POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Educación para vivir mejor

Construcción de elementos de software web 1

Semana 4

Fundamentos de HTML - CSS - Modelo de Caja

Media Técnica en Programación de Sistemas de Información
Politécnico Jaime Isaza Cadavid - 2020

Modelo de Caja

En HTML cada elemento se encuentra representado visualmente como una caja, esto lo podemos evidenciar fácilmente añadiendo un borde a un elemento HTML y observando como el navegador lo pinta, por ejemplo:

```
<h1>Encabezado 1</h1>
```

Caja Básica

El modelo de caja es una especificación que nos define las características específicas de esa caja, y como infieren en el resto de elementos de la página. de esta forma le indica al navegador web como debe **pintar cada caja**, es decir, cada caja resulta siendo un elemento.

Es por ello que al entrar en profundidad con los modelos de caja debemos tener claro los conceptos básicos sobre aspectos de alto, ancho, borde, margen y relleno además de las formas básicas de visualización de los elementos HTML.

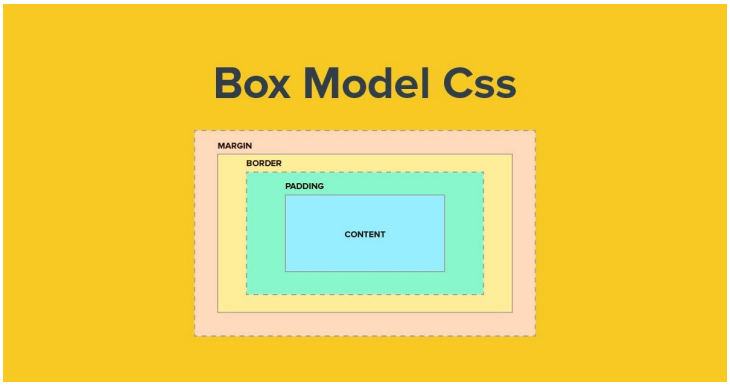
De esta forma el modelo de caja nos va a servir para:

- Comprender cómo se comportan los contenedores (cajas) cuando les cambiamos propiedades de apariencia como margen, borde, padding y ancho/alto. Y qué pasa cuando el contenido no cabe en nuestro contenedor.
- Entender cómo se aplican las imágenes de fondo.

Height, width, border, padding, margin

Cada elemento tiene una *altura* (height) y *anchura* (width). Además, puede tener otros atributos relacionados que influyen en su tamaño y su posición, que son el padding, los márgenes y los bordes:

- El ***border*** de un elemento es una línea que puede tener distinto grosor y que encuadra el contenido del elemento.
- El ***padding*** es la distancia desde el contenido del elemento hasta el borde.
- El ***margin*** es la distancia desde borde del elemento hasta los elementos que están a su alrededor.



Visualización (display)

La propiedad CSS display, se encarga de definir cómo se va a visualizar un elemento HTML, cómo va a colocarse en la página y cómo se colocarán el resto de elementos respecto a este. Según el valor que tenga asignado **display**, un elemento puede ocupar el ancho entero de su contenedor, ocupar solo el espacio que necesite para mostrar su contenido, mostrarse como si fuese una casilla de una tabla o directamente ocultarse.

Los navegadores web aplican por defecto un valor display a todos los elementos HTML de nuestra web. Hay muchos valores distintos para display pero, por el momento, nosotros solo veremos cuatro:

- block
- inline
- inline-block
- none

Block

Los elementos *en **bloque*** se muestran ocupando el ancho completo de su contenedor. En otras palabras, si tenemos un elemento en bloque dentro de una etiqueta aside, este ocupará el ancho completo del **aside**, si por lo contrario el elemento está directamente dentro del **body**, este ocupará el ancho del **body** y así siempre. Los elementos en bloque siempre empiezan en una nueva línea y nunca van a tener más elementos a su misma altura dentro del mismo contenedor, estarán más arriba o más abajo.

