pseudoelementos en CSS 3 comienzan con "::" para distinguirlos de las pseudoclases, que empiezan con ":". En CSS 2 empezaban igual, y esa sintaxis todavía se admite.

Pseudoelementos

Algunos pseudoelementos son todavía experimentales, y su comportamiento puede cambiar en el futuro.

- `::first-letter` – Selecciona la primera letra del texto.
- `::first-line` – Selecciona la primera línea del texto.
- `::selection` – Aplica los estilos al fragmento de texto seleccionado por el usuario.
- `::target-text` – Aplica estilos al fragmento de texto enlazado tras el ancla de una URL (tras el símbolo #)
- `::spelling-error` y `::grammar-error` – Aplica estilos a un fragmento resaltado por errores ortográficos / gramaticales

Pseudoelementos

- `::marker` – Aplica estilos al "bullet" (ul) o número (ol) de cada elemento de una lista.
- `::backdrop` – Aplica estilo al fondo que rodea a un elemento, sin que afecte al propio elemento.
- `::placeholder` – Aplica estilos al placeholder de un campo input, cuando este es visible (el campo no tiene ningún valor)
- `::file-selector` – Aplica estilos al botón que acompaña a un campo de tipo file (`<input type="file">`). Por motivos relacionados con la seguridad, los navegadores / sistemas operativos generan este botón con estilos muy "rústicos". Este pseudoelemento ayuda a dar estilos a un componente que antes era difícil de maquetar.

Pseudoelementos

Generación de contenido

Hay dos pseudoelementos que permiten añadir contenido al documento desde CSS:

- `::before` – Añade contenido antes del selector asociado al pseudoelemento.
- `::after` – Igual, pero añade el contenido después.

Ejemplos:

```
p::before { ... }  
a::after { ... }
```

El primero añadiría contenido antes de todos los párrafos.

El segundo añadiría contenido después de cada elemento a.

Pseudoelementos

Generación de contenido

`::before` y `::after` funcionan siempre junto a la propiedad "content".

`content` establece o reemplaza el contenido seleccionado por un selector. En el caso de `::before` y `::after`, del pseudoelemento generado.

La propiedad `content` puede recibir:

- `none` – No genera contenido.
- `normal` – Volver a comportamiento por defecto. En el caso de `::before` y `::after`, el comportamiento es como si usáramos `none`.
- Un texto – Crea el pseudoelemento con el texto indicado.
- `open-quote` y `close-quote`. Genera comillas de cita ("«" o "»").
- `no-open-quote` y `no-close-quote`. Elimina las comillas de cita.

Pseudoelementos

Generación de contenido

La propiedad content puede recibir (continuación):

- attr(nombre-del-atributo). Extrae el valor del atributo indicado y lo usa como contenido. Ejemplo:
`a::after { content: attr(title); }`
- Un degradado (linear-gradient, radial-gradient, etc.).
- url(url-de-elemento-multimedia). Inserta un elemento multimedia en el pseudoelemento. Normalmente se utiliza con imágenes, pero puede usarse con otros medios.
- counter(nombre de contador) – Inserta un contador que se va incrementando en cada ocasión que se cumple el selector. Los contadores CSS se definen/ personalizan con counter-reset, counter-increment y counter-set.

Lenguajes de marcas y sistemas de gestión de la información



UT03 – CSS

7 – Listas y tablas

Listas – Estilos específicos

Las listas `` y `` disponen de propiedades específicas:

- `list-style-type` – Permite cambiar el tipo de "bullet/símbolo" de los ``.
 - Puede tener los valores:
 - `none` – No habrá símbolo.
 - Símbolos habituales (`ul`): `disc`, `circle`, `square`, `disclosure-open` (flecha hacia abajo), `disclosure-closed` (hacia derecha)
 - Numeraciones habituales (`ol`): `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, y otras muchas para otros idiomas
 - Un icono en UTF8
 - Una lista de símbolos, con la función `symbols()`

Listas – Estilos específicos

Las listas `` y `` disponen de propiedades específicas:

- `list-style-position` – Permite cambiar el modo de indentación para los elementos de la lista cuando estos ocupan más de una línea.
 - Puede tener los valores:
 - `outside` – Valor por defecto. El texto de la segunda y siguientes líneas se alinea con el texto de la primera línea.
 - `inside` – El texto se alinea con el "bullet" de la lista

Importante: esta propiedad hay que ponerla en los elementos "li", no funciona en ul / ol

Listas – Estilos específicos

Las listas `` y `` disponen de propiedades específicas:

- `list-style-image` – Permite sustituir el "bullet" por una imagen.
 - Puede tener los valores:
 - `none` – No habrá imagen. Valor por defecto. Útil para sobrescribir una regla previa.
 - `url(image.png)` – Indica la URL de la imagen que se usará. Si es relativa, es relativa a la ubicación del CSS.
 - Se puede usar en:
 - El elemento `ol` / `ul`: aplicará a todos los elementos de la lista
 - Un elemento `li` específico. Sólo aplicará a este elemento

Listas – Estilos específicos

Existe el atajo list-style, en el que agrupamos las tres propiedades.

El orden recomendado en este atajo es:

- type
- position
- image

Si no hace falta alguno de ellos, podemos omitirlo. Por ejemplo, si ponemos image no será necesario poner el tipo.

Tablas – Estructura de tabla

Como recordatorio, la estructura recomendada para tablas HTML es:

```
<table>
  <caption>Título de la tabla</caption>
  <thead>
    <!-- Filas de encabezado -->
  </thead>
  <tbody>
    <!-- Filas de datos -->
  </tbody>
  <tfoot>
    <!-- Filas del pie -->
  </tfoot>
</table>
```

La agrupación en thead, tbody y tfoot es opcional, pero recomendada. El título de tabla también es opcional.

Tablas – Estructura de tabla

Cada fila está formada por un elemento tr, que a su vez contendrá una o varias celdas.

Las celdas pueden ser de dos tipos:

- Celdas de encabezado. Con el elemento th. Es obligatorio indicar a qué se aplica el encabezado, con el atributo scope. Puede ser: "col" o "row".
- Celdas de datos. Con el elemento td

Tablas – Estilos específicos

A las tablas, filas o celdas podemos usar propiedades generales, como, por ejemplo, border, background-color, font-family, padding, margin, etc.

