**HTML**

Lenguaje de Marcado de Hipertexto (HTML) es el código que se utiliza para estructurar y desplegar una página web y sus contenidos. Este código se maneja con etiquetas, por lo cual no se considera lenguaje de programación.

Para esto creamos un archivo con la extensión .html, esta es la forma en la cual el servidor que interpreta la pagina pueda mostrar el contenido.

En el puedo añadir texto, imágenes, videos, CSS, JavaScript, etc.

Estas paginas pueden ser estáticas (solo código) o dinámicas (con contenido extraído de algún medio externo, como puede ser una base de datos).

Como funciona?

1- El navegador (cliente) pide la información al servidor.

2- El servidor devuelve la información al cliente en un archivo HTML.

3- El navegador en el cliente lee el archivo de arriba hacia abajo y de izquierda a derecha para interpretar su información.

4- Tiene en cuenta las etiquetas que tiene el documento y las va renderizando en la pantalla (lo que se ve en el navegador).

**Sitio Web**

Son un conjunto de paginas web estructuradas en un dominio. Se pueden crear escribiendo código o mediante CMS o builders. Pertenecen a empresas, organizaciones o personas.

**Front-End y Back-End**

El Front-End es la parte visible por el usuario de una pagina, esta relacionada a la creación de una pagina funcional que puede verse en un navegador. Esto se construye con HTML, CSS Y JS.

El Back-End es la que se encarga del manejo de algoritmos y la utilización de bases de datos para guardar o procesar información, incluyendo también el manejo de servidores. Esta se construye con la mencionada base de datos, python y frameworks.

**FullStack**

Es el desarrollador que se encarga tanto del Front-End como del Back-End.

**Trabajando con HTML.**

Para trabajar con este formato utilizaremos el editor de texto Visual Basic Studio, aunque se puede usar Notepad++ u otros.

En el, abriremos un archivo con extension html, ej: index.htmlm, que es donde trabajaremos.

Una vez en el editor con el archivo creado, escribiremos ! O html:5 que nos brindara la estructura principal de una pagina web. Nos quedara lo siguiente:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <link rel="styles" type = "text" href=".css\styles.css">

    <title>Pagina Web</title>

</head>

<body>

</body>

</html>

<html lang="en"> Expresa el lenguaje en el que sera escrita la pagina.

<head> … </head> Es donde se contiene la información de la pagina.

<meta charset="UTF-8"> Es donde se especifica el tipo de caracteres, en este caso UTF-8

<meta http-equiv="X-UA-Compatible" content="IE=edge"> Donde se pone el formato y el navegador predeterminado, que al ser una herramienta de microsoft VisualCodeStudio aparece Edge.

<title></title> Contiene el titulo de la pagina.

<body>...</body> Etiqueta donde se desarrollara todo el cuerpo de la pagina, tanto los encabezados como los pie de paginas, diseños, colores, fuentes, incluso la misma arquitectura de la pagina.

Notamos que para cerrar las etiquetas va una barra previa a su nombre:

<title> <**/**title>

**Etiquetas:**

**Atributos:** Es la modificacion que se le puede aplicar a una etiqueta. Por ejemplo

<h1 type=”’x’’’>

<!-- … -- > donde se podra comentar el codigo escrito.

<hn> </hn> donde n refiere al tipo de preferencia que tiene la etiqueta, se usa para agregar titulos o subtitulos, si usamos h1, tendremos un tamaño mayor que en h2, h3,…, hn

<p> </p> donde podremos añadir párrafos de texto en la pagina.

<hr> representa el cambio de tema de parrafo.

Si utilizamos \*n ,donde n es un numero elegido por el programador, sobre una etiqueta, podemos multiplicar una etiqueta, ej: h3\* 3 = <h3> </h3> <h3> </h3> <h3> </h3>

<br> Es un salto de linea

<small> </small> para texto pequeño.

<u></u> texto subrayado.

<i> </i> para texto en cursiva o itálica.

<del> </del> para texto tachado.

<sup> </sup> para texto como superindice (en formato de potencia para una referencia matemática)

<sub> </sub> para texto como subíndice (en formato de logaritmo para una referencia matemática)

<em> </em> para texto con énfasis.

<strong> </strong> para texto en negrita.

Listas:

<li></li> donde puedo agregar datos en forma de lista, uno por cada etiqueta <li></li>

<ol> </ol> donde puedo agregar las listas dentro de la etiqueta para que esten de forma ordenada

Para modificar el tipo de indice de los elementos podemos aplicar atributos como I, A a, etc. Ej:

<ol type =”I” > aplicara indice en numeros romanos.

<ul> </ul> donde puedo agregar las listas dentro de la etiqueta para tener una lista desordenada

Para crear una lista ordenada/desordenada dependiente, debemos crear una lista dentro de otra, es decir:

<ol> dentro va<li></li> dentro va <ol> <li> </li> cierra </ol>

<dl> </dl> para crear una lista de definicion, donde dentro habra otra lista donde estaran las palabras con su definicion, es decir:

<dl>

definition title o titulo de definicion

<dt>

definicion

<dd> donde va el texto</dd>

</dt>

</dl>

<table></table> etiqueta de tablas donde encontraremos en ella:

<tr></tr> etiqueta que agrega una linea en la tabla, es decir, una fila

<td> </td> para agregar casilleros de una tabla, se empiezan por las filas y luego las columnas

<td rowspan/colspan = x> donde rowspan indica la cantidad de columnas que se quiere que ocupe un dato y colspan la cantidad de filas.

**Enlaces:**

<a href = “<https://www.google.com/>”> google</a> para indicar texto que sera un vinculo a otro sitio.

<a href = “<https://www.google.com/>” target =”\_blank”> ir a google </a> donde target abre una pestaña para la referencia que indica la etiqueta.

<a href=”img/random.png””>abrirImagen</a> donde se vincula una imagen

<a href = “[random@gmail.com](mailto:random@gmail.com)?subject=”Contacto desde web””>Enviar Correo</a> donde se vincula un correo a un hipervinculo, subject indica el asunto del correo automaticamente.

<a href =”archivos/archivo.pdf” download =””> Descargar </a> donde download indica que el archivo puede ser descargado.

**Contenido dentro de pagina:**

<img src = “imagenes/imagen.png” alt = “imagen”> donde src trae la imagen desde su ubicacion (preferentemente en el mismo grupo de carpetas que el archivo donde se trabaja) y donde alt imprime un texto alternativo en caso de que la imagen no se muestre. Tambien funciona para las personas que tienen problemas de vision y la pagina permite leer el texto que se le esta mostrando a la persona.

