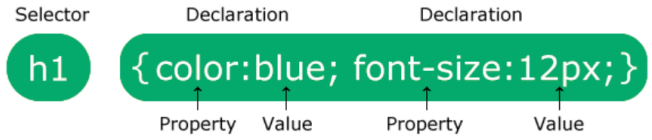
T.2. Uso de estilos

2.1. Introducción

- CSS son las siglas de Cascading Style Sheets (Hojas de estilo en cascada)
- CSS describe cómo se deben mostrar los elementos HTML en la pantalla, en papel, o en otros medios

2.2. Sintaxis. Una regla CSS consta de un selector y un bloque de declaración.



El selector apunta al elemento HTML al que desea aplicar estilo.

El bloque de declaración contiene una o más declaraciones separadas por punto y coma.

Cada declaración incluye un nombre de propiedad CSS y un valor, separados por dos puntos.

Varias declaraciones CSS se separan con punto y coma y declaración Los bloques están rodeados por llaves.

2.3. ¿Cómo insertar css?

- CSS externo→ <link rel="stylesheet" href="mystyle.css">
- CSS interno —>

```
 <head>

<style>

body {

  background-color: linen;
```

- CSS en línea (inline)

```
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>
```

2.4. Colores css→ font, bg and border color. Values in RGB, RGBA, hex etc.
Los colores se muestran combinando luz ROJA, VERDE y AZUL.

Los colores se pueden especificar de diferentes maneras:

- Por nombres de color
- Como valores RGB (compatibles con todos los navegadores.)

Un valor de color RGB se especifica con: rgb(ROJO , VERDE , AZUL).

Cada parámetro define la intensidad del color como un número entero entre 0 y 255.

- Como valores hexadecimales (compatibles con todos los navegadores.)

Un color hexadecimal se especifica con: #RRGGBB.

RR (rojo), GG (verde) y BB (azul) son enteros hexadecimales entre 00 y FF especificando la intensidad del color.

- Como valores HSL (CSS3)
- Como valores HWB (CSS4)
- Con la palabra clave currentcolor

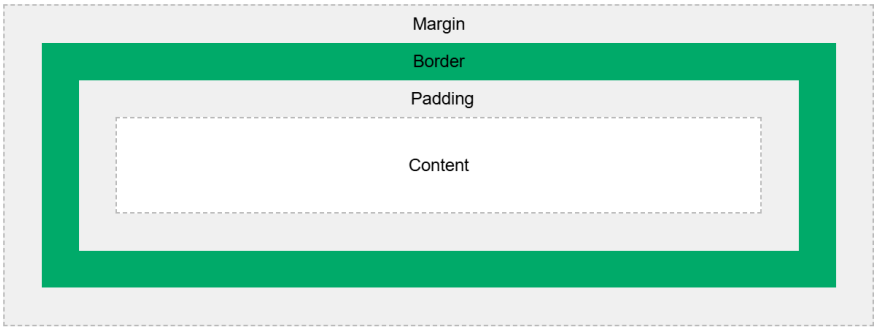
2.5. Background Image

```
body {background-image: url("paper.gif");}
```

2.6. Opacity. Especifica la opacidad/transparencia de un elemento. The lower the value, the more transparent

```
img { opacity: 0.5;}
```

2.7. Box Model (Todos los elementos HTML pueden ser considerados como cajas). El modelo de caja CSS es esencialmente una caja que envuelve cada elemento HTML. Se compone de: contenido, relleno, bordes y márgenes.



- Contenido: el contenido del cuadro, donde aparecen el texto y las imágenes.
- Relleno: despeja un área alrededor del contenido. El acolchado es transparente. Generate space around an element’s content
- Borde: un borde que rodea el relleno y el contenido. Specify the style, width, and color of an element's border
- Margen: despeja un área fuera del borde. El margen es transparente. To create space around elements

2.7.1. Box-Sizing

2.7.2. Height/Width (Se utilizan para establecer la altura y anchura de un elemento)

- max-height min-height max-width min-width

- No incluyen relleno, bordes ni márgenes; solo afectan al área de contenido.

Valores posibles:

- auto: Valor predeterminado, el navegador calcula el tamaño.
- length (px, cm, etc.): Especifica un tamaño fijo.
- %: Define el tamaño como un porcentaje del contenedor padre.
- initial: Restaura el valor predeterminado.
- inherit: Hereda el valor de su elemento padre.