Pero hay algunas propiedades específicas para maquetación de tablas:

- border-collapse. Determina si habrá separación entre las celdas de una tabla. Puede ser:
 - separate: Hay separación. Valor por defecto.
 - collapse: Sin separación.
- border-spacing: Solo si se ha fijado border-collapse en separate, esta propiedad indica el tamaño del espacio entre celdas. Puede especificarse un único valor, o dos. Si se indican dos, el primero es la separación horizontal y el segundo la vertical.

Tablas – Estilos específicos

- `caption-side`: Ubicación del título de tabla. El elemento `caption`, si lo hay, debe ser el primero dentro de la tabla. Visualmente, se coloca sobre la tabla, pero se puede colocar bajo ella con el valor `"bottom"`.
- `empty-cells`. Determina si se mostrarán los bordes de celdas vacías. Sólo tiene efecto cuando `border-collapse` es `"separate"`:
 - `show`: Valor por defecto. Se muestran los bordes.
 - `hide`: Se ocultan los bordes.
- `table-layout`: Determina como se debe adaptar la tabla a cambios de tamaño de la ventana del navegador.
 - `auto`. Valor por defecto. La tabla se adapta al tamaño de la ventana.
 - `fixed`: la tabla tiene un tamaño fijo.

Lenguajes de marcas y sistemas de gestión de la información



UT03 – CSS
8 – Float – Flex

Float

La propiedad "float" permite determinar como un elemento debe "flotar" o "empujarse" a un lado u otro.

Puede tomar los valores:

- none: no flota. Se representa como corresponda en el flujo de página. Es el valor por defecto.
- right: se desplaza a la derecha de su contenedor
- left: se desplaza a la izquierda de su contenedor

El caso de uso más habitual es flotar las imágenes dentro de un párrafo a la izquierda o la derecha.

También permite "apilar" elementos a un lado o al otro.

Float

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus imperdiet, nulla et dictum interdum, nisi lorem
egestas odio, vitae scelerisque enim ligula venenatis dolor.
Maecenas nisl est, ultrices nec congue eget, auctor vitae



. Mauris ante
esent convallis
Nunc sagittis



Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus imperdiet, nulla et dictum interdum, nisi lorem
egestas odio, vitae scelerisque enim ligula venenatis dolor.
Maecenas nisl est, ultrices nec congue eget, auctor vitae
massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante
ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis
urna a lacus interdum ut hendrerit risus congue. Nunc sagittis
dictum nisi, sed ullamcorper ipsum diam

Float Next To Each Other

In this example, the three divs will float next to each other.

Div 1

Div 2

Div 3

Clear

Clear permite "romper" el flotado. Puede tomar los siguientes valores:

- right: dejarán de flotar los elementos que flotaban a la derecha
- left: dejarán de flotar los elementos que flotaban a la izquierda
- right: dejarán de flotar todos los elementos, ya sea que flotaran a la izquierda o a la derecha.

Dejar de flotar no significa que no floten los anteriores al elemento con "clear". Significa que ese elemento "reinicia el flotado". Si no tiene float, se comportará de forma natural, y si lo tiene, volverá a empezar a flotar. En este último caso sería como insertar un salto de línea.

Float / Clear para maquetación

Hace años, antes de que aparecieran mejores sistemas (flex y grid), se usaba float para realizar la maquetación de los diseños.

Esto, hoy día no es correcto. No debe usarse, en general, para maquetación, salvo casos concretos.

Como se ha comentado, el caso de imágenes flotadas junto a párrafos es uno de los más habituales, sobre todo si:

- Se usan imágenes con fondos transparentes.
- Se combina con propiedades para "rodear" la imagen con texto, ocupando las partes transparentes: shape-outside, shape-margin y shape-threshold.

Consultar MDN para ejemplos de uso de shape-outside y otras propiedades relacionadas.

Flex

Flex (y grid) son dos formas de posicionar / organizar elementos dentro de un contenedor.

Antes de flex se usaba float y position para organizar elementos, pero era un sistema rudimentario, con gran cantidad de problemas asociados, y que no respondía del todo bien a la variedad de navegadores y dispositivos.

Flex, también llamado flexbox, está pensado para organizar elementos en una dimensión, esto es, en una fila o en una columna.

Aunque puede responder bien para formatos en "tabla", si queremos hacer eso, es mejor usar grid.

Flex – Utilidad

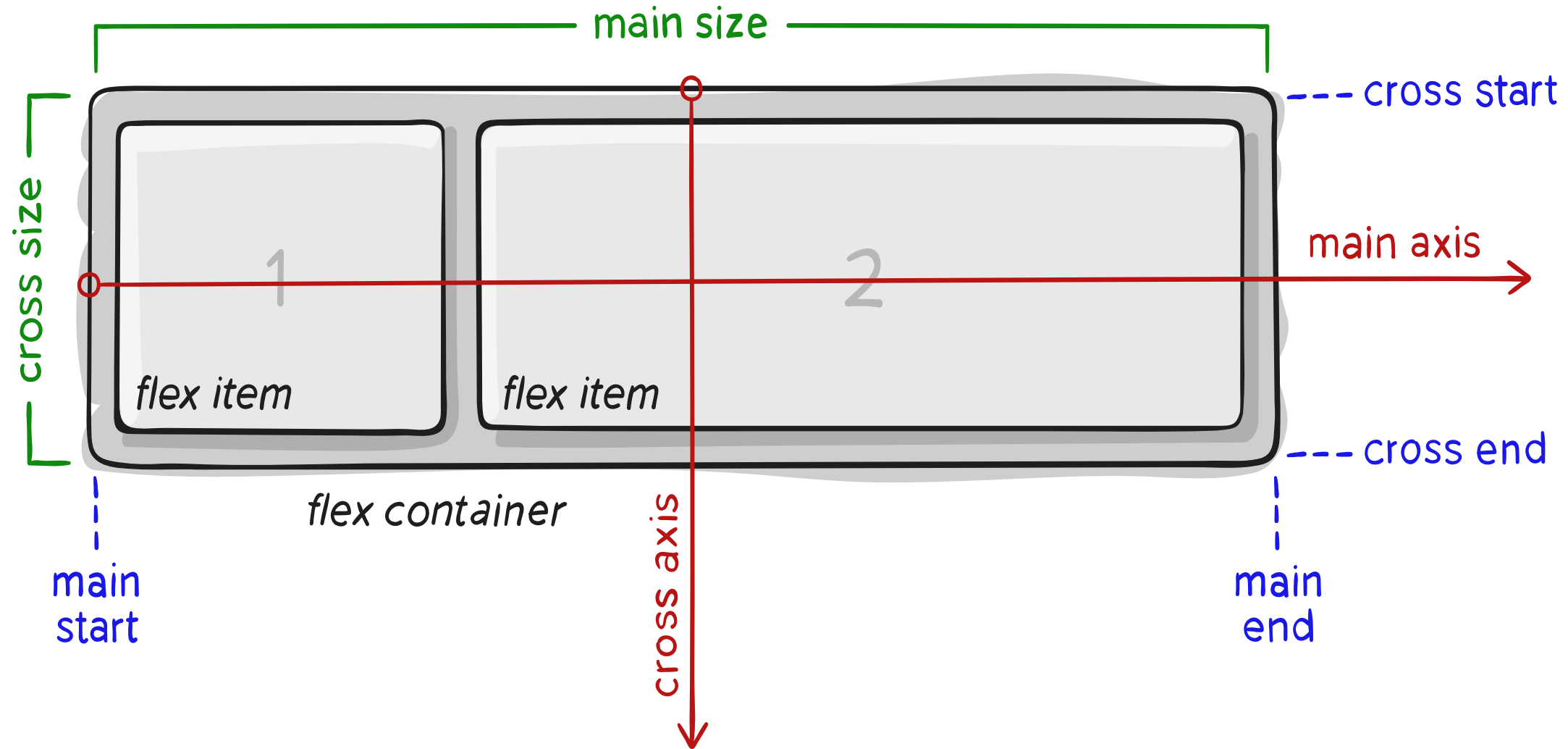
Viene a hacer más fáciles tareas que con float / position son bastantes complicadas, aunque parezcan básicas, como:

- Centrar verticalmente un elemento dentro de su contenedor
- Centrar horizontalmente un elemento dentro de su contenedor
- Hacer que todos los elementos dentro de un contenedor ocupen el mismo espacio
- Hacer que el espaciado entre elementos sea siempre el mismo
- Que varias columnas contiguas tengan el mismo alto independientemente de que tengan más o menos contenido

Flex – Elementos que lo forman

- Flex container: contenedor (elemento externo) donde vamos a utilizar flexbox.
- Flex ítems: elementos secundarios (dentro del flex container) que queremos alinear / posicionar.
- Para utilizar Flexbox hay que aplicar propiedades CSS en el flex container, y casi siempre en los flex items.
- Eje principal y transversal. Flex está diseñado para trabajar en una dimensión, pero esta puede ser horizontal o vertical, y puede trabajar en el sentido de escritura o en el contrario. Los ejes determinan como se distribuyen los elementos.

Flex – Elementos que lo forman



Flex – Elementos que lo forman

- Eje principal (main axis): a lo largo de él se colocan los flex ítems. Usando flex-direction podemos definir la dirección (vertical/horizontal) y sentido (izquierda/derecha/arriba/abajo).
- Main-start y main-end: los flex ítems se colocan en el contenedor empezando por el lado start, dirigiéndose hacia el lado end. Ya no se utiliza left o right, ni top o bottom.
- Tamaño principal (main size): El ancho o alto de un flex container, dependiendo de la dirección del contenedor, es el tamaño principal del ítem. La propiedad de tamaño principal de un flex ítem puede ser tanto width como height, dependiendo de cuál esté en la dirección principal.

Flex – Elementos que lo forman

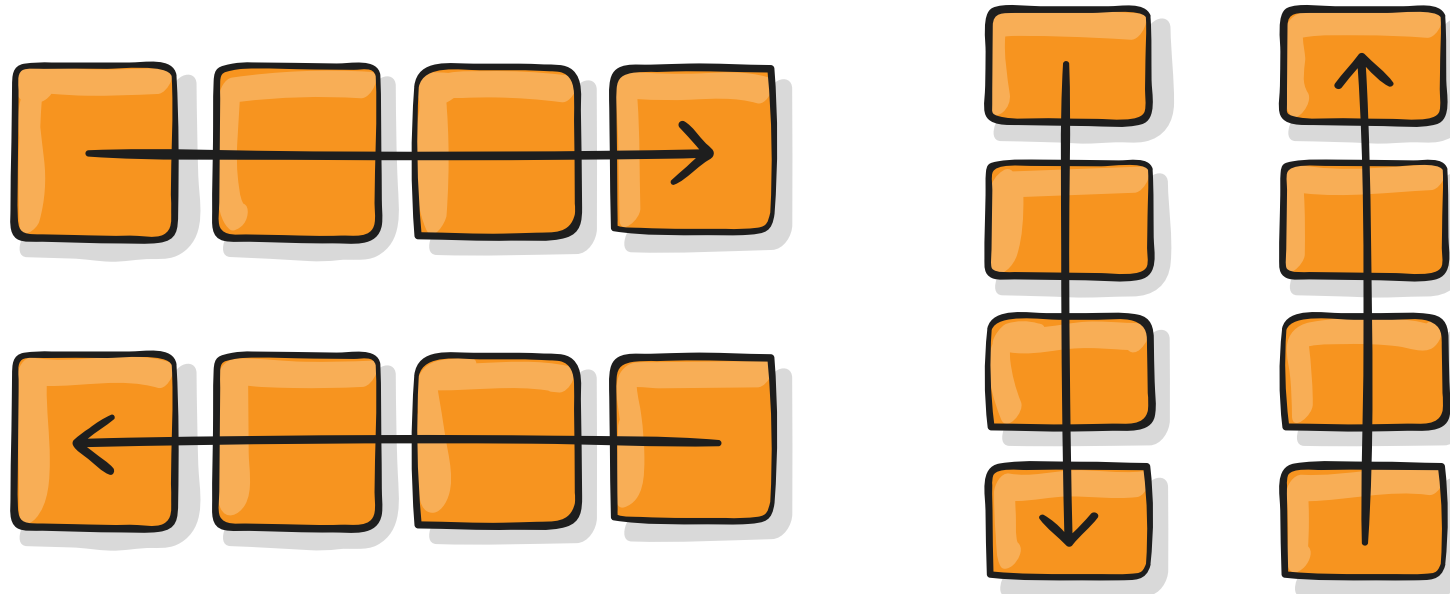
- Eje transversal (cross axis): El eje perpendicular al eje principal se llama eje transversal. Su dirección depende de la dirección del eje principal.
- Cross-start | cross-end: Líneas flex se llenan con ítems y se agregan al contenedor, comenzando desde el lado cross start del flex container hacia el lado cross end.
- Tamaño transversal (cross size): El ancho o alto de un flex ítem, dependiendo de lo que haya en la dimensión transversal, es el cross size del ítem. La propiedad cross size puede ser el ancho o el alto del ítem, lo que se encuentre en la transversal.