<audio > </audio> etiqueta que contendra la fuente de audio local y su tipo, dentro encontramos:

<source src = “audio.mp3” type= “audio/mp3”>

Para traer audios de una fuente externa como Spotify, debemos ir a la playlist o a la cancion que deseamos, click en compartir y en insertar cancion o lista, esto nos dara un codigo con etiqueta iframe que es la que se encarga de traer contenido de fuentes externas a nuestro computador. Ej:

<iframe style="border-radius:12px" src="https://open.spotify.com/embed/playlist/37i9dQZF1DWWEJlAGA9gs0?utm\_source=generator" width="100%" height="352" frameBorder="0" allowfullscreen="" allow="autoplay; clipboard-write; encrypted-media; fullscreen; picture-in-picture" loading="lazy"></iframe>

El mismo caso seria para traer videos de youtube o fuentes que permitan compartir un iframe de lo que estamos viendo, en las opciones de compartir, indicaremos que queremos insertar codigo, lo que nos copiara el iframe de una playlist, video, cancion, audio, ubicacion de maps, etc..

**Etiquetas semanticas:**

Nos sirven para identificar las partes principales que tiene una web, son la forma de poder ordenar un sitio en funcion de su estructura.

Entre ellas encontramos:

<header></header> etiqueta que define el encabezado de una pagina

<nav> </nav> etiqueta que define el texto con formato de hipervinculo hacia otras paginas o archivos.

<section> </section> etiqueta que define los sectores de una pagina y a donde nos podemos direccionar con los vinculos ubicados en nav. Por ejemplo

Si en nav tenemos una etiqueta de este estilo

<nav>

<a href=”#Ejemplo”>ejemplo</a>

(El # , numeral , sirve para crear un ancla con el id de un elemento)

</nav>

con section podemos indicar hacia donde apunta la referencia

<section id=”Ejemplo”>Texto de ejemplo</section>

<article> </article> etiqueta que define el cuerpo principal de un escrito

<aside> </aside> etiqueta que define el contenido no relacionado al cuerpo principal, utilizada mayormente para anuncios.

<footer> </footer> etiqueta que define el pie de pagina, lo que se vera debajo de todo.

**Formularios:**

<form action =””></form> etiqueta que contendra los campos del formulario.

<fieldset></fieldset> etiqueta que servira de marco para el formulario

<legend> </legend> para agregar titulo al fieldset.

<label> </label> etiqueta que declara el campo del formulario, en ellas iran los atributos

<label for=”nombre”> Nombre</label> donde en for se indica el campo de referencia.

<input type=”text” name=”nombre” id=”nombre” placeholder=”Escribi tu nombre”> etiqueta que crea el campo donde el usuario ingresara los datos, en type se indica el tipo de dato que ingresara, en name e id se indicara el campo al que se hara referencia, placeholder es un mensaje dentro del campo de ingreso de datos que le indica al usuario que hacer, sera un texto gris a modo de indicacion.

<select name=”” id=””> </select> etiqueta que nos deja elegir dentro de un marco de opciones un input.

**CSS**

CSS (Cascading StyleSheet) es un lenguaje de diseño que nos permite darle estilos a los componentes de un documento en función de su jerarquía. Se ocupa de la estética, el aspecto.

Cuando diseñamos un sitio web profesional, con un equipo de trabajo, mantener los estilos separados de la estructura y contenido (HTML) facilita la división de tareas entre los desarrolladores.

Tenemos 3 tipos de CSS para aplicar estilo:

**CSS en linea**: Dentro del atributo style=”” incorporamos los estilos que se van a aplicar solo en esa misma etiqueta. Opcion no recomendable.

<p style=”color: white ; background-color: violet”> Esto es un parrafo</>

Utilizando el atributo style dentro de una etiqueta le proporcionamos estilo al parrafo. Se pueden utilizar a la vez varias parejas de: propiedad:valor.

**CSS Interno:** Incluimos la etiqueta <style> dentro del <head> en nuestro documento. Opcion menos recomendable:

<style>

h1{

color:black;

background-color: red;

}

</style>

En este caso todas las etiquetas h1 tendran color de fuente negro y color de fondo rojo.

**CSS Externo:** En el head del documento HTML tenemos que incluir una referencia al archivo css dentro del elemento <link>. Es la forma mas recomendada:

<link rel=”stylesheet” href=”estilos.css”>

La referencia al archivo externo debe incluir la ruta completa, el nombre del archivo y su extension si se encuentra en alguna subcarpeta dentro del proyecto.

En caso de que el archivo de estilos se encuentre en la misma carpeta que el documento html, unicamente se debe incluir el nombre del archivo y su extensión. Recordemos que es aconsejable mantener estos archivos (CSS Y HTML) en distintas carpetas.

**Dentro del CSS Externo:**

Encontramos dos factores fundamentales a la hora de trabajar con CSS, sus selectores y los atributos que se le agregan a cada selector.

**Selectores:**

Dentro de los selectores tenemos tanto el universal (\*) que aplica el estilo escrito a toda la pagina, sin que ningun elemento tenga algún detalle particular.

\*{

margin: 5px;

paddin:10px;

}

Obvio que si se quiere trabajar sobre alguna etiqueta semantica o sobre algun campo en particular, podemos cambiar el selector al nombre del cual refiere el elemento a trabajar.

h1{

color:red;

background-color:blue;

}

A tener en cuenta que, el CSS que este en linea, tiene mucha mas prioridad que el CSS externo, debido a que es mucho mas especifico en cuanto al estilo, es decir, que siempre que haya estilo en linea, sera prioridad para su interprete, en vez de tomar el estilo dado en CSS.

Puede suceder que el interprete tome atributos de CSS externo e interno, es decir, puede tomar el color de margen de un texto desde el CSS externo, pero puede tomar el color de la letra del CSS en linea, por lo que se recomienda trabajar los estilos por un lado y las estructuras por el otro.

**Selectores de Clase:** Son aquellos que nos permiten modificar el estilo de los elementos que tengan el atributo class definido en su declaracion. Se puede reutilizar.

En html

<h2 class=”subtitulos”>

En css:

.subtitulos{

color:red;

}

**Selector de Id:** Es similar al selector de clase, solo que se utiliza una sola vez.

En css:

#texto{

color:red;

margin:5px;

}

En html:

<p id=”texto”> hola </p>

**Span y Div:**

Ambas etiquetas son contenedoras de informacion, dentro de las etiquetas se ponen los elementos para que queden prolijos y ordenados.

Span es un elemento en linea, quiere decir que todos los elementos que ponga o las etiquetas que ponga dentro de span, estaran en la misma linea

Div es un elemento en bloque, quiere decir que todos los elementos que ponga dentro de la etiqueta estaran uno debajo del otro.

**Selectores avanzados:**

Utilizan combinadores, signos graficos que establecen la relacion entre los elementos y permiten hacer una selección especifica:

Agrupados: se especifican colocando el nombre seguido de una coma, ej: p , a , div.  
Se utiliza cuando varios elementos comparten una serie de declaraciones iguales, se aplican las reglas CSS a los sectores involucrados.

Descendientes: se especifican colocando un espacio entre los elementos, ej: div p. Apunta a los elementos contenidos dentro de otro en la estructura del documento, sin improtar la profundidad o los descendientes interpuestos entre ellos.  
  