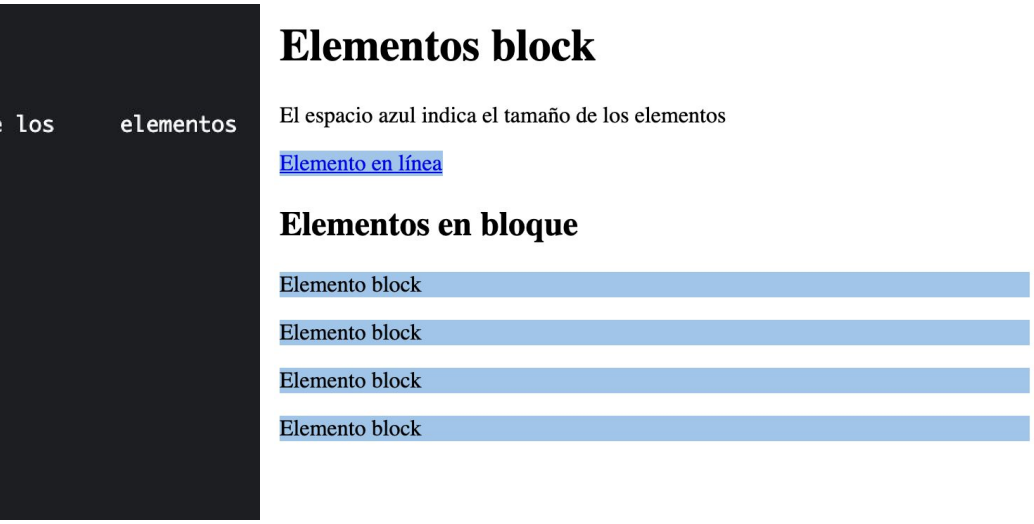
Como hemos comentado antes, los navegadores asignan automáticamente un modo de visualización por defecto a todos los elementos HTML. Algunos elementos como los párrafos **(p**), los divs o las listas (**ol y ul**) por defecto se muestran en bloque debido a esos estilos que aplica el navegador.

```
<h1>Elementos block</h1>

<p class="info">
  El espacio azul indica el tamaño de los     elementos
</p>

<a href="#">Elemento en línea</a>

<h2>Elementos en bloque</h2>
<div class="row">
  <p>Elemento block</p>
  <p>Elemento block</p>
  <p>Elemento block</p>
  <p>Elemento block</p>
</div>
```



Teniendo en cuenta la imagen anterior vemos representado cómo funcionan los elementos en bloque. En el resultado se ha puesto un fondo azul para que se vea el ancho total de cada elemento. Como se puede ver, en el caso del enlace, este ocupa el ancho de su contenido (del texto) mientras que los párrafos ocupan el ancho total del contenedor. El texto de los párrafos cabría perfectamente uno al lado del otro pero al ser bloques y ocupar ambos todo el ancho del contenedor, se muestran cada uno en una línea.

Usando CSS podemos hacer que un elemento que no se muestra en bloque cambie y se muestre de esta manera. Para ello aplicaremos **display: block** en el elemento. De esta forma, si queremos hacer que un elemento con la clase block se muestre en bloque, utilizaremos el siguiente código:

```
1  .block {
2    display: block;
3  }
```

Habíamos detallado con anterioridad cuáles etiquetas se muestran en bloque en nuestro navegador web. Como es difícil memorizar la lista completa de elementos que se muestran en bloque, podemos hacernos la siguiente pregunta para saber si un elemento se mostrará en bloque o no: ***"¿Tendría sentido meter este elemento dentro de un párrafo?"***. Si la respuesta es "no", es muy probable que el elemento sea un bloque; si por lo contrario, la respuesta es sí, probablemente sea un elemento en línea. Por ejemplo, no tendría sentido meter una **lista**, un **aside** o un **div** dentro de un párrafo y por eso los navegadores muestran estos como bloques.

Inline

Los elementos en línea o *inline* son aquellos que ocupan lo que ocupa su contenido. En estos, el tamaño será exactamente el tamaño de su contenido. Por ejemplo, si tenemos un enlace con el texto "pulsame", el ancho de ese enlace será el mismo ancho que el texto.

Debido a que los elementos en línea solo ocupan el ancho de su contenido, estos pueden colocarse uno al lado del otro hasta que no quede más espacio restante en la fila, en cuyo caso se colocarán en la fila siguiente. Cuando un elemento es muy largo y no cabe completamente en una línea, la parte que no cabe se baja a la línea siguiente. Por poner un ejemplo claro, un elemento en línea se comportaría como una letra más dentro de un texto, de hecho, estos también respetan los espacios entre ellos como se puede comprobar en el siguiente ejemplo.