Flex – Propiedades del container

- display:
 - flex, inline-flex
 - Ambas opciones hacen que el contenedor sea un flex container.
 - La diferencia está en cómo se comporta el contenedor respecto a lo que le rodea:
 - "flex" hace que el contenedor sea un elemento en bloque, y ocupe todo el ancho disponible, provoque salto de línea, etc.
 - "inline-flex" hace que sea un elemento en línea
- Internamente, dentro del contenedor, los elementos se distribuirán siguiendo la estructura flex.

Flex – Propiedades del container

- flex-direction:
 - row, row-reverse, column, column-reverse
 - establece el eje principal (dirección y sentido)



Flex – Propiedades del container

- flex-wrap:
 - nowrap, wrap, wrap-reverse
 - Por defecto todos los flex ítems se colocan en una sola fila o columna (nowrap).
 - Esto permite que los ítems pasen a la siguiente línea cuando sea necesario.
 - La diferencia entre wrap y wrap-reverse es que el primero llena la primera línea y añade en la dirección del eje transversal, y el segundo añade líneas en sentido contrario al eje transversal, es decir, "antes" de la primera línea.

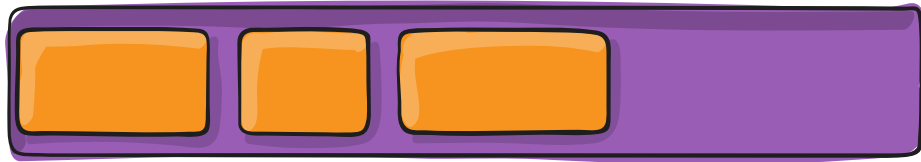
Flex – Propiedades del container

- flex-flow:
 - Es una abreviatura para flex-direction y flex-wrap. Ej: flex-flow: row wrap
- justify-content:
 - flex-start, flex-end, center, space-between, space-around, space-evenly
 - Alineación de los items respecto al eje principal
 - Ayuda a distribuir el espacio libre en el contenedor