Hijos directos: se especifican colocando un signo de > (mayor que) entre elementos, ej: span > a. Selecciona los elementos que sean de hijos directos del contenedor padre, descartando nietos y sucesivos.

Hermano adyacente: se especifican colocando un signo + (mas) entre los elementos, ej: div + p . Aplica estilos a los elementos que siguen inmediatamente a otros. Deben tener el mismo elemento padre para ser inmediatamente el siguiente a el.

General de hermanos: se especifican colocando el signo ~ (virgulilla), ej: div ~ p. Selecciona todos los elementos que son hermanos del especificado, sin necesidad de que sean adyacentes.

**Pseudoclases:**

Una pseudoclase es un selector que marca los elementos que están en un

estado específico o tienen un comportamiento determinado. Todas las

pseudoclases se denominan mediante una palabra precedida por dos puntos y

se comportan del mismo modo. Afectan a un fragmento del documento que

está en un estado determinado y se comportan como si se hubiera añadido una

clase a su HTML.

:first-child : esta pseudoclase modifica el estilo del primer elemento de un grupo de elementos hermanos dentro de un contenedor, es decir, “el primer hijo de su padre”.  
En html:  
div>

<p> Párrafo 1 </p>

<p> Párrafo 2 </p>

<p> Párrafo 3 </p>

</div>

En css:

p:first-child {

color: red;

}

Esto modificaria solo el texto del primer parrafo.

:last-child: se utiliza para representar al ultimo elemento de un grupo de elementos hermanos dentro de un contenedor, es decir, “el ultimo hijo de su padre”.

En html:

div>

<p> Párrafo 1 </p>

<p> Párrafo 2 </p>

<p> Párrafo 3 </p>

</div>

En css:

p:last -child {

color: red;

}

Esto modificaria solo al ultimo texto del parrafo.

:nth-child: el selector coincide con cada elemento que es el n-esimo hijo, independientemente del tipo de su padre. N puede ser un numero, una palabra clave o una formula.

En html:

<div>

<p> Párrafo 1 </p>

<p> Párrafo 2 </p>

<p> Párrafo 3 </p>

</div>

En css:

p:nth-child(2) {

background: cyan;

}

Esto cambia solamente el contenido del parrafo 2, aunque podria ser cualquiera que se especifique.

Si colocamos 3n o 2n, afectara solo a los que sean multiplo de 3 o 2, si tenemos 4 parrafos y aplicamos 2n, el contenido sera accionado sobre los parrafos 2 y 4. Por eso se dice que puede ser palabras o formulas.

**Pseudoclases para hipervinculos:**

Se aplican a las etiquetas <a>, que pueden tener cuatro estados:

:link se refiere a un enlace que todavia no ha sido visitado.

:hover se refiere a un elemento sobre el que se coloca el puntero del mouse

:visited se refiere a un enlace que ya ha sido visitado.

:active se refiere a cualquier elemento que ha sido activado por el usuario.

**Pseudoelementos:**

Los pseudoelementos se añaden a los selectores, pero no describen un estado especial, sino que permiten añadir estilos a una parte concreta del documento.

Se utilizan para darle estilos a partes especificas de un elemento. Estan precedidas por cuatro puntos (::):

::first-letter se utiliza para darle estilo a la primera letra de un texto.

::first-line se utiliza para darle estilo a la primera linea de un parrafo.

::selection agrega estilos a una parte del documento que ha sido resaltada por el usuario.

::before y ::after agregan contenido antes y despues, despectivamente, del contenido.

**Transformaciones, animaciones y transiciones.**

Transformaciones: permiten mover, rotar, escalar, y sesgar elementos, es decir, efectos visuales 2d y 3d. Las propiedades principales para realizar transformaciones son las siguientes:

transform :(formato) -> funcion1, funcion2: aplica una o varias funciones de transformacion sobre un elemento.

transform-origin: (formato) → posx, posy, posz: Cambia el punto de origen de un elemento en una transformacion.

transform-stile:(formato)→ flat| preserve-3d: modifica el tratamiento de los elementos hijos.

Con la propiedad transform podemos indicar una o varias transformaciones para realizar sobre un elemento, ya sean 2d ( sobre dos ejes) o 3d (sobre tres ejes).

2d – Traslaciones (translate) :

Las funciones de translacion son aquellas que realizan una transformacion en la que mueven un elemento de un lugar a otro. Si especificamos un valor positivo en el eje x (horizontal), lo moveremos hacia la derecha, y si especificamos valor negativo, lo moveremos hacia la izquierda. Lo mismo con el eje y (vertical).

translateX(x) translada el elemento una distancia de size x horizontalmente.

translateY(y) translada un elemento de size y verticalmente

translate(x,y) atajo de las dos anteriores.

2d-Escalado (scale):

Las funciones de escalado realizan una transformacion en la que aumentan o reducen el tamaño de un elemento, basandose en el parametro indicado, que no es mas que un factos de escala:

scaleX(fc) reescala un elemento a un nuevo tamaño con un factor number fx horizontal.

scaleY(fy) reescala el elemento a un nuevo tamaño con un factos number fy vertical.

scale(fx,fy) atajo de las dos anteriores.

2d-Rotaciones (rotate):

Las funciones de rotacion simplemente giran el elemento en un numero de grados indicado:

rotateX(xdeg) establece una rotacion 2d en direction xdeg grados solo para el eje horizontal x.

rotateY(ydeg) establece una rotacion 2d en direction ydeg grados solo para el eje vertical y.

rotate(deg) establece una rotacion 2d en direction deg grados sobre si mismo.

Funciones 3D:

A las funciones anteriores, tambien podemos añadir las funciones equivalentes de css para hacer uso del eje Z, y con esto acceder a las dimensiones espaciales o 3D. Las propiedades de transformacion son:

translateZ(z)

translate3d(x,y,x)

scaleZ(fz)

scale3d(fx,fy,fz)

rotateZ(zdeg)

rotate3d(x,y,x,deg)

perspective(n): establece una perspectiva 3d de size n.

matrix3d(n,n,…): establece una matriz de transformacion (16 valores).

Transformaciones múltiples

Si indicamos dos propiedades transform en el mismo elemento, con diferentes funciones, la segunda propiedad sobreescribirá a la anterior, como ocurre con cualquier propiedad de CSS:

div {

transform: rotate(5deg);

transform: scale(2,2); /\* Esta línea sobreescribe a la anterior \*/

}

Para evitar esto se suelen emplear múltiples transformaciones, separándolas mediante un espacio. En el siguiente ejemplo, aplicamos una función de rotación, una función de escalado y una función de traslación de forma simultánea:

div {

transform: rotate(5deg) scale(2,2) translate(20px, 40px);

}

Transiciones

Las transiciones CSS le permiten cambiar los valores de una propiedad, durante un período determinado. Se basan en un principio muy básico: conseguir un cambio de estilo con un efecto suavizado entre un estado inicial y un estado final. Para crear un efecto de transición, debemos especificar dos cosas:

● La propiedad CSS a la que desea agregar un efecto (¿qué propiedad modifico?)