Elementos inline

El espacio azul indica el tamaño de los elementos

Elementos en línea

Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline Elemento inline

Elementos en línea sin espacios entre ellos en el HTML

Elemento inlineElemento inlineElemento inlineElemento inlineElemento inlineElemento inlineElemento inlineElemento inlineElemento inline

Al ocupar solo el ancho de su contenido, los elementos en línea permiten colocar un elemento al lado del otro siempre y cuando ambos quepan en la misma fila, como sucede con los elementos en línea. Aquí también se respetarán los espacios entre los elementos en línea como si se tratasen de palabras normales y corrientes.

Como norma general, los elementos **inline** no deberían contener otros elementos de bloque.

Una regla muy importante que se aplica sobre los elementos en línea es que estos no pueden cambiar su ancho ni su alto, no pueden tener márgenes horizontales y se puede aplicar margen y padding vertical pero este no se tiene en cuenta a la hora de definir su altura y su posición vertical. Esto los diferencia de los elementos en bloque, que permiten tener un ancho y un alto específico y márgenes y padding tanto vertical como horizontal. Veremos la importancia de esto en esta misma sesión, cuando hablemos del modelo de cajas.

Importante recordar: Las imágenes son un tipo especial de elemento en línea que por sus características actúa como una mezcla de elemento en línea y elemento en bloque, ya que pueden tener márgenes y padding verticales y se les pueden asignar un ancho y un alto.

Usando CSS podemos cambiar la visualización de un elemento para hacer que se muestre en línea. Para ello aplicaremos display: inline en el elemento. Si quisiéramos hacer que un elemento con la clase inline se muestre en línea, utilizamos el siguiente código:

```
.inline {
  display: inline;
}
```

Inline-block

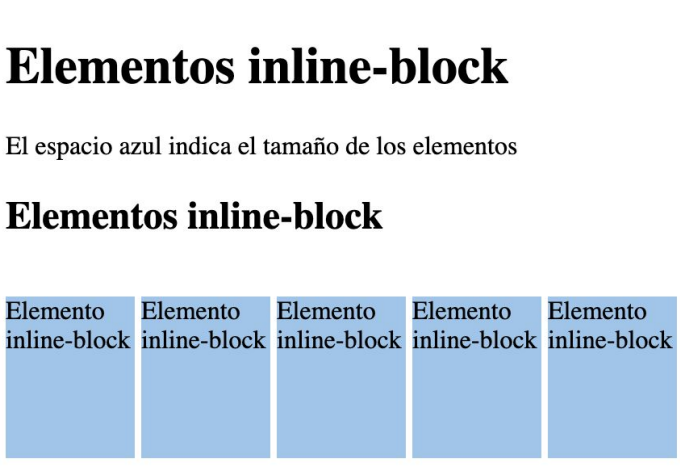
En este caso y como su nombre indica, el comportamiento de los elementos **inline-block** es una mezcla entre el comportamiento de los elementos en línea y los elementos en bloque.

Los elementos **inline-block** ocupan por defecto el ancho de su contenido y se comportan como si se tratase de una palabra más dentro de un texto, al igual que los elementos en línea, pero permiten tener un ancho, un alto, padding y márgenes verticales, como sucede con los elementos en bloque

```
<h1>Elementos inline-block</h1>

<p class="info">
  El espacio azul indica el tamaño de los     elementos
</p>

<h2>Elementos inline-block</h2>
<div class="row">
  <p class="inline-block">Elemento inline-block</p>
  <p class="inline-block">Elemento inline-block</p>
  <p class="inline-block">Elemento inline-block</p>
  <p class="inline-block">Elemento inline-block</p>
  <p class="inline-block">Elemento inline-block</p>
</div>
```



Elementos ocultos

A veces queremos que un elemento esté oculto, por ejemplo, el típico mensaje de aviso de cookies que aparece cada vez que entramos en una página. Con JavaScript, haremos que este mensaje se muestre o se oculte dependiendo si hemos visitado antes la página o no, pero desde JavaScript lo que haremos será añadir o quitar una clase CSS, los estilos los gestionaremos siempre desde el CSS.

Entonces, para poder ocultar un elemento (imaginemos que tiene una clase **hidden**) lo haremos desde el CSS.

```
.hidden {
    display: none;
}
```

Dimensiones y box-sizing

Una vez vistos los modos principales de visualización podemos entrar al modelo de caja. Recordemos que el modelo de caja es el que le dice al navegador cómo debe pintar cada caja.