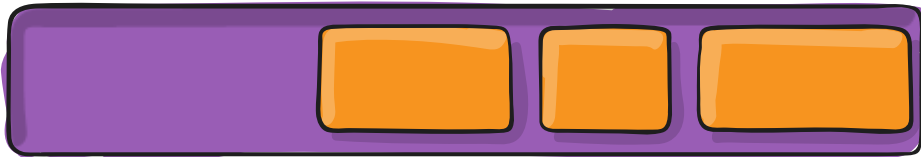
Flex – Propiedades del container

- justify-content

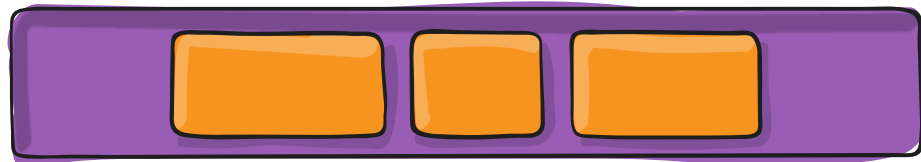
flex-start



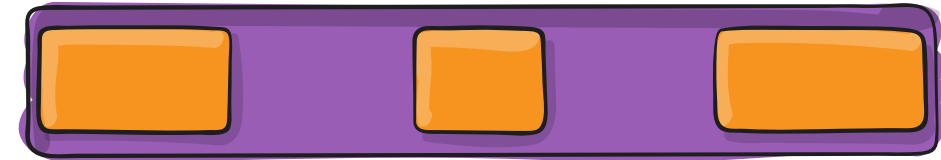
flex-end



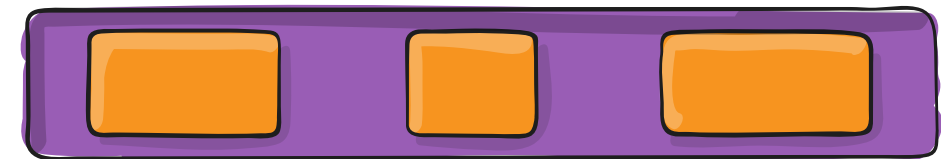
center



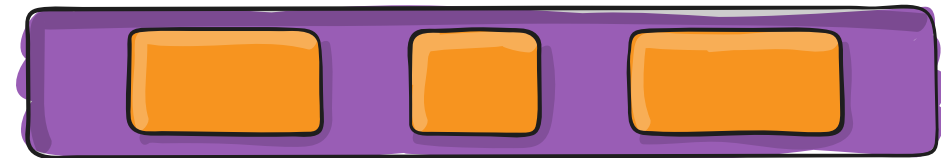
space-between



space-around

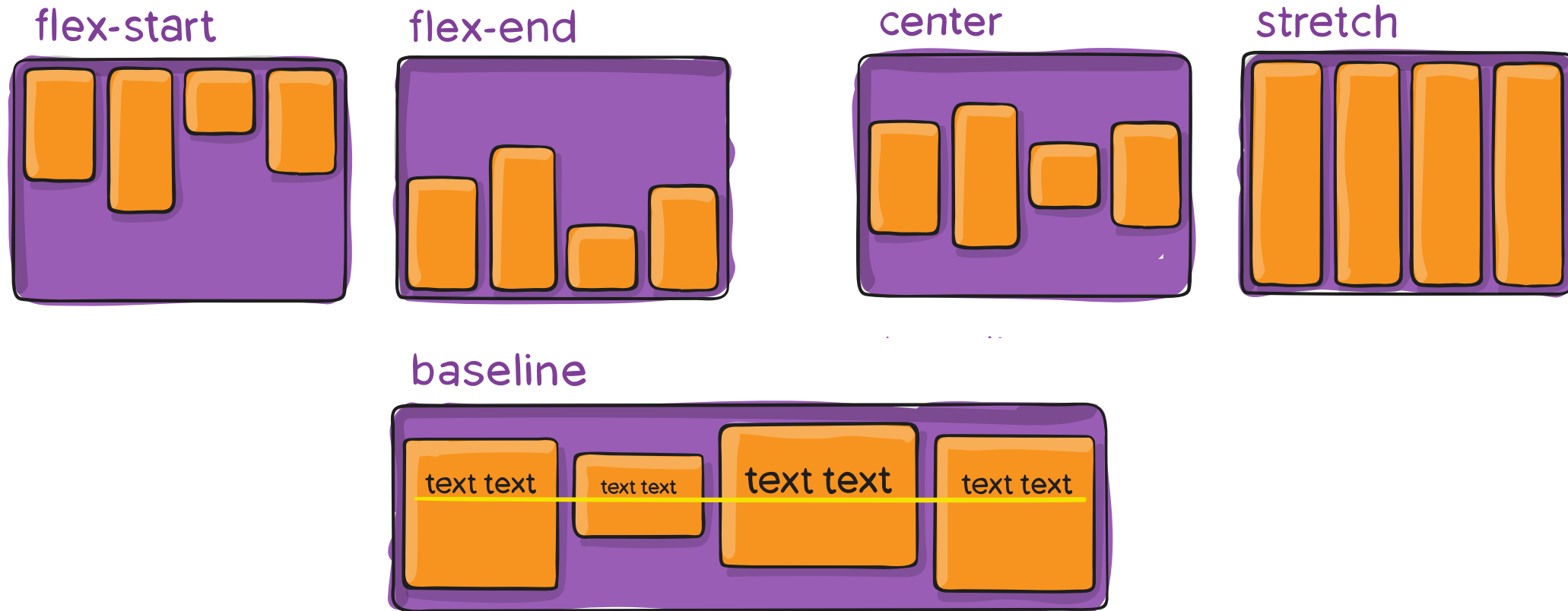


space-evenly



Flex – Propiedades del container

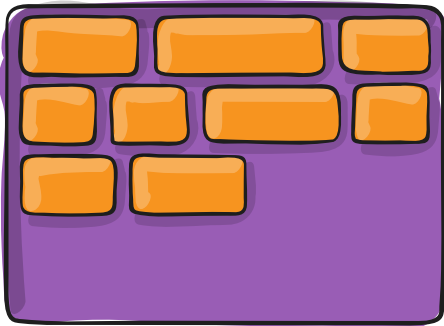
- align-items:
 - stretch, flex-start, flex-end, center, baseline
 - Comportamiento respecto al eje transversal (cross axis)



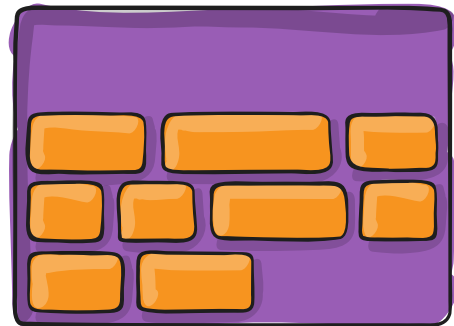
Flex – Propiedades del container

- align-content:
 - flex-start, flex-end, center, space-between, space-around, stretch
 - Alineación de los items respecto al eje transversal, ayuda a distribuir el espacio libre en el contenedor