● La duración del efecto (¿durante cuánto tiempo?)

Las propiedades relacionadas que existen son las siguientes:Transiciones

● transition-property: Indica la propiedad que será afectada por la transición. Puede ser una propiedad concreta (width o color, por ejemplo) o simplemente all para que se aplique a todos los elementos. Por otro lado, none hace que no se aplique ninguna transición.

● transition-duration: Establece la duración de la transición. Se recomienda comenzar con valores cortos, para que las transiciones sean rápidas y elegantes. Una duración demasiado grande producirá una transición con detenciones intermitentes, y pueden resultar molestas a muchos usuarios.

● transition-timing-function: indica el ritmo de la transición que queremos conseguir. Se recomienda comenzar con linear (ritmo constante) y luego utilizar otros valores para variar el ritmo sea al inicio y/o al final de la transición.

Los valores que puede tomar la propiedad son los siguientes:

Un valor linear siempre es constante, mientras que ease comienza suavemente, aumenta la velocidad y finaliza suavemente. Ease-in y ease-out son variaciones que van más lento al principio o al final, y ease-in-out una mezcla de ambas. La función Cubic-Bezier() nos permite configurar con más detalle la transición.

La función de tiempo Cubic-Bezier()

Es una función personalizada. Podemos asignar valores que definen la velocidad que queramos que tenga la transición. En la última columna de la tabla anterior podemos ver los valores equivalentes a cada una de las palabras clave mencionadas. En principio, el formato de la función es cubic-bezier(A, B, C, D), donde:

A (x1) eje x del primer punto que orienta la curva Bezier

B (y1) eje y del primer punto que orienta la curva Bezier

C (x2) eje x del segundo punto que orienta la curva Bezier

D (y2) eje y del segundo punto que orienta la curva Bezier

transition-delay nos ofrece la posibilidad de retrasar el inicio de la transición los segundos especificados.

Atajo: Transiciones

Como siempre, podemos resumir todas estas operaciones en una propiedad de atajo denominada transition. Los valores del ejemplo superior, se podrían escribir como se puede ver a continuación (si no necesitas algún valor, se puede omitir):

div {

/\* transition: <property> <duration> <timing-function> <delay> \*/

transition: all 0.2s ease-in;

}

Animaciones

Una animación permite que un elemento cambie gradualmente de un estilo a otro. Podemos cambiar tantas propiedades CSS como deseemos, tantas veces como sea necesario.

Las animaciones amplían el concepto de transiciones convirtiéndolo en algo mucho más flexible y potente, partiendo del mismo concepto de realizar cambios en ciertos estados inicial y final pero incorporando más estados, pudiendo realizar cambios desde un estado inicial, a un estado posterior, a otro estado posterior, y así sucesivamente. Además, esto será posible de forma automática, sin que el usuario tenga que realizar una acción concreta.

Para utilizar la animación CSS, primero debemos especificar algunos fotogramas clave (@keyframes) para la animación, que contendrán los estilos que tendrá el elemento en determinados momentos. Además tendremos que utilizar las propiedades de las animaciones, que definen el comportamiento de la misma.

Propiedades de animación CSS

Para definir este comportamiento necesitamos conocer las siguientes propiedades CSS:

animation-name permite especificar el nombre del fotograma a utilizar.

animationduration, animation-timing-function y animation-delay funcionan exactamente

igual que en transiciones.

La propiedad animation-iteration-count permite indicar el número de veces que se repite la animación, pudiendo establecer un número concreto de repeticiones o indicando infinite para que se repita continuamente. Por otra parte, especificando un valor en animation-direction conseguiremos indicar el orden en el que se reproducen los fotogramas, pudiendo escoger un valor de los siguientes:

normal: los fotogramas se reproducen desde el principio hasta el final.

reverse: los fotogramas se reproducen desde el final hasta el principio.

alternate: en iteraciones par, de forma normal, impares a la reversa.

alternate-reverse: en iteraciones impares, normal, pares, normal.

Por defecto, cuando se termina una animación que se ha indicado que se reproduzca sólo una vez, la animación vuelve a su estado inicial (primer fotograma). Mediante la propiedad animation-fill-mode podemos indicar que debe mostrar la animación cuando ha finalizado y ya no se está reproduciendo; si mostrar el estado inicial (backwards), el estado final (forwards) o una combinación de ambas (both).

La propiedad animation-play-state nos permite establecer la animación a estado de reproducción (running) o pausarla (paused).

Atajo: Animaciones

Nuevamente, CSS ofrece la posibilidad de resumir todas estas propiedades en una sola, para hacer nuestras hojas de estilos más específicas. El orden de la propiedad de atajo sería el siguiente: Debemos ser cuidadosos al indicar el tiempo en las propiedades de duración. Al ser una unidad diferente a las que solemos manejar (px, em, etc...) hay que especificar siempre la s, por segundos, aunque sea un valor igual a 0.

div {

/\* animation: <name> <duration> <timing-function> <delay>

<iteration-count> <direction> <fill-mode> <play-state> \*/

animation: changeColor 5s linear 0.5s 4 normal forwards running;

}

Fotogramas (keyframes)

Para definir los fotogramas de una animación utilizaremos la regla @keyframes. Primero elegimos un nombre para la animación (ya que podemos tener varias en una misma página), mientras que podremos utilizar varios selectores para definir el transcurso de los fotogramas en la animación.

div {

width: 100px;

height: 100px;

background-color: blue;

animation-name: cambiarColor;

animation-duration: 2s;

animation-delay: 1s;

/\* animation: cambiarColor 2s 1s; \*/ }

@keyframes cambiarColor {

from {background-color: red;}

to {background-color: yellow;}

}

En este ejemplo partimos de un primer fotograma con un elemento con color de fondo rojo. Si observamos el último fotograma, indicamos finalice con color de fondo verde. La regla @keyframes creará la animación intermedia para conseguir que el elemento cambie de color.

Los selectores from y to son realmente sinónimos de 0% y 100%. Al modificarlos

podremos ir añadiendo nuevos fotogramas intermedios:

div {

width: 100px;

height: 100px;

background-color: blue;

animation-name: cambiarColor;

animation-duration: 2s;

animation-timing-function: ease;

animation-iteration-count: infinite;

}

@keyframes cambiarColor {

0% {background: red; width: 200px;}

/\* Primer fotograma \*/

50% {background: yellow; width: 400px;}

/\* Segundo fotograma \*/

100% {background: green; width: 600px;}

/\* Último fotograma \*/

}

Encadenar animaciones

Es posible encadenar múltiples animaciones, separando con comas las animaciones individuales y estableciendo un tiempo de retardo a cada animación posterior:

.animated {

animation:

moveRight 5s linear 0, /\* Comienza a los 0s \*/

lookUp 2.5s linear 5s, /\* Comienza a los 5s \*/

moveLeft 2s linear 7.5s, /\* Comienza a los 7.5s (5 + 2.5) \*/

dissapear 2s linear 9.5s; /\* Comienza a los 9.5s (2 + 7.5) \*/

}

Hemos aplicado varias animaciones a la vez, estableciendo un retardo equivalente a la suma de la duración de las animaciones anteriores, encadenando una animación con otra.