Propiedad max-width (Limita el ancho máximo de un elemento)

- Evita que un elemento sea más ancho que el valor especificado, lo que mejora la visualización en ventanas pequeñas y evita barras de desplazamiento horizontales.
- Si se utilizan width y max-width juntas, se prioriza max-width si hay conflicto.

Ancho (width) y Márgenes Automáticos

- Un **elemento de nivel de bloque** ocupa todo el ancho disponible del contenedor de forma predeterminada.
- **Establecer width** en un valor específico (por ejemplo, 500px) evita que el elemento se estire completamente.
- **margin: auto** centra el elemento horizontalmente dividiendo el espacio restante en partes iguales a ambos lados.

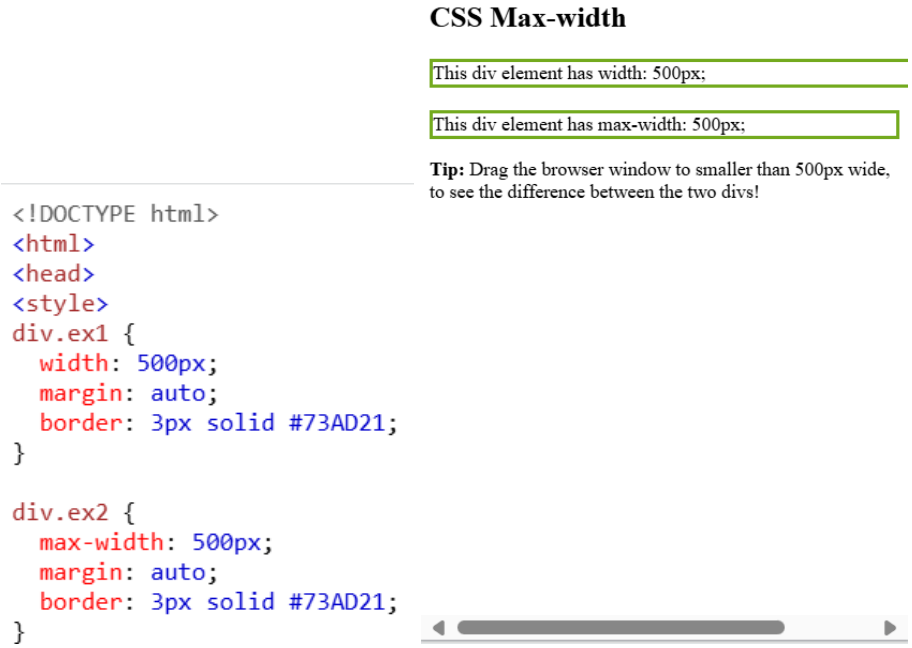
Problema: Si el ancho del elemento es mayor que la ventana del navegador, se genera una barra de desplazamiento horizontal.

Ancho Máximo (max-width)

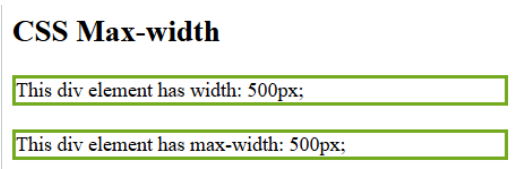
- **max-width** se utiliza para mejorar la adaptabilidad, permitiendo que el elemento reduzca su tamaño en ventanas pequeñas sin necesidad de una barra de desplazamiento.
- Esto es especialmente útil para garantizar que los sitios sean más usables en dispositivos pequeños.

width: 500px: Fijo, puede causar desplazamiento horizontal.

max-width: 500px: Adaptativo, evita el desplazamiento al reducirse en pantallas pequeñas.