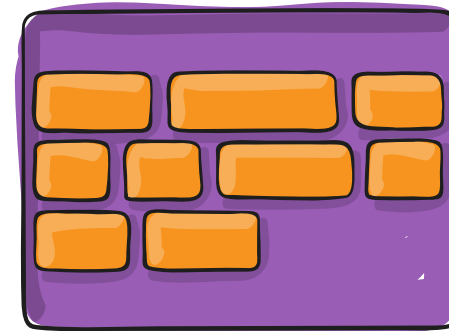
flex-start



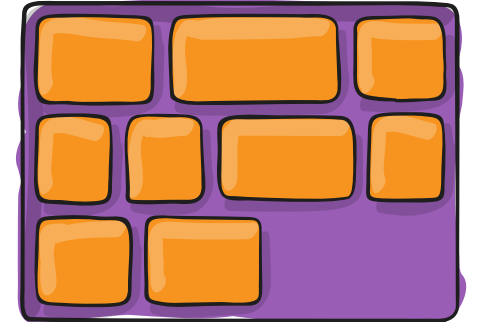
flex-end



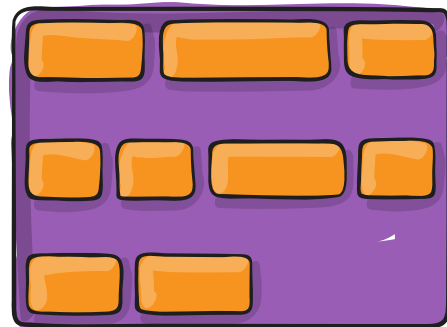
center



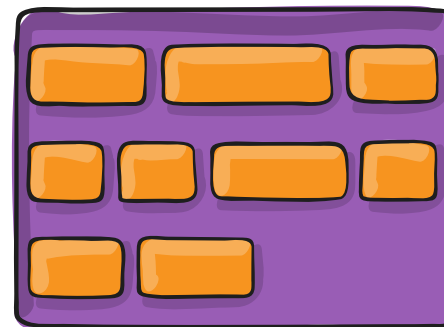
stretch



space-between

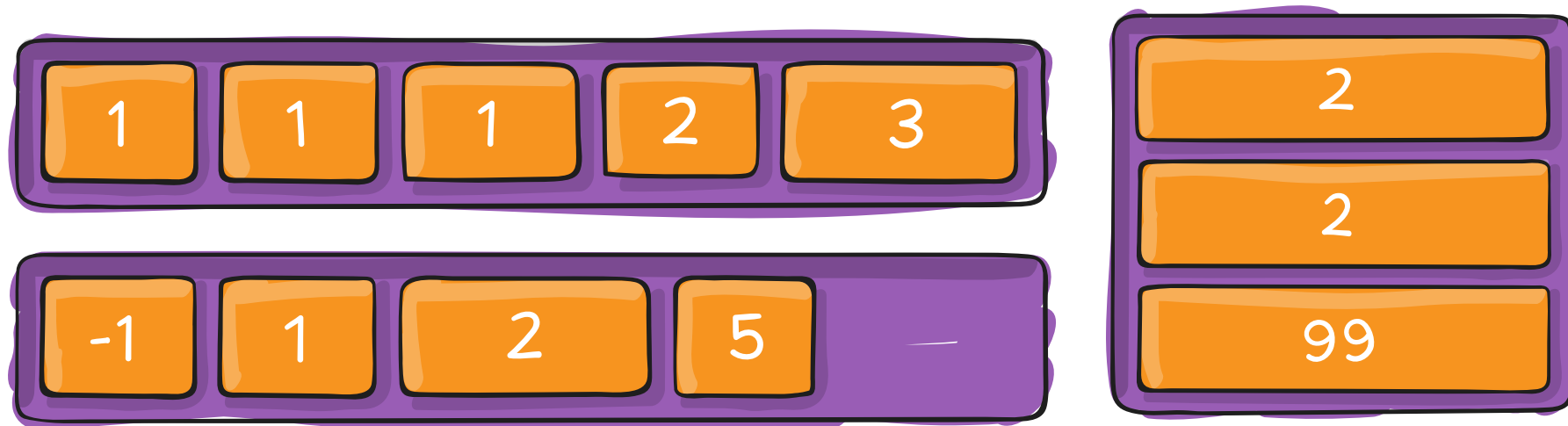


space-around



Flex – Propiedades de los items

- order:
 - Determina el orden del elemento.
 - Por defecto es 0: se respeta el orden de los elementos.
 - Si se usa un número distinto de cero, permite mover uno o varios elementos el elemento al inicio o al final (según el sentido del eje principal).

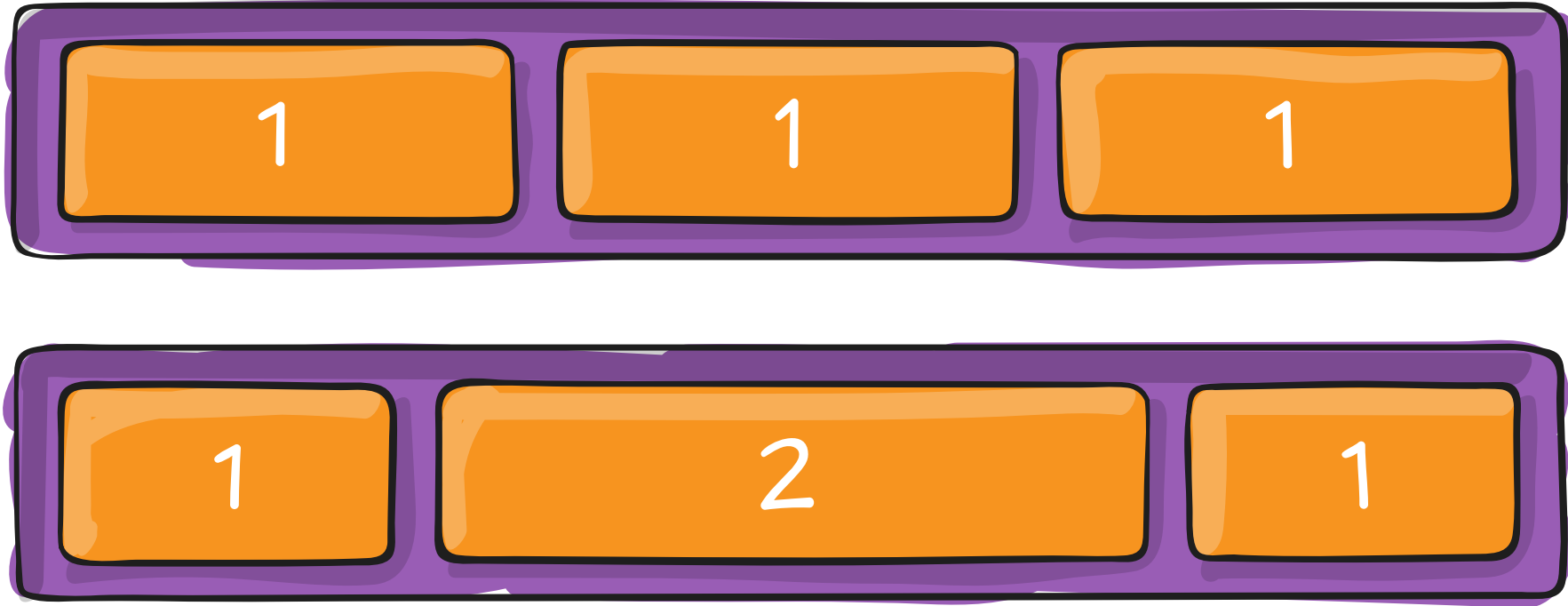


Flex – Propiedades de los items

- flex-basis
 - Define el tamaño base de los flex ítems antes de distribuirlos.
 - Se puede establecer un tamaño inicial (fijo, porcentaje, etc.)
 - Si no se especifica, es como si valiera "content", que asume el contenido como tamaño base.
- flex-grow / flex-shrink:
 - Define si un elemento debe ser del tamaño base o tiene que ser más grande / pequeño
 - Es un valor numérico (sin unidades) que se usa para calcular la proporción. Si es 2, usará el doble / la mitad de espacio que el resto, y así con el 3, 4, 5, etc.

Flex – Propiedades de los items

- flex-basis "content" con flex-grow:

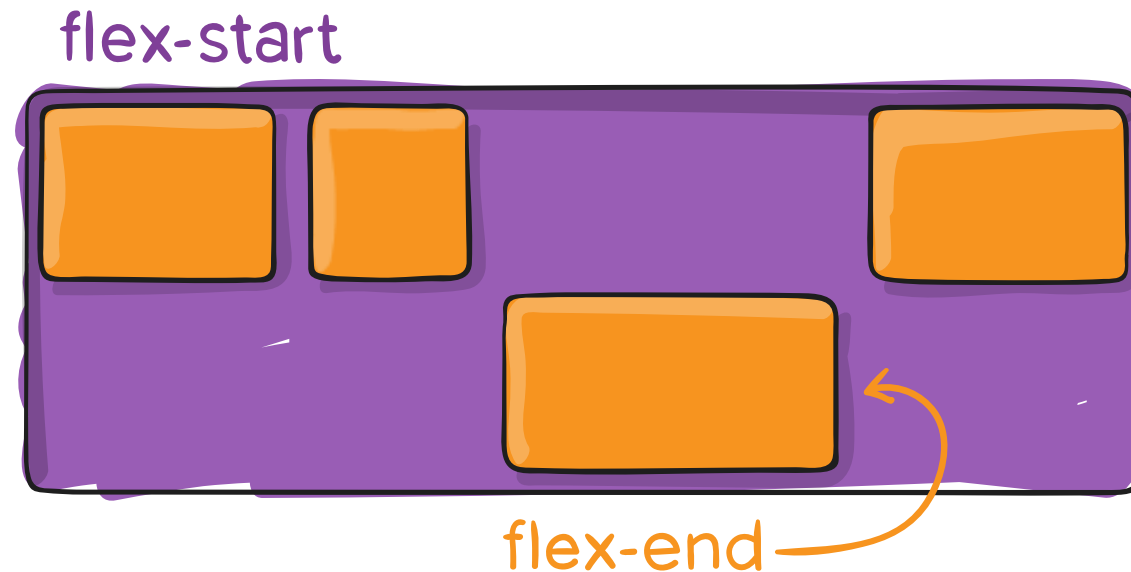


Flex – Propiedades de los items

- flex:
 - Atajo para las propiedades flex-grow, flex-shrink y flex basis.
 - La segunda y tercera (flex-shrink y flex basis) son opcionales.
 - Por defecto es “0 1 auto”.
 - Se recomienda usarlo en lugar de usarlas independientemente.

Flex – Propiedades de los items

- align-self:
 - auto, flex-start, flex-end, center, baseline, stretch
 - Sobrescribe la alineación establecida por “align-items” en el contenedor.
 - Permite alinear un item de forma diferente al resto



Flex – Consideraciones finales

- CSS solo ve la jerarquía de contenedor-item
- No aplicará propiedades Flex a elementos que no estén directamente relacionados, es decir, elementos descendientes que estén dentro de los flex-items
- Para que las propiedades de los ítems funcionen, el contenedor debe tener propiedad display: flex / inline-flex.
- Las propiedades float, clear y vertical-align no tienen ningún efecto en flex-items.

Recursos y juego de práctica

- Referencia y tutoriales:

<https://lenguajecss.com/css/maquetacion-y-colocacion/flex/>

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

https://www.w3schools.com/csS/css3_flexbox.asp

<https://www.tutorialrepublic.com/css-tutorial/css3-flexible-box-layouts.php>

- Práctica online:

<https://flexboxfroggy.com>

<http://www.flexboxdefense.com>

<https://codingfantasy.com/games/flexboxadventure/play>

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox_skills

Lenguajes de marcas y sistemas de gestión de la información



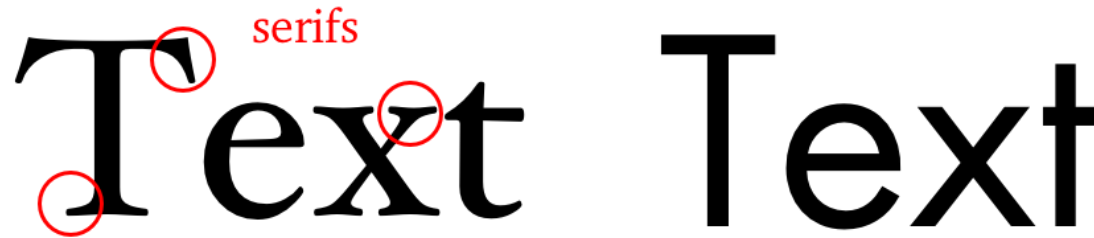
UT03 – CSS

9 – Tipografía – Fuentes – Texto

Tipografía – Serif vs Sans-serif

En términos generales, hay dos tipos de tipografías:

- Serif, con serifas: con "rabitos" o terminaciones al final de los trazos de la fuente.
- Sans serif, sin serifas, paloseco: sin terminaciones en los trazos.



The image displays two versions of the word "Text". The first version is in a serif font, with red circles highlighting the small decorative strokes (serifs) at the ends of the letters. The second version is in a sans-serif font, which lacks these decorative strokes.

Serif Font

Sans Serif Font

Habitualmente, encontramos las fuentes serif en medios impresos. Se supone que mejoran la legibilidad.

En medios digitales se suelen usar más las fuentes sans-serif porque permiten un aspecto más "limpio".

Tipografía – Proporcional vs monospace

En cuanto a cómo se distribuyen los caracteres tenemos:

- Monospace: Todos los caracteres ocupan el mismo espacio.
- Proporcional: Cada carácter ocupa sólo lo necesario. Caracteres como el 1 o la l minúscula ocupan menos que la M mayúscula.

Las fuentes monospace son buenas para mostrar datos tabulares en texto.

También es muy habitual que las fuentes que se usan en los entornos de desarrollo sean monospace. Hace el código más fácil de leer y de alinear.

Proportional

Monospace

Tipografía – Peso

El peso de una tipografía es su grosor.

El peso se expresa con un número que va, por lo general, de 100 a 900.

El peso considerado "normal" es el centro de esta escala, es decir, 400.

Es importante entender que:

- No todas las tipografías disponen de todos los pesos.
- En algunos casos, para usar un peso específico necesitaremos usar una versión específica de una fuente.
- Hay algunos casos en que los pesos puede bajar de 100 o sobrepasar 900, pero no es lo habitual..

Tipografía – Interlineado e interletraje

En tipografía hay dos formas de definir el espacio entre elementos de texto:

- Vertical. Interlineado. La separación entre dos líneas de texto.
- Horizontal: Interletraje, interletrado, o kerning. La separación entre caracteres.



Fuentes

Las fuentes son ficheros que contienen la información necesaria para para mostrar caracteres de cierta tipografía.

Almacenan información sobre la forma, tamaño, posición de los trazos de cada carácter. También sobre el peso e interletraje.

Los ficheros de fuente suelen estar instalados en el sistema, pero con CSS se puede forzar la descarga de ciertos ficheros de fuentes, para garantizar un formato correcto del sitio web.

Los formatos más habituales en web son .woff (Web Open Font Format) y .woff2, aunque también se usan los formatos .ttf (True Type Font).

Fuentes en CSS – font-family

Especifica una familia de fuentes.

Una familia de fuentes está formada por una o más fuentes:

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

Con esto estamos expresando una prioridad. Estamos indicando que, siempre que se pueda, se utilice arial. Si no hay arial, helvética. Y si no hay ninguna de las dos, que el sistema elija una fuente sans-serif.

La última opción, es un valor genérico, denominado "fuente segura". Al indicar este valor, nos aseguramos de que, al menos, no usará una fuente serif donde deseamos sans serif o de otro tipo.