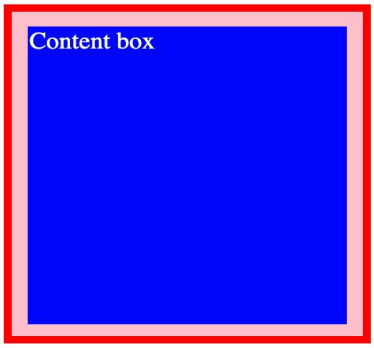
Si pensamos en el conjunto global, una página sería como un conjunto de cajas una dentro de otra, por lo tanto si pensamos en cada elemento a partir de ahora como un rectángulo nos será mucho más fácil visualizar cómo se compone la estructura de una web y cómo podemos pensar en ella combinando elementos que contienen otros elementos a su vez.

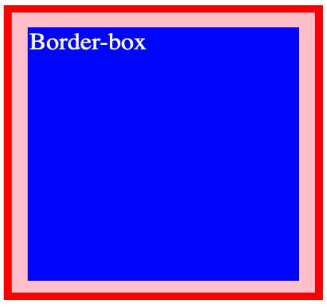
EJEMPLO:
Si tengo una caja de 100x100px, con un borde de 2px y con un padding de 16px, tendría una caja de 2+16+100+16+2: 136x136px.

Por defecto los elementos tienen el modelo de caja content-box. Con la propiedad CSS box-sizing podemos cambiarlo asignando el valor border-box, que es el otro modelo existente. En border-box tanto el borde como padding están incluidos en el ancho/alto del elemento, de manera que en el caso anterior nuestra caja tendría 100x100px pero el espacio para el contenido de nuestra caja no sería de 100x100 sino de 100-(2+2+16+16): 64x64px. Mira y entiende el siguiente ejemplo.

HTML

```
<div class="box content-box">
  <div class="content">Content box</div>
</div>
<div class="box border-box">
  <div class="content">Border-box</div>
</div>
```





CSS

```
body {
  margin: 0;
}
.box {
  margin: 10px;
  background: pink;
  width: 200px;
  height: 200px;
  border:5px solid red;
  padding: 10px;
}
.content {
  border: 1px solid blue;
  height: 100%;
  background: blue;
  color: white;
}
.content-box {
  box-sizing: content-box;
}
.border-box {
  box-sizing: border-box;
}
```

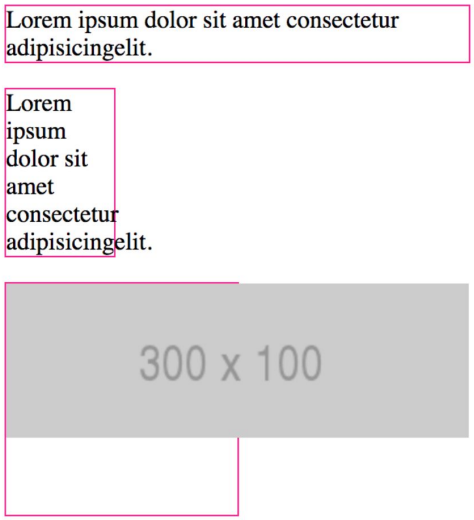
Si quisiéramos cambiar el modelo de caja para todos los elementos podemos usar el selector *, que modifica todos los elementos de la página, y por lo tanto debemos de usarlo con mucho tiento.

```
* {
  box-sizing: border-box
}
```

Overflow

Por defecto, nuestros contenedores tomarán el tamaño del contenido pero desde el momento en que definimos un tamaño para el contenedor puede pasar que el contenido no quepa ¿Y entonces, qué?

Pueden pasar dos cosas, que el contenido se pueda adaptar, como pasa con el texto o que el contenido simplemente se salga de nuestro contenedor (también puede pasar con el texto):



Overflow Básico

Podemos controlar cómo se comporta un contenedor en los casos en que el contenido se salga, tenemos 3 opciones:

1. No hacer nada y lidiar con las consecuencias
2. Ocultar todo lo que se salga
3. Incluir scroll en el contenedor

Alineando elementos en línea

text-align nos permite alinear horizontalmente texto y elementos en línea con los valores **right, left, center** y **justify**. De esta manera cuando lo aplicamos sobre una etiqueta podemos modificar la alineación horizontal de todas sus hijas cuyas display sea **inline.**

DevTools

Desde que aparecieron las *Devtools* en todos los navegadores decentes, la vida del front-end es mucho más tranquila. Estas herramientas nos permiten saber qué está pasando en un módulo concreto (medidas, posicionamiento, CSS aplicados) o qué está cargando nuestra web (hojas de estilos, imágenes, vídeos/audios, JavaScript).