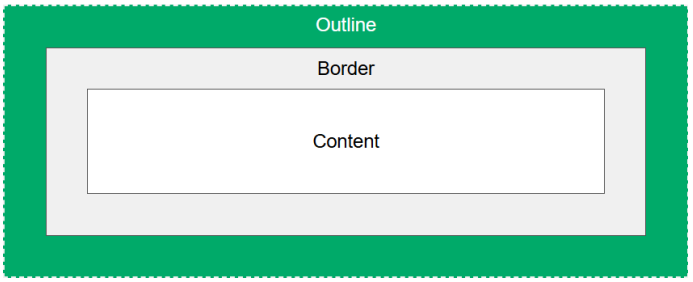



```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}

div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```




```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}

div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
```

2.7.3. Outline. El outline en CSS es una línea dibujada fuera del borde de un elemento, lo que ayuda a que el elemento destaque. Esta línea no se incluye en las dimensiones del elemento, por lo que no afecta su altura o anchura totales.



Propiedades del Outline en CSS

- outline-style: Establece el estilo de la línea y puede ser:

  - `dotted` (punteado)

  - `dashed` (discontinuo)

  - `solid` (sólido)

  - `double` (doble)

  - `groove`, `ridge`, `inset`, `outset` (efectos 3D)

  - `none` o `hidden` (sin contorno)

- outline-color: Define el color.

- outline-width: Define el grosor.

- outline-offset: Separa el outline del borde del elemento.

Diferencias con los Bordes. El outline se dibuja fuera del borde del elemento y puede superponerse a otros contenidos, a diferencia del borde que está dentro del espacio del elemento. Además, el outline no se incluye en las dimensiones totales del elemento.

2.8. Fuentes e iconos

2.8.1. Texto (Propiedades Principales para Formatear Texto)

- **color:** Define el color del texto usando nombres de color, valores HEX o RGB.
- **text-align:** Alinea el texto (izquierda, derecha, centro, justificar)**.**
- **text-transform:** Controla la capitalización del texto (mayúsculas, minúsculas, capitalización de cada palabra).
- **text-indent:** Indenta la primera línea de un párrafo.
- **letter-spacing:** Ajusta el espacio entre caracteres.
- **text-decoration:** Modifica o elimina subrayados y otras decoraciones.

2.8.2. Fuentes

Familias Genéricas de Fuentes en CSS:

1. **Serif**: Con trazos en los bordes de las letras; da una sensación formal y elegante (ej. *Times New Roman*, *Georgia*).
2. **Sans-serif**: Sin trazos, líneas limpias y aspecto moderno (ej. *Arial*, *Verdana*).
3. **Monoespaciadas**: Letras de ancho fijo, aspecto mecánico (ej. *Courier New*, *Monaco*).
4. **Cursivas**: Imitan la escritura manual (ej. *Brush Script MT*).
5. **Fantasía**: Fuentes decorativas o creativas (ej. *Papyrus*).

Propiedad font-family: Se usa para especificar la fuente de un texto. Se recomienda incluir múltiples fuentes como respaldo, terminando con una familia genérica para asegurar la compatibilidad entre navegadores.

.p1 { font-family: "Times New Roman", Times, serif; }

2.8.3. Iconos

- La manera más sencilla es usar una biblioteca de íconos, como Font Awesome, Bootstrap Icons o Google Icons.

- Para agregar íconos a tu página HTML usando una biblioteca de íconos, simplemente debes añadir la clase del ícono especificado a un elemento en línea, como <i> o .

- Usar un **CDN links (Content Delivery Network)** es la forma más sencilla y eficiente de cargar bibliotecas, como las de íconos, ya que estas pueden mantenerse en la caché del navegador, acelerando la carga de la página.

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">

-Have a look at the FA Tutorial in W3Schools

2.9. Link, list y tabla

2.9.1. Link. Los pseudoclases en CSS se utilizan para modificar el comportamiento de un enlace y aplicar estilos específicos según el estado en el que se encuentre

Estados de los Enlaces

1. **a:link -** Enlace normal y no visitado.
2. **a:visited -** Enlace que el usuario ya ha visitado.
3. **a:hover -** Enlace cuando el usuario pasa el cursor por encima.
4. **a:active -** Enlace en el momento en que se hace clic.

Reglas de Orden

- **a:hover** debe venir después de **a:link** y **a:visited.**
- **a:active** debe venir después de **a:hover**

/* Enlace activo */ a:active { color: blue; }

2.9.1. Lista (Tipos de listas en HTML)

1. **Listas desordenadas (<ul>):** Utilizan viñetas como marcadores.

2. **Listas ordenadas (<ol>):** Utilizan números o letras para marcar los elementos.

Personalización de listas con CSS:

list-style-type: Define el tipo de marcador (por ejemplo, círculo, cuadrado, números romanos).

ul.a { list-style-type: circle; }

list-style-type: none; se utiliza comúnmente cuando se crean menús basados en listas desordenadas (<ul>). Esto permite eliminar las viñetas predeterminadas, logrando un aspecto limpio y ordenado ideal para menús de navegación en sitios web.

list-style-image: Utiliza una imagen como marcador de la lista.

list-style-position:

- **outside**: El marcador queda fuera del contenido de la lista (predeterminado).
- **inside**: El marcador se alinea con el texto de la lista, desplazándolo al inicio.