Podemos utilizar Animate.css para dar dinamismo a nuestro contenido.

● En pocos pasos se pueden agregar animaciones CSS a cualquier elemento

con esta sencilla librería.

● En la creación de cualquier contenido web resulta interesante incorporar

animaciones que nos ayuden a mejorar la experiencia del usuario durante la

interacción con el contenido.

● Permite disponer de una gran variedad de animaciones CSS sin necesidad de

crearlas nosotros mismos.

● Esta librería permite conseguir que el contenido sea más atractivo y

dinámico.

**Atributos en CSS:**

**Unidades de Medidas Absolutas:** Son medidas fijas e indican cantidades exactas en alguna

unidad. Su valor real es directamente el valor indicado y se ve igual en todos los dispositivos No dependen de otro valor de referencia, por eso se llaman absolutas. La desventaja que tienen es que son muy poco flexibles.

cm: centimetros (10mm)

mm:milimetros

px:pixeles

pt:puntos, equivalente a 0.35mm

in:pulgada, equivalente a 2.54cm(25.4mm)

pc:picas, equivalente a unos 4.23 milimetros

**Unidades de medida Relativas:** Definen su valor en relación con otra medida y para obtener su

valor real se debe realizar alguna operación con el valor indicado. Dentro de las medidas relativas están las flexibles que son relativas al tamaño del viewport.

em :<<M>> 1em = tamaño de fuente establecida en navegador.

1em quivale a 16 pixeles por defecto, salvo que el usuario lo modifique.

ex: <<x>> (0,5 em apróx) 1ex = mitad del tamaño de la fuente del navegador aproximadamente.

ch: <<zero width>> 1ch = tamaño de ancho del cero (0).

Estas son de las menos utilizadas, tanto ex como ch. Ex es aproximadamente la mitad del tamaño de la fuente establecida, basado en un sistema de medida en base a la x minuscula de la fuente en la que se trabaja.

En tanto a ch, equivale a ancho 0 de la fuente actual

rem :<<root M>> 1rem = tamaño fuente raíz.

La unidad rem (root em) toma la idea de la unidad em, pero permite establecer un tamaño base personalizado (generalmente para el documento general, utilizando html o la pseudoclase :root). De esta forma podemos trabajar con multiplos del tamaño base

:root {font-size:22px:} tamaño base

h1{font-size 2rem} tamaño equivalente al doble del root, 44px

h2{font-size 1rem} tamaño equivalente al root, 22px

% :Porcentaje Relativa a herencia (contenedor padre)

El % define una unidad en funcion de otra, es decir, si trabajamos con 12px y definimos una unidad como 150%, obtendremos 18px.

**Unidades de medida flexibles:** Las unidades de medida flexibles (vw y vh, o vmin y vmax) son relativas al tamaño del ancho o alto del viewport (ventana grafica, region visible de la pagina web en el navegador, no el body)

vw: viewport width. Por ejemplo, si decimos que un div debe medir 50vw, es el equivalente al 50% del ancho total del viewport

vh: viewport height.Por ejemplo, si definimos que un div mide 50vh y el alto del viewport es 800px, nuestro div medira 400px.

**Propiedades:**

border: es el limite que separa el interior del exterior del elemento

margin: 5px; establece el margen para los 4 lados de un elemento, puede ser un numero cualquiera. Parte exterior del elemento, por fuera del borde.

paddin: 10px; establece el espacio de relleno. Parte interna del elemento, entre el contenido y el borde

content: es la parte interior del elemento, incluyendo el relleno.

margin-left: 10px; establece el margen al lado izquierdo del recuadro o pantalla.

Color: se puede usar en cualquier elemento, aunque principalmente se usa para modificar el color del texto y el background de un elemento. Se puede pasar el valor por palabra clave, valor hexadecimal, valor rgb, o valor rgba.

background-color: red; cambia el color de relleno

text-align: center; establece el texto en el centro del recuadro o pantalla

font-size: tamaño de la fuente (px,em,rem)

font-style: estilo de fuente (normal, italic, oblique)

font-family: lista de fuentes (arial, helvetica, sans-serif,etc)

font-weight: grosor (peso) de la fuente (bold, 400, 600, 800).

Modelo de caja:

Las cajas de las paginas no son visibles a simple vista porque inicialmente no muestran margen, ningún color de fondo ni ningún color de borde. Cada elemento incluido en el documento html genera una caja que contiene varios atributos modificables. El comportamiento de esa caja depende de su clasificación, es decir, si se trata de un elemento de linea o de bloque.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si estan definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia pagina (si están definidos).

Dimensiones y desbordamiento:

Proporcionamos tamaños específicos a los diferentes elementos de un documento HTML asignando valores a las propiedades width (ancho) y height (alto).

En el caso de utilizar el valor auto en las propiedades anteriores (valor por defecto), el navegador se encarga de calcular el ancho o alto necesario, dependiendo del contenido del elemento. El tamaño automático dado a un elemento depende del tipo de elemento que se trate (en bloque o en línea).

Si en lugar de utilizar la opción auto, o simplemente no indicamos valores para ancho y alto, el tamaño de la caja suele acomodarse al contenido sin problemas. Pero cuando asignamos valores a estos atributos, forzamos al elemento a tener un aspecto concreto, obteniendo resultados inesperados si su contenido es más grande que el tamaño que hemos definido.

Mínimos y máximos:

Una forma de mitigar el problema mencionado consiste en utilizar las propiedades

hermanas de width: min-width y max-width; y las propiedades hermanas de height:

min-height y max-height. Con estas propiedades, en lugar de establecer un tamaño

fijo, establecemos límites máximos y mínimos, donde el ancho o alto de la caja queda

comprendido entre esos valores.

Las propiedades de mínimos (min-width y min-height) por defecto tienen valor 0,

mientras que las propiedades de máximos (max-width y max-height) tienen por defecto

valor none.

Zonas de un elemento;

En CSS existen ciertas palabras clave para hacer referencia a una zona u orientación concreta sobre un elemento. Son conceptos muy sencillos y lógicos:

● Top: Parte superior

● Left: Parte izquierda

● Right: Parte derecha

● Bottom: Parte inferior

● Center: Se refiere a la posición central entre los extremos horizontales y verticales.

Desbordamiento:

Puede ocurrir que apliquemos un tamaño de alto y ancho a un elemento HTML, pero su contenido de texto sea tan grande que no quepa dentro de la caja.

En ese caso lo que ocurre es que el contenido se desborda. Podemos modificar este comportamiento con la propiedad de CSS overflow o con alguna de sus propiedades específicas overflow-x u overflow-y.

Dichas propiedades pueden tomar varios valores, donde visible es el valor que tiene por defecto, permitiendo el desbordamiento. Otras opciones son las siguientes:

overflow-x y overflow-y permiten ocultar alguna barra de desplazamiento (habitualmente la barra de desplazamiento horizontal).