DevTools: Inspector

El inspector es una de las muchas herramientas de desarrollo que incluye el navegador web Google Chrome. Este grupo de herramientas recibe el nombre de Chrome DevTools.

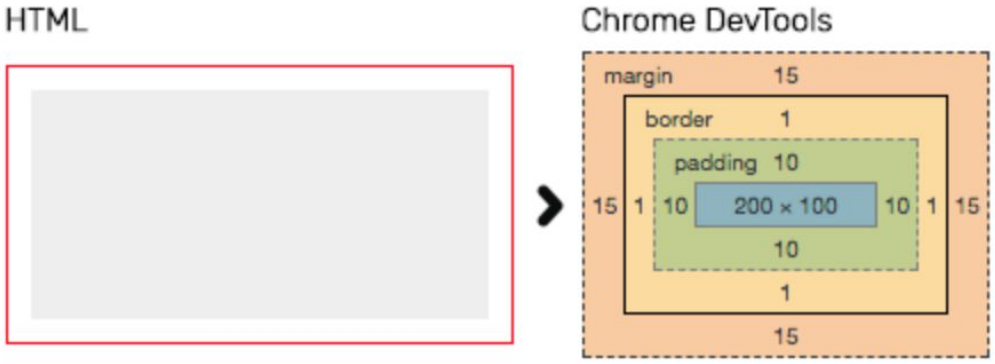
El inspector es una herramienta que viene con nuestro navegador y por tanto es parte de la aplicación del navegador, está incluida en, prácticamente, todos los navegadores más famosos (Chrome, Firefox, Safari, Internet Explorer, Edge...) y sirve para leer, añadir, editar o eliminar tanto CSS como HTML (y sus atributos) de nuestra página. Con él haremos de cirujanos de la web, veremos sus tripas y las modificaremos a nuestro antojo.

El inspector nos permite indagar y modificar tanto en páginas que tengamos en nuestro ordenador como otras que estén publicadas en Internet. Cuando modifiquemos estas páginas no estaremos modificando las páginas como tal realmente, solo temporalmente para ver qué sucedería si aplicamos ciertos cambios pero la web como tal, ya sea la de nuestro ordenador o la de Internet, no va a verse modificada. Esos cambios serán temporales y una vez que recarguemos la página se perderán y ésta volverá a su estado inicial.

Dado que nos permite controlar qué está pasando con una web, podemos ver los recursos que se están cargando y cuáles fallan. Nos permite ver el código tanto de nuestra página, para ver si está funcionando correctamente, como de otras, para ver cómo aplican ciertas técnicas o coger inspiración.

Por otro lado nos permite investigar qué cambios queremos hacer sin guarrear nuestro CSS o HTML y corregir de forma más rápida y sencilla los errores de nuestro código.

Por ejemplo, podemos ver información del modelo de caja:



Modelo de Caja

Colores

Para empezar, vamos a ver los distintos formatos que podemos usar para indicar colores, que podemos usar por ejemplo como valor de nuestro querida propiedad CSS color.

Colores con palabras clave

La primera forma de indicar un color es mediante la palabra clave que indica el nombre del color.

```
p {
  color: fuchsia;
}
```

Colores en hexadecimal

De forma equivalente a las palabras clave, podemos expresar un color con formato hexadecimal. En este formato declaramos un color con una almohadilla # y sus 3 componentes RGB - R (rojo), G (verde), B (azul). Cada uno de los componentes se representa con 2 dígitos en hexadecimal, es decir, cada dígito puede tener 16 valores, entre 0 - 9 y A - F. Por ejemplo, el color fucsia se compone de una componente máxima de rojo (ff), nada de verde (00) y máxima de azul (ff).

```
p {
  color: #ff00ff;
}
```