2.9.2. Tablas (css para estilizar tablas)
○ **IMPORTANT: Tables are used to display text, not to display images**

Aplicación de Bordes en Tablas. Para aplicar bordes a toda la tabla y a sus celdas, usa la propiedad border. Esto hará visibles los bordes separados para los elementos <table>, <th>, y <td>.

table, th, td {
border: 1px solid; }

Cuando se aplican bordes separados a <table>, <th>, y <td>, pueden aparecer bordes dobles. Para evitar esto, puedes colapsar los bordes en una sola línea.

Firstname	Lastname
Peter	Griffin
Lois	Griffin

border-collapse permite fusionar los bordes en una sola línea.

table { border-collapse: collapse; }

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Si solo quieres aplicar un borde alrededor de la tabla sin añadir bordes a las celdas, especifica la propiedad **border** solo en **<table>**. table { border: 1px solid; }

Firstname	Lastname
Peter	Griffin
Lois	Griffin

3. Vendor-prefixed properties (Propiedades de los prefijos de proveedores)

[-webkit- -moz -ms -o] —> Los prefijos de proveedor son esenciales para asegurar que los estilos CSS funcionen de manera consistente en diferentes navegadores.

T.3. Control del diseño (layout)

3.1. Propiedad Display

display: block: El elemento siempre comienza en una nueva línea y ocupa todo el ancho disponible.

display: inline: El elemento solo ocupa el ancho que necesita. No puede tener valores de altura o ancho.

display: inline-block: Es un elemento en línea con altura y ancho. Por ejemplo, una imagen tiene, por defecto, el valor inline-block. A diferencia de display: inline, que no permite definir estos valores. En comparación con display: block, la diferencia principal es que inline-block no agrega un salto de línea después del elemento, lo que permite que se alineen horizontalmente junto a otros elementos.

display: flex: El contenedor toma el tamaño que necesita para sus elementos.

display: none: El elemento desaparece del flujo del documento (a diferencia de visibility: hidden, que oculta el elemento pero mantiene su espacio).

3.2. Propiedad Overflow (desbordamiento)

- Controla lo que sucede con el contenido que es demasiado grande para caber en un área.
- Solo funciona con elementos que tienen una altura y/o anchura especificadas.
- Valores posibles: visible, hidden, scroll, auto.
- Propiedades overflow-x y overflow-y para controlar solo el desbordamiento horizontal o vertical.

3.3. Propiedad position

- position: static (predeterminado): el elemento se posiciona de acuerdo con el flujo del documento.
- position: relative: se posiciona en relación a su posición predeterminada utilizando las propiedades top, left, bottom y right.
- position: absolute: se posiciona en relación al ancestro más cercano que tenga una posición diferente de estática, utilizando las propiedades top, left, bottom y right.
- position: sticky: si haces scroll, se queda fijado en la parte superior de la página. Más sobre STICKY.
- position: fixed: posición fija en relación con la ventana gráfica (viewport). No se moverá al hacer scroll.

Usa z-index para elementos que se superponen a otros elementos. Nota: z-index solo funciona en elementos posicionados (con position: absolute, position: relative, position: fixed, o position: sticky) y en elementos flexibles (elementos que son hijos directos de elementos con display: flex).

3.4. Propiedad Float y Clear

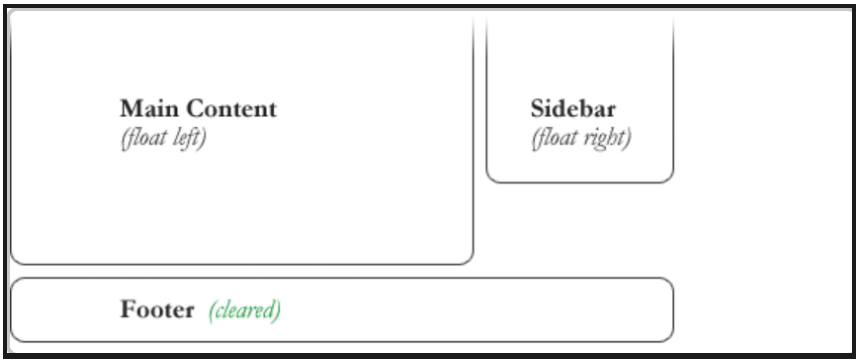
La propiedad float puede tener los siguientes valores:

- left: hace que el elemento flote hacia la izquierda.
- right: hace que el elemento flote hacia la derecha.
- none: el elemento no flota (valor predeterminado).
- inherit: hereda el valor de su elemento padre.

La propiedad clear especifica qué elementos pueden flotar al lado del elemento que ha sido "limpiado". Los valores para clear pueden incluir: (La propiedad clear está directamente relacionada con los elementos flotantes (floats). Si un elemento puede caber horizontalmente en el espacio al lado de otro elemento que está flotando, lo hará, a menos que se aplique clear en la misma dirección que el float. En ese caso, el elemento se moverá hacia abajo, quedando por debajo del elemento flotante.)

- left: el elemento no permitirá que los elementos flotantes a la izquierda se coloquen a su lado.
- right: el elemento no permitirá que los elementos flotantes a la derecha se coloquen a su lado.
- both: el elemento no permitirá que los elementos flotantes a la izquierda o a la derecha se coloquen a su lado.
- none: permite que los elementos flotantes se coloquen a su lado (valor predeterminado).