Margen (margin):

Se utilizan para crear espacio alrededor de los elementos, fuera de los bordes definidos. Margin especifica el espacio que existe entre el borde de un elemento y el borde de otros elementos adyacentes. Las opciones son:

● margin-top establece un valor de margen superior.

● margin-right establece un valor de margen derecho.

● margin-bottom establece un valor de margen inferior.

● margin-left establece un valor de margen izquierdo.

Podemos aplicar diferentes márgenes a cada zona de un elemento utilizando cada una de estas propiedades, o dejando al navegador que lo haga de forma automática indicando el valor auto.

Para centrar un elemento podemos aplicar un ancho fijo al contenedor. Por ejemplo: width: 500px con margin: auto. El navegador, al conocer su tamaño, se encarga de repartirlo equitativamente entre los márgenes, dado que conoce el resto del tamaño de la ventana.

Por defecto, si tenemos dos elementos adyacentes con, por ejemplo, margin: 20px cada uno, ese espacio de margen se solapará y tendremos 20px en total, y no 40px (la suma de cada uno) como podríamos pensar en un principio.

Relleno (padding):

Al igual que con los márgenes, los rellenos tienen varias propiedades para indicar

cada zona:

● padding-top aplica un relleno interior en el espacio superior de un elemento.

● padding-right aplica un relleno interior en el espacio derecho de un elemento.

● padding-bottom aplica un relleno interior en el espacio inferior de un elemento.

● padding-left aplica un relleno interior en el espacio izquierdo de un elemento.

Como se puede ver en la tabla, por defecto no hay relleno (el relleno está en cero), aunque puede modificarse tanto con las propiedades anteriores como la propiedad de atajo: Modelo de caja (en la siguiente página)

Borde (border):

Establece un límite entre la parte interior y la parte exterior de la caja. Se pueden especificar estilo, ancho y color. Las opciones son:

● border-top

● border-right

● border-bottom

● border-left

Las propiedades básicas y específicas de los bordes en CSS son las siguientes:

● border-color especifica el color que se utilizara en el borde.

● border-width especifica el tamaño predefinido para el grosor del borde.

● border-style define el estilo del borde a utilizar.

El estilo de borde más frecuente es solid (borde liso y continuo), y que además es la

opción por defecto. Pueden utilizarse cualquiera de los estilos indicados en la tabla

anterior e incluso combinar con otras propiedades.

● hidden: Oculto. Idéntico a none, salvo para conflictos con tablas.

● dotted: Borde basado en puntos.

● dashed: Borde basado en rayas (línea discontinua).

● solid: Borde sólido (línea continua).

● double: Borde doble (dos líneas continuas).

● groove: Borde biselado con luz desde arriba.

● ridge: Borde biselado con luz desde abajo. Opuesto a groove.

● inset: Borde con profundidad «hacia dentro».

● outset: Borde con profundidad «hacia fuera». Opuesto a inset.

Bordes múltiples:

Sólo hemos utilizado un parámetro en cada propiedad, aplicando el mismo valor a cada borde de un elemento. Sin embargo, podemos especificar de uno a cuatro parámetros, dependiendo de lo que queramos hacer:

border-color: color 1 parametro aplica el mismo color a todos los bordes.

color color 2 parametros. Aplica al borde top/bottom y left/right.

color color color 3 parametros. Aplica al top, a left/right y a bottom.

color color color 4 parametros. Aplica al top, right, bottom, left.

Podemos hacer lo mismo con las propiedades border-width y border-style. Teniendo en cuenta esto, disponemos de una gran flexibilidad a la hora de especificar esquemas de bordes más complejos.

Con la propiedad border-width pasa exactamente lo mismo que con margin y padding,

actuando en este caso en relación al grosor del borde de un elemento. Se pueden utilizar

de 1 a 4 parámetros.

Bordes – Atajos:

Con tantas propiedades, incluso para hacer algo sencillo necesitamos varias líneas de código. Pero podemos utilizar la propiedad de atajo border, con la que podemos hacer un resumen sin necesidad de indicar múltiples propiedades individuales por separado, realizando el proceso de forma más corta:

border size – style – color Propiedad de atajo para simplificar valores

div{

border: 1px solid red

}

Bordes especificos:

Una forma más intuitiva, es utilizar las propiedades de bordes específicos (por zonas) y

aplicar estilos combinándolos junto a la herencia de CSS. Para utilizarlas bastaría con

indicar la zona justo después de border-.

Border-bottom-color: black;

Y asi con todas sus propiedades.

Box-sizing:

Indica cómo se debe calcular el ancho y el alto total de un elemento. Acepta los valores:

● box-sizing: content-box: Es el valor que cualquier caja tiene asignada por defecto. Las propiedades width y height no incluyen el borde, padding o margin.

● box-sizing: border-box: Las propiedades width y height incluyen el contenido, padding y borde pero no el margin.

● box-sizing: initial: configura esta propiedad en su valor predeterminado

● box-sizing: inherit: hereda la propiedad de su elemento padre.

En el modelo de caja CSS “clásico”, el borde y los márgenes interior y exterior se añaden al tamaño del elemento definido con las propiedades width y height.

La propiedad box-sizing, permite modificar este comportamiento y hacer que el borde y los márgenes interior y exterior se puedan incluir en el interior del tamaño definido con las propiedades width y height. En este caso se reducirá el espacio disponible para el contenido.

Posicionamiento:

Los elementos en línea y en bloque nos permiten diseñar esquemas complejos, alineando y combinando elementos.

El posicionamiento por defecto es el estático (static): todos los elementos aparecen con un orden natural según donde estén colocados en el HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Existen otros modos alternativos de posicionamiento, que podemos cambiar mediante la propiedad position, que nos permiten modificar la posición de los elementos y su contenido.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

A la propiedad position se le pueden indicar los siguientes valores:

● static: es el valor por defecto, un elemento con este valor no está posicionado con CSS.

● relative: mediante las propiedades top | bottom | right y/o left el elemento se posiciona en forma relativa a su contenedor.

● absolute: hace que un elemento se ubique considerando su contenedor posicionado más cercano. Si no encuentra ningún contenedor cercano, el elemento se colocará respecto al viewport. El resto de elementos de la página ignoran la nueva posición del elemento.

● fixed: la caja está posicionada con respecto a la ventana del navegador. Se mantiene en el mismo lugar incluso al hacer scroll en la página. La referencia es el viewport, la parte visible del navegador.

● sticky: al realizar un scroll y después de alcanzar una posición de desplazamiento determinada, se “pega” al borde elegido.

Si utilizamos un modo de posicionamiento diferente al estático (absolute, fixed, sticky o relative), podemos emplear una serie de propiedades para modificar la posición de un elemento.

Las propiedades top, bottom, left y right sirven para mover un elemento desde la orientación que su propio nombre indica hasta su extremo contrario. Esto es, si utilizamos left e indicamos 20px, estaremos indicando mover desde la izquierda 20 píxeles hacia la derecha.