Las fuentes seguras más habituales son serif, sans-serif, cursive, fantasy y monospace.

Fuentes en CSS – font-size

Indica el tamaño de la fuente. Puede indicarse de tres formas:

- De forma absoluta: xx-small, x-small, small, medium, large, x-large, -xxlarge.
- Relativas al tamaño actual (heredado, por ejemplo): smaller, larger.
- Medidas específicas. Pueden usarse todas las unidades de distancia y dimensión que vimos hace unos días. Por ejemplo:
 - Absolutas: px, cm.
 - Relativas: rem, em, %

Fuentes en CSS – font-style

Aplica variaciones de estilo al texto

- normal: Estilo normal de la fuente. Valor por defecto. Permite volver al estilo normal si la herencia o cascada está afectando a la fuente.
- italic: Cursiva. El navegador usará, siempre que esté disponible, la versión inclinada de la fuente. Si no se dispone de una versión inclinada, intentará emularla con oblique.
- oblique: Igual que cursiva, pero inclina la fuente "recta" de forma artificial, normalmente 14°. Se puede indicar un grado de inclinación menor o mayor usando unidades deg

```
font-style: oblique 30deg;
```

Fuentes en CSS – font-weight

Indica el grosor de la fuente

- normal / bold: Predefinido (400) / negrita (700).
- bolder / lighter: hace más o menos gruesa la fuente, en función del valor que se le aplicaba por cascada o herencia.
- Valor numérico de 100 a 900, de 100 en 100.
 - 100 – Thin, 200 – Extra light, 300 – Light
 - 400 – Normal, 500 – Medium
 - 600 – Semi bold, 700 – Bold, 800 – Extra bold, 900 – Heavy

No todas las tipografías / fuentes soportan todos los valores.

Fuentes en CSS – @font-face

Las fuentes pueden ser problemáticas. Si un usuario no tiene ninguna de las fuentes "no seguras" declaradas en font-family, acabará usando la segura, y no es lo ideal.

@font-face nos permite indicar al navegador desde qué url se puede descargar el fichero de fuente (woff, woff2, etc.)

```
@font-face {  
    font-family: 'Dancing Script';  
    font-style: normal; /* Opcional */  
    font-weight: 400; /* Opcional */  
    font-display: swap; /* Opcional */  
    src: url(https://fonts...woff2) format('woff2');  
}
```

Fuentes en CSS – @font-face

Las únicas partes obligatorias son "font-family" y "src". En src se pueden poner distintos ficheros con la fuente, de forma que el navegador use el que más le convenga. Por compatibilidad, por ejemplo.

Es buena práctica empezar con "local("fuente")", para que el navegador tome la fuente de sus fuentes locales, si es posible. Luego se deberían poner las fuentes woff2 y woff, que son las que mejor soporte tienen.

```
@font-face {  
    font-family: 'Dancing Script';  
    src:  
        local('Dancing Script'),  
        url(https://.....woff2) format('woff2'),  
        url(https://.....woff) format('woff'),  
        url(https://.....ttf) format('ttf');  
}
```

Texto – Espacios

Por defecto, HTML "compacta" los espacios en blanco, tabuladores y saltos de línea, de modo que no son visibles en la página.

Se puede forzar la visualización de un espacio con la entidad , pero css tiene propiedades para gestionar espacios en blanco consecutivos y el ajuste del texto.

- white-space: comportamiento de
 - normal: Se compactan, y el texto se ajusta. Valor por defecto.
 - nowrap: Se compactan, e ignora saltos de línea. No se ajusta.
 - pre: Se respetan y muestran espacios literalmente. No se ajusta.
 - pre-wrap: Como pre, pero se ajusta.
 - pre-line: Como pre-wrap, pero sólo respeta los espacios al principio.

Texto – Alineaciones

- `text-align`: alineación de texto. También aplica a los elementos dentro de un contenedor (por ejemplo, una imagen en un `div`).
 - `start`: Al principio (equivalente a `left`, en modelo "antiguo")
 - `end`: Al final (equivalente a `right`)
 - `center`: centrado
 - `justify`: justificado. Se ocupa toda la línea. No se recomienda.
 - `match-parent`: la misma que el elemento padre.
 - `justify-all`: usa `justify` en `text-align` y en `text-align-last`
- `text-align-last`: igual que `text-align`, pero sólo aplica a la última línea del texto.

Texto – Alineaciones

- vertical-align: Alineación vertical. Útil para alinear texto junto a imágenes:
 - baseline: Valor por defecto. Base con base.
 - sub/super: Para subíndices y superíndices.
 - top: Alineadas las partes superiores.
 - middle: Centros alineados
 - bottom: Alineadas las partes inferiores
 - text-top: Parte superior con la parte superior del texto que contiene el elemento.
 - text-bottom: Parte inferior con la parte inferior del texto que contiene el elemento.

Texto – Espaciado

- Espaciados
 - letter-spacing: Espaciado entre letras (interletraje)
 - word-spacing: Espaciado entre palabras
 - line-height: Alto de línea, es el equivalente a interlineado.
- Sangría:
 - text-indent: Sangría. Indenta la primera línea de texto la medida indicada. Puede incluir los modificadores:
 - hanging: invierte la sangría, indenta el resto del texto, no la primera línea.
 - each-line: afecta también en los

Texto – Decoraciones

- `text-decoration-line`: Tipo de decoración (posición de la línea) .
`none / underline / overline / line-through`
- `text-decoration-style`: Tipo de línea.
`solid / double / dotted / dashed / wavy`
- `text-decoration-color`: color de la decoración
- `text-decoration-thickness`: grueso de la decoración
- `text-underline-position`: posición de la decoración
`auto / from-font / under`
- `text-underline-offset`: desplazamiento del trazo de decoración
- `text-decoration`: Atajo para `line`, `thickness`, `style` y `color`.

Texto – Transformaciones

- text-transform: Transforma el texto. Puede ser:
 - capitalize: A mayúsculas la primera letra de cada palabra.
 - uppercase: A mayúsculas todas las letras.
 - lowercase: A minúsculas todas las letras.

Hay que tener en cuenta que CSS sólo cambia la apariencia del texto.

Si copiamos un texto que está marcado con, por ejemplo, uppercase, lo copiaremos con su forma original, si era mezcla de mayúsculas y minúsculas lo seguirá siendo.