Suele ser habitual expresar algunos colores comunes de forma simplificada. Si los dígitos de cada componente son iguales (por ejemplo, ff) puede escribirse el color de una forma simplificada escribiendo sólo una vez el dígito repetido. Por ejemplo, el fucsia puede simplificarse porque todos los componentes tienen el dígito repetido.

```
p {
  color: #f0f;
}
```

rgb y rgba

Como hemos visto en el caso anterior, los colores podemos expresarlos con sus componentes RGB (Red, Green, Blue). En CSS existe la posibilidad de, en lugar de usar 2 dígitos hexadecimales, expresar el color usando el valor decimal (número normal) de cada componente RGB, que tendrá un valor entre 0 y 255 (los mismos valores que podíamos indicar con 2 dígitos hexadecimales).

```
p {
  color: rgb(255, 0, 255);
}
```

Existe además la posibilidad de indicar un nivel de opacidad al color con el formato RGBA que añade el canal **alpha** o transparencia. Este último componente tiene valores decimales entre 0 (totalmente transparente) y 1 (totalmente opaco).

```
p {
  color: rgba(255, 0, 255, 0.7);
}
```

hsl y hsla

Igual que el RGB nos permite expresar colores a partir de sus componentes de color rojo/verde/azul, existe otro sistema, HSL, que nos permite expresarlos a través de H (hue - matiz), S (saturation - saturación), L (lightness - luminosidad). El matiz se expresa con un valor numérico y tanto saturación como luminosidad con un valor en %. En este caso, también existe la posibilidad de añadir un canal alpha para indicar transparencia.

```
p {
  color: hsl(300, 100%, 50%)
}

p {
  color: hsl(300, 100%, 50%, 0.7)
}
```

Background

Una vez entendido que cada elemento se puede ver como una caja, veamos cómo añadir un fondo a dicha "caja":

Gracias a la propiedad *background* podemos rellenar el fondo de nuestro contenedor con una imagen, con un color, o ambos:

La propiedad background se construye con estos posibles valores:

- url de la imagen
- posición de la imagen dentro del contenedor (horizontal y vertical)
- modo de repetición de la imagen
- color de fondo

Realmente, la propiedad *background* es una versión acortada de estas propiedades:

- **background-image:**: Ver detalle
- **background-position:** Ver detalle
- **background-repeat:** Ver detalle
- **background-color:** Ver detalle

Cuando usar las propiedades de background o la versión acortada

Usar estas dos propiedades produce el mismo resultado pero no son lo mismo:

```
.box {
    background-color: green;
}
.box {
    background: green;
}
```

Mientras que en el primer caso estamos diciendo que el color de fondo sea green, en el segundo estamos diciendo eso y que el resto de valores pasen a su valor por defecto.

Cuando usamos un acortador estamos definiendo TODAS las opciones, aunque no las usemos, por eso siempre deberíamos usar las propiedades que necesitemos, y solo usar los acortadores cuando realmente estemos usando la mayoría de las propiedades que acortan.

Si solo queremos cambiar el color de fondo deberíamos usar background-color. Si por el contrario queremos poner una imagen, en una posición y con una repetición, deberíamos usar el acortador background ya que realmente se escribe menos:

```
.box {
    background-image: url('https://fillmurray.com/150/150');
    background-position: left top;
    background-repeat: no-repeat;
}
```

Background-size

Desde hace tiempo hay una propiedad nueva que nos permite redimensionar la imagen de fondo y hasta definir cómo se debe ajustar a nuestro contenedor: background-size.

Nosotros vamos a ver cómo ajustar el fondo a nuestro contenedor pero puedes consultar la documentación completa.

Hay dos valores especialmente interesantes ya que permiten definir cómo se ajustará nuestra imagen de fondo al contenedor: **contain** y **cover**.

Contain

Aumenta o reduce la imagen proporcionalmente todo lo que pueda sin deformarla para que quepa en nuestro contenedor.

Cover

Aumenta o reduce la imagen proporcionalmente para asegurarse que siempre cubre todo el área de nuestro contenedor, aunque eso signifique que parte de la imagen pueda quedar oculta.