```
#footer {

  clear: both;

}
```



3.5. Alineación Horizontal y Vertical

margin: 0 auto; /* Centra el elemento horizontalmente */

T.4. Selectores css

4.1. Selectores básicos

1. Selector Universal

    - Selecciona todos los elementos. Por ejemplo, * coincide con todos los elementos del documento.
2. Selector de Tipo
    - Selecciona todos los elementos que tienen el nombre de nodo dado. Por ejemplo, input coincide con cualquier <input>.
3. Selector de Clase
    - Selecciona todos los elementos que tienen la clase especificada. Por ejemplo, .index coincide con todos los elementos que tienen la clase index.
4. Selector de ID
    - Selecciona elementos basados en el valor de su atributo id. Por ejemplo, #toc coincide con el elemento que tiene el id toc.
5. Selector de Atributo
    - Selecciona todos los elementos que tienen el valor indicado (o un valor) en un atributo dado.
    - Ejemplo: [attr=value] coincide con todos los elementos cuyo atributo attr está establecido en un valor específico.
    - Ejemplo 2: [id] coincide con todos los elementos que tienen el atributo id configurado (a cualquier valor).

4.2. Selectores Agrupados

- Usando la coma (,), podemos agrupar nodos para aplicarles un estilo.
    - **Sintaxis:** A, B
    - **Ejemplo:** div, span coincidirá con ambos elementos y .

4.3. Combinadores

- **Combinador descendiente (space)**
    - Selecciona todos los descendientes (no solo los hijos).
- **Combinador de hijos >**
    - Selecciona todos los hijos (no todos los descendientes).
- **Combinador de hermanos general ~**
    - Selecciona elementos que son vecinos o hermanos (no necesariamente inmediatos).
- **Combinador de hermanos adyacentes +**
    - Selecciona el elemento que es el vecino inmediato (posterior, no anterior).

4.4. Pseudo-clases

Una pseudo-clase define un estado especial de un elemento.
Una pseudo-clase se puede usar para:

- Estilizar un elemento cuando el usuario pasa el mouse sobre él.
- Estilizar enlaces visitados y no visitados de manera diferente.
- Estilizar un elemento cuando recibe el foco.

selector:pseudo-class { property: value; }

El **:** (dos puntos) permite la selección de elementos basándose en información de estado que no está contenida en el árbol del documento.

Existen muchas pseudoclases:

- **Acciones del usuario:**

  :link→ representa un elemento que aún no se ha visitado.

  a:link {

  color: forestgreen;

  text-decoration-color: hotpink;

  }

  :visited→ una vez que el enlace ha sido visitado por el usuario.

  a:visited {

  color: forestgreen;

  text-decoration-color: hotpink;

  }

  :hover→ se activa cuando el usuario pasa el cursor sobre un elemento (puntero del ratón).

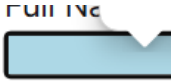
  .joinBtn:hover {

  background-color: bisque;

  }

  :active→ Cuando se usa un mouse, la "activación" generalmente comienza cuando el usuario presiona el botón principal del mouse.

  :focus→ se activa cuando el usuario hace clic o toca un elemento o lo selecciona con la tecla del teclado.



- **Para inputs:**

  :valid→ se aplica a los elementos de formulario que cumplen con las restricciones de validación definidas en el HTML.
  input:valid {
  border: 2px solid green; /* Cambia el borde a verde si el valor es válido */
  }

  :invalid, etc.

- **Estructura del árbol:**

  :root→ Selecciona el elemento raíz del documento, que generalmente es el <html>.

:first-child→ Selecciona el primer hijo de su elemento padre. Puede ser utilizado con cualquier tipo de elemento.

Ejemplo: p:first-child selecciona cualquier <p> que sea el primer hijo de cualquier elemento.

Ejemplo: p i:first-child selecciona el primer elemento <i> en todos los elementos <p>.

Ejemplo: p:first-child i selecciona todos los elementos <i> en elementos <p> que son el primer hijo.

:nth-child()--> Coincide con el hijo o los hijos que queremos modificar según un patrón específico.

- Pseudoclases pueden ser combinadas con clases CSS
- Crear interacciones usando pseudoclases

div:hover p { color:red } Text inside a p that is descendant of a div is colored in red on

hover the div

div p:hover { color:red } Text inside the p is colored in red when hovering on the p that

a descendant of a div

div.box:hover p { color:red } Text inside the p is colored in red on hover the div of class box

4.5. Pseudo-elementos. Se utilizan para estilizar una parte específica de un elemento:

selector::pseudo-element { property: value; }

- **::first-line:** Se aplica estilo a la primera línea de un elemento.

  p::first-line { color: #ff0000; font-variant: small-caps; }

- **::first-letter:** Se aplica a la primera letra del texto de un elemento.
- **::before:** Se utiliza para insertar contenido antes del contenido de un elemento.

  h1::before { content: url(smiley.gif); }

- **::after:** Se utiliza para insertar contenido después del contenido de un elemento.
- **::selection:** Coincide con la parte de un elemento que es seleccionada por un usuario

  ::selection { color: red; background: yellow; }

T.5. Unidades en css

5.1. ¿Por qué debería aprender las unidades de CSS?

- Puedes utilizar diferentes unidades según tus necesidades.
- El diseño de tu sitio será más fácil de mantener con las unidades de CSS adecuadas para el texto y los espacios.
- Especialmente útiles para el diseño responsivo