Posicionamiento relativo (relative):

Si utilizamos la palabra clave relative activaremos el modo de posicionamiento relativo. En este modo, los elementos se colocan exactamente igual que en el posicionamiento estático (permanecen en la misma posición), pero dependiendo del valor de las propiedades top, bottom, left o right variamos la posición del elemento.

.contenedor {

position: relative;

left: 30px;

top: 10px;

border: 2px solid blue;

background-color: lightblue;

width: 300px;

height: 100px;

margin: 0px;

}

Posicionamiento absoluto (absolute):

Utilizando la palabra clave absolute indicamos que el elemento utiliza posicionamiento absoluto. Este tipo de posicionamiento coloca los elementos utilizando como punto de origen el primer contenedor con posicionamiento diferente a estático. Si no existe, emplea el documento completo como referencia.

Si el contenedor padre tiene posicionamiento estático, se analiza el posicionamiento del padre del contenedor padre, y así sucesivamente hasta encontrar un contenedor con posicionamiento no estático o llegar a la etiqueta <body>, situación en la que se comportaría como en el ejemplo.

Ejemplo: Si establecemos left:40px, el elemento se colocará 40 píxeles a la derecha del extremo

izquierdo de la página. Sin embargo, si indicamos right:40px, el elemento se colocará 40 píxeles a

la izquierda del extremo derecho de la página.

Posicionamiento fijo (fixed) y “pegajoso” (sticky):

El posicionamiento fijo (fixed) hace que el elemento se muestre en una posición fija dependiendo de la región visual del navegador. Es decir, aunque el usuario haga scroll y se desplace hacia abajo en la página web, el elemento seguirá en el

mismo sitio posicionado.

Ejemplo: Si establecemos top:0 y right:0, el elemento se colocará justo en la esquina superior derecha y se mantendrá ahí aunque hagamos scroll hacia abajo en la página.

El posicionamiento sticky se utiliza cuando queremos que un elemento se posicione en un lugar específico de forma fija («sticky», pegajoso).

Ejemplo: Al hacer scroll y llegar a un elemento este podrá quedarse fijo en la parte superior mientras continuamos haciendo scroll. No debemos confundir con el fijo ya que no queda en una posición fija, sino que flota respecto del fondo y se queda adherido a la parte superior.

Profundidad (z-index):

Establece el nivel de profundidad de un elemento sobre los demás. De esta forma, podremos superponer los elementos, quedando “apilados”. Los elementos se posicionan de acuerdo al nivel de profundidad, quedando “encima” los que poseen un valor de index mayor.

.box{

position: absolute;

width:100px;

height:140px;

border: 2px solid black; opacity: 90%;

}

.z1 {

background-color: lightcoral;

left: 0px;

top: 0px;

z-index: 20;

}

.z2 {

background-color: lightblue;

left: 30px;

top: 30px;

z-index: 10;

}

.z3 {

background-color: lightgreen;

left: 60px;

top: 60px;

z-index: 0;

}

Propiedad visibility y display:

La propiedad de visibilidad especifica si un elemento es visible o no.

Nota: los elementos ocultos ocupan espacio en la página. Puede usar la propiedad display para ocultar y eliminar un elemento del diseño del documento.

Sintaxis: visibility: visible|hidden|collapse|initial|inherit;

La propiedad display especifica el comportamiento de visualización de un elemento. Ejemplo: especifica si un elemento es tratado como block or inline.

En HTML, el valor de la propiedad de visualización predeterminada se toma de las especificaciones de HTML o de la hoja de estilo predeterminada del navegador/usuario.

Sintaxis: display: value;

Cada elemento tiene un valor de display por defecto. Recordemos que los navegadores le dan a los elementos valores por defecto block e inline:

● block: el elemento empieza en una nueva línea (div, h1-h6, header, etc)

● inline: puede contener algo de texto dentro de un párrafo sin interrumpir el flujo del párrafo.

● none: es utilizado para ocultar elementos sin eliminarlos, no deja un espacio donde el elemento se encontraba.

● inline-block: Los elementos inline-block fluyen con el texto y demás elementos como si fueran elementos en línea y además respetan el ancho, el alto y los márgenes verticales.

Cada etiqueta HTML tiene un tipo de representación visual en un navegador, lo que habitualmente se suele denominar el tipo de caja. En principio, se parte de dos tipos básicos: inline y block.

Tipos de display:

La tabla siguiente muestra una lista de los valores de la propiedad:

* block : Se aplica en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
* Inline : Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora width o height.
* Inline-block: Combinacion de los dos anteriores. Se comporta como inline pero no ignora width y height.
* Flex: Utiliza un modelo de cajas flexibles Flexbox. Muy util para diseños adaptables.
* Inline-flex: La version en linea (ocupa solo su contenido) del modelo de cajas flexibles flexbox.
* Grid: Utiliza las cuadriculas o rejillas con modelo de cajas Grid CSS.
* Inline-grid: La version en linea (ocupa solo su contenido) del modelo de cajas grid css.
* List-item: Actua como un item de una lista. Es el comportamiento de etiquetas como <li>.
* Table: Actua como una tabla. Es el comportamiento de etiquetas como <table>.
* Tablle-cell: Actua como la celda de una tabla. Es el comprotamiento de etiquetas como <th> o <td>.
* Table-row: Actua como la fila de una tabla. Es el comportamiento de etiquetas como <tr>.

Ocultar elementos:

En la lista anterior, falta un valor de la propiedad display. Mediante la mencionada propiedad, es posible aplicar un valor none y ocultar completamente elementos que no queramos que se muestren.

Es muy útil para hacer desaparecer información cuando el usuario realiza alguna acción.

No obstante, también existe una propiedad CSS llamada visibility que realiza la misma acción, con la ligera diferencia de que no sólo oculta el elemento, sino que además mantiene un vacío con el mismo tamaño de lo que antes estaba ahí.

Dicha propiedad visibility puede tomar los siguientes valores:

* visible: el elemento es visible. Valor por defecto.
* Hidden: el elemento no es visible, pero sigue ocupando propiedad y posicion.
* Collapse: solo para tablas. El elemento se contrae pero no ocupa espacio.

Utilizar visibility: hidden es muy interesante si queremos que un elemento y su contenido se vuelva invisible, pero siga ocupando su espacio y así evitar que los elementos adyacentes se desplacen, lo que suele ser un comportamiento no deseado en algunas ocasiones cuando se aplica display: none.

Otra opción interesante es utilizar la propiedad opacity junto a transiciones o animaciones, desplazarse desde el valor 0 al 1 o viceversa. De esta forma conseguimos una animación de aparición o desvanecimiento.

**FlexBox:**

Tradicionalmente, en CSS se ha utilizado el posicionamiento (static, relative, absolute...), los elementos en línea o en bloque (y derivados) o los float, lo que a grandes rasgos no deja de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que tenemos hoy en día: sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc…

Flexbox es un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños. Está especialmente

diseñado para crear, mediante CSS, estructuras de una sola dimensión.