Usando fuentes de Google Fonts

Para utilizar fuentes tipográficas de un sitio externo como Google Fonts, tenemos que seguir 2 sencillos pasos: 1) Añadir una etiqueta link a nuestro body con un enlace que cargue la fuente

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Font+Name">

```
p{
  font-family: 'Font Name', serif;
}
```

Enlazando la fuente

En la propia URL añadimos family= y escribimos el nombre de la tipografía a usar. Si tiene espacios, los sustituimos por +. Si queremos importar varias fuentes, podemos cargar todas en el mismo enlace a Google Fonts poniendo los nombres separados por |.

https://fonts.googleapis.com/css?family=Tangerine|Inconsolata|Droid+Sans

Para cada tipo de fuente se importa por defecto la fuente normal, pero puede que queramos usar la fuente con otro peso (como negrita) o estilo (como cursiva). Para esto, añadimos al final del nombre de la fuente : y separados por , los estilos o pesos extra que necesitemos. El peso puede expresarse también como valor numérico que indica el grosor (400 es normal, 700 es negrita).

https://fonts.googleapis.com/css?family=Tangerine:bold

https://fonts.googleapis.com/css?family=Tangerine:bold,italic

https://fonts.googleapis.com/css?family=Tangerine:400,700

Extensión Live server en Visual Studio Code

Cuando creamos una web, al final será **"servida"** por un servidor web en Internet para que todo el mundo pueda visitarla. Por tanto, es útil tener un servidor web local en nuestro ordenador de desarrollo. En Visual Studio Code, contamos con la extensión live-server, que nos permite lanzar un servidor web local desde una carpeta de nuestro ordenador y ejecutándose en un puerto concreto.

Por defecto, el servidor web se lanza con origen en la carpeta raíz que tengamos en nuestro proyecto actual. Como cualquier servidor web, busca en la raíz un fichero index.html como punto de entrada a la web. Si no lo encuentra, un servidor web real daría un error, pero live-server muestra una página que nos permite navegar por las subcarpetas de nuestro ordenador hasta llegar al html que queramos mostrar.

Cada vez que modifiquemos los ficheros usados en la web que estamos visualizando en el navegador con Live Server, ésta se recargará **automágicamente** en el navegador.

Taller: Ejercicios Práctico

1. Usar la etiqueta **<mark>** dentro de varios párrafos y explicar para qué sirve y cómo funciona.

2. **Displays:** Prepara tres divs con un tamaño de 100x100, cambia sus displays (block, inline, inline-block, none), observa y explica cómo se comportan.

3. **Imagen en párrafo:** Dentro de un párrafo de texto incluir una imagen de 100x100 y explicar cómo se distribuye el contenido.

4. **Imagen entre párrafos:** Entre dos párrafos añadir una imagen de 200x200 y explicar cómo se distribuye el contenido.

5. **Ajustando imágenes:** Hacer un div de 100x100px usando las propiedades width y height, incluir dentro una imagen de 100x100px y probar:

    a. Añadir un padding de 10px.
    b. Añadir un borde de 5px.
    c. Cambiar el modelo de caja a border-box y explica qué ha pasado.
    d. Centrar la caja utilizando el valor auto en los márgenes horizontales.

6. Entra a https://www.wikipedia.org/ y:

    a. Cambiar el color de los enlaces a naranja.
    b. Sobre los idiomas destacados que aparecen sobre la imagen de la pelota de Wikipedia, añadir uno falso.
    c. Explicar cómo están compuestos estos módulos de idioma.
    d. Explicar cómo están colocados.
    e. Examinar la versión de tablet de Wikipedia.
    f. Examinar la versión de móvil de Wikipedia.
    g. Averiguar las dimensiones de la caja de búsqueda y
        i. Cuánto tiene de separación con el botón de buscar
        ii. ¿Qué hay de raro con esa separación?

7. ¿Sabríamos ir a la web https://duckduckgo.com, buscar el logo con el id "logo_homepage_link" y aplicar estos estilos desde nuestro inspector? Evidenciar con una captura de pantalla que el cambio se realice.

```
background-color: black;
border-radius: 10px;
width: 250px;
```

Quiero saber más..

- https://uniwebsidad.com/libros/css/capitulo-4?from=librosweb
- https://uniwebsidad.com/libros/css/capitulo-4/anchura-y-altura?from=librosweb
- https://developers.google.com/web/tools/chrome-devtools/?hl=es
- https://developers.google.com/web/tools/chrome-devtools/shortcuts?hl=es
- https://es.khanacademy.org/computing/computer-programming/html-css/web-development-tools/a/using-the-browser-developer-tools
- https://devdocs.io/css/background-size
- https://developers.google.com/fonts/docs/getting_started