5.2. Unidades de longitud absolutas

- **px (píxeles)**
  - (1px = 1/96 de 1 pulgada)
  - El tamaño del píxel depende del dispositivo de visualización.
- **Otras (no utilizadas):**
  - **cm (centímetros)**
  - **mm (milímetros)**
  - **in (pulgadas)** (1in = 96px = 2.54cm)
  - **pt (puntos)** (1pt = 1/72 de 1 pulgada)
  - **pc (picas)** (1pc = 12 pt)

5.3. Unidades de longitud relativas

- Las unidades de longitud relativas especifican una longitud en relación con otra propiedad de longitud.
- Las más comúnmente utilizadas son:
  - **%**
    Utilizado en medidas de diseño (altura, ancho). Relativo al contenedor padre.
  - **em**
    Utilizado en tamaños de fuente. Relativo al tamaño de fuente heredado del elemento actual.
  - **rem**
    Utilizado en tamaños de fuente. Relativo al tamaño de fuente del elemento raíz del documento.
- Las menos utilizadas son:
  - **vw:** Relativo al 1% del ancho del viewport.
  - **vh:** Relativo al 1% de la altura del viewport.
  - **vmin:** Relativo al 1% de la dimensión más pequeña del viewport (altura o ancho).
  - **vmax:** Relativo al 1% de la dimensión más grande del viewport (altura o ancho).

T.6. Flexbox

Flexbox es un modelo unidimensional que distribuye los elementos dentro de un contenedor.

➔ Crea diseños flexibles y autónomos.

6.1. El contenedor

■ display: flex ¡OBLIGATORIO!

■ flex-direction Indica en qué dirección se muestran los elementos.

■ flex-wrap Indica si el contenido debe envolver o no.

■ flex-flow Propiedad abreviada para establecer tanto flex-direction como flex-wrap.

■ justify-content alineación horizontal de los elementos flexibles.

■ align-items alineación vertical de los elementos en cada fila, además de estiramiento y línea base.

■ align-content alineación de todas las filas - columnas.

6.2. Los elementos (items)

● **order**

- Establece un orden para cada elemento.
- A valores más altos de orden, el elemento se sitúa más atrás.

● **align-self**

- Permite alinear un elemento de manera independiente del resto.

● **flex** Propiedad abreviada para:

- **flex-grow** Aumenta el tamaño de un elemento de manera proporcional.
- **flex-shrink** Disminuye el tamaño de un elemento o mantiene su tamaño.
- **flex-basis**
  - Valor a partir del cual un elemento crece, si utilizamos flex-grow.
  - Valor a partir del cual no disminuye, si utilizamos flex-shrink: 0.

T.7. Grid

7.1. ¿Qué es Grid?

El diseño de layouts ha evolucionado desde el uso de tablas HTML hasta los elementos flotantes de CSS, y finalmente al CSS Grid.

● El diseño de CSS Grid crea una cuadrícula donde podemos colocar nuestros contenedores.

● Las tablas, los floats, el posicionamiento y el inline-block eran básicamente soluciones improvisadas que dejaban de lado muchas funcionalidades importantes.

● Flexbox ayudó, pero está destinado a diseños más simples y unidimensionales.

● Grid es el primer módulo de CSS creado específicamente para resolver problemas de diseño.

7.2. Template-areas

○ grid-template-areas: Establecemos una cuadrícula con los nombres de las secciones

"header header header"

"main main aside"

"main main aside"

"footer footer footer";

○ grid-area: El nombre de cada sección se aplica a un elemento con CSS.

7.3. Introducción

● El contenedor

○ **display: grid o inline-grid**
○ **grid-template-columns**: ancho de cada columna de la cuadrícula (y nombre de las líneas)
○ **grid-template-rows**: altura de cada fila de la cuadrícula (y nombre de las líneas)
○ **grid-column-gap, grid-row-gap**: espacio entre filas y entre columnas

● El elemento (items)
○ **grid-column**: columnas donde el elemento comienza/termina
○ **grid-row**: fila donde el elemento comienza/termina

7.4. El contenedor

Grid se combina con las propiedades de Flexbox para crear diseños.

○ **justify-content:** Para alinear horizontalmente dentro del contenedor.

Valores: **space-around, space-evenly, space-between, center, start, end.**

○ **align-content:** Para alinear verticalmente dentro del contenedor.

Valores: **space-around, space-evenly, space-between, center, start, end.**

7.5. Contenido extra

- **CSS Grid vs. otros métodos de diseño**

○ Flexbox funciona principalmente en 1 dimensión.

○ Grid es mejor para diseños de 2 dimensiones.

○ **Enfoque de diseño**: Flexbox adapta el diseño en función del contenido ("de adentro hacia afuera"), mientras que grid permite estructurar primero la disposición ("de afuera hacia adentro").

T.8. Responsive

8.1. ¿Qué es el diseño responsive?

● El diseño responsive hace que tu sitio web se vea bien en todos los dispositivos.
● 3 dispositivos: escritorio, tableta y móvil.

Características clave: Utiliza solo HTML y CSS, sin necesidad de programas o JavaScript.

8.2. Media Queries

● Las consultas de medios en CSS permiten crear diseños responsivos.

Características y tipos de medios comunes:

- all: Para todos los tipos de dispositivos.
- print: Para vista previa de impresión.
- screen: Para pantallas de computadoras, tabletas y smartphones.

Palabras clave:

- not: Invierte el resultado de la consulta.
- only: Previene que navegadores antiguos apliquen los estilos.
- and: Combina un tipo de medio y características.

● Cómo usar la regla @media de CSS.
● Puedes usar todas las consultas de medios en un mismo archivo: ejemplo.
● Para cada resolución, crearemos una consulta de medios, ejemplo:

```
@media screen and (max-width: 480px) {
  /* Para pantallas con un ancho igual o menor a 480px */
}
```

8.3. Configurando el Viewport

○ El viewport es el área visible en la ventana del navegador.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

○ **width="device-width"** → el ancho de la página sigue el ancho de la pantalla del dispositivo.

○ **initial-scale=1.0** establece el nivel de zoom inicial cuando la página se carga por primera vez en el navegador.

8.4. Principios del Diseño Web Responsivo

1. Responsivo vs. Adaptativo. En el diseño responsivo, los elementos fluyen, mientras que en el diseño adaptativo no tienen que hacerlo.
2. El flujo. La dimensión vertical debe tenerse en cuenta → los contenedores y los elementos deben fluir de manera ordenada y no deben superponerse entre sí.
3. Unidades relativas. Usa unidades relativas (em, %, rem, vw, vmin, vmax) siempre que sea posible.
4. Puntos de interrupción (Breakpoints). Usa Media Queries para colocar los contenedores según el diseño (escritorio, móvil, tableta).
5. Valores máximos y mínimos. Usa valores máximos y mínimos para las medidas.
6. Objetos anidados.Envolver elementos en un contenedor y crear diferentes niveles facilita la gestión de todo.
7. ¡Primero móvil!. Según algunos expertos, es mejor diseñar primero para dispositivos móviles y luego para escritorio.

8.4.1. Diseño Móvil Primero

● Comienza creando el diseño para dispositivos móviles antes de trabajar en el diseño clásico para escritorio.

● Pero... ¿Por qué?

○ Porque el uso de internet en dispositivos móviles superó al uso en escritorio desde 2016 (y sigue creciendo...).

○ El mercado de teléfonos inteligentes ha superado a las PCs

○ Porque, según personas que saben mucho más que yo y hablan más rápido, como KP, es más eficiente.

Cómo Hacer Imágenes Responsivas

Las **imágenes responsivas** se ajustan automáticamente para adaptarse al tamaño de la pantalla del dispositivo, mejorando la experiencia de usuario en diferentes tamaños de pantalla.

Pasos para crear imágenes responsivas:

```
<img src="nature.jpg" alt="Nature" class="responsive">
```

Para hacer que la imagen sea 100% del ancho disponible y mantener la proporción:

```
.responsive {

  width: 100%;

  height: auto;}
```

Para evitar que la imagen se amplíe más allá de su tamaño original:

```
.responsive {

  max-width: 100%;

  height: auto;}
```

Para limitar el tamaño máximo de la imagen a un valor específico (en píxeles):

```
.responsive {

  width: 100%;

  max-width: 400px; /* Ajusta el valor según el tamaño deseado */

  height: auto;}
```

Tutoriales

CSS Tutorial

CSS Exercises → **138 exercises of CSS3, 2 to 4 per chapter**

CSS Examples **—> Examples of CSS organized by categories**

W3Schools How TO - Code snippets for HTML, CSS and JavaScript →Plenty of snippets for HTML, CSS and JS

Design Sites Faster in Framer — Design & Layout Features

Learn CSS Display Property In 4 Minutes - YouTube

Learn CSS Position In 9 Minutes - YouTube

CSS selectors - CSS: Cascading Style Sheets | MDN

Learn Every CSS Selector In 20 Minutes - YouTube

CSS Selectors Reference

CSS Diner - Where we feast on CSS Selectors! **(A fun game to learn about combinators, pseudo-classes and pseudo-elements)**

Learn CSS Units In 8 Minutes - YouTube

CSS Units

CSS values and units - Learn web development | MDN

CSS Flexbox Layout Guide | CSS-Tricks

CSS Flexbox (Flexible Box)

Basic concepts of flexbox - CSS: Cascading Style Sheets | MDN

Learn Flexbox in 15 Minutes - YouTube

Flexbox CSS In 20 Minutes

Flexbox design patterns you can use in your projects

Learn CSS Grid in 20 Minutes - YouTube

CSS Grid Layout Crash Course

CSS Grid Layout

CSS Grid Layout Guide | CSS-Tricks

Grid by Example - Usage examples of CSS Grid Layout

CSS Grid Layout Module Level 2

Responsive Web Design | 10 Basics - YouTube