Conceptos

Elementos básicos de Flexbox:

● Contenedor: Es el elemento padre que tendrá en su interior cada uno de

los ítems flexibles.

○ Eje principal: Los contenedores flexibles tienen una orientación principal

específica. Por defecto es el eje horizontal.

○ Eje secundario: La orientación secundaria es perpendicular a la principal.

● Ítem: Son los elementos hijos flexibles del contenedor.

Imaginemos el siguiente escenario:

Para activar el modo flexbox, hemos utilizado sobre el elemento contenedor la propiedad display y especificamos el valor flex o inline-flex (dependiendo de cómo queramos que se comporte el contenedor)

<div class="container"> <!-- Flex container -->

<div class="item item-1">1</div> <!-- Flex items -->

<div class="item item-2">2</div>

<div class="item item-3">3</div>

</div>

Propiedad display:

inline-flex: establece un contenedor en linea, similar a inline-block (ocupa solo el contenido.  
flex: establece un contenedor en bloque, similar a block (ocupa todo el ancho del padre)

Dirección de los ejes

Existen dos propiedades principales para manipular la dirección y comportamiento de los ítems a lo largo del eje principal del contenedor. Son las siguientes:

flex-direction: row|row-reverse|column|column-reverse: cambia la orientacion del eje principal.

Flex-wrap: nowrap|wrap|wrap-reverse: evita o permite el desbordamiento (multilinea).

Mediante la propiedad flex-direction podemos modificar la dirección del eje principal del contenedor para que se oriente en horizontal (por defecto) o en vertical. Además, también podemos incluir el sufijo -reverse para indicar que coloque los ítems en orden inverso.

row: establece la direccion del eje principal en horizontal.

row-reverse: establece la direccion del eje principal en horizontal (invertido).

column: establece la direccion del eje principal en vertical.

column-reverse: establece la direccion del eje principal en vertical (invertido).

row y row-reverse determinan el orden de los elementos. Aplicando estas propiedades modificamos el flujo del eje principal:

.container {

background: steelblue;

display: flex;

flex-direction: column;

}

.item {

background: grey;

}

row

row-reverse

column

columnreverse

Flexbox | flex-wrap

Existe otra propiedad llamada flex-wrap con la que podemos especificar el comportamiento del contenedor respecto a evitar que se desborde (nowrap, valor por defecto) o permitir que lo haga, en cuyo caso, estaríamos hablando de un contenedor flexbox multilínea.

nowrap: establece los items en una sola linea (no permite que se desborde el contenedor).

wrap: establece los items en modo multilinea (permite que se desborde el contenedor).

wrap-reverse: establece los items en modo multilinea, pero en direccion inversa.

Teniendo en cuenta estos valores de la propiedad flex-wrap, podemos conseguir cosas como la siguiente:

.container {

background: steelblue;

display: flex;

width: 200px;

flex-wrap: wrap; /\* Comportamiento por defecto: nowrap \*/

}

.item {

background: grey;

width: 50%;

}Flexbox | flex-wrap

Con nowrap los 3 ítems se muestran en una misma línea. El tamaño de los items se ajustan al contenedor, manteniendo sus proporciones.

Si especificamos wrap el contenedor se puede desbordar, pasando a ser un contenedor multilínea que muestra uno o más elementos en la línea siguiente.

nowrap

wrap

wrapreverse

Atajo: dirección de los ejes

Existe una propiedad de atajo (short-hand) llamada flex-flow, con la que podemos resumir los valores de las propiedades flex-direction y flex-wrap, especificandose en una sola propiedad y ahorrándonos utilizar las propiedades concretas:

.container {

/\* flex-flow: <flex-direction> <flex-wrap>; \*/

flex-flow: row wrap;

}

Propiedades de alineación

Disponemos de 4 propiedades relativas a la alineación, la primera relativa al eje principal y las restantes al secundario:

justify-content: flex-start|flex-end|center|space-between|space-around|space-evenly. eje1

align-content: flex-start|flex-end|center|space-between|space-around|space-evenly|stretch. eje2

align-items: flex-start|flex-end|center|stretch|baseline. eje3

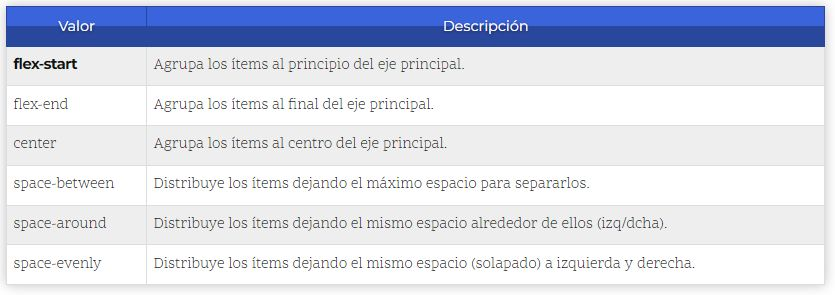
align-self: auto|flex-start|flex-end|center|stretch|baseline. eje4

● justify-content: Alinea los ítems del eje principal.

● align-items: Alinea los ítems del eje secundario.

Eje principal

La propiedad justify-content sirve para colocar los ítems de un contenedor mediante una disposición concreta a lo largo del eje principal:



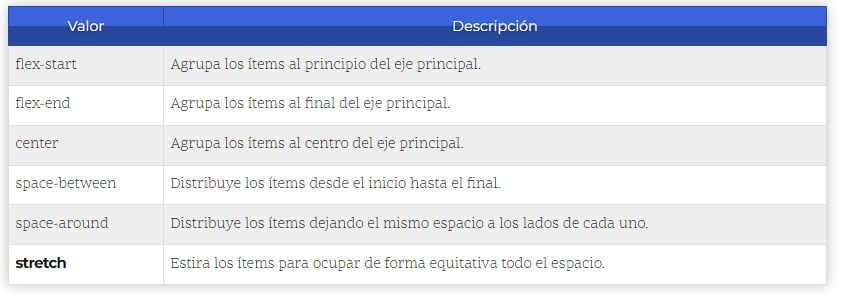
Con estos valores de la propiedad justify-content modificamos la disposición de los ítems del contenedor, distribuyéndose como se ve en el siguiente ejemplo (nótense los números para observar el orden de cada ítem):

flex-start space-between

flex-end space-around

center space-evenly

align-content permite manejar contenedores flex multilínea. Estos contenedores dividen el eje principal en múltiples líneas, dado que los ítems no caben en el ancho disponible. Sus valores son los siguientes:



En un contenedor multilínea de 200 píxeles de alto con ítems de 50px de alto, podemos utilizar la propiedad align-content para alinear los ítems de forma vertical de modo que se queden en la zona inferior del contenedor:

.container {

background: #CCC;

display: flex;

width: 200px;

height: 200px;

flex-wrap: wrap;

align-content: flex-end;

}

.item {

background: #777;

width: 50%;

height: 50px;

}

flex-start

flex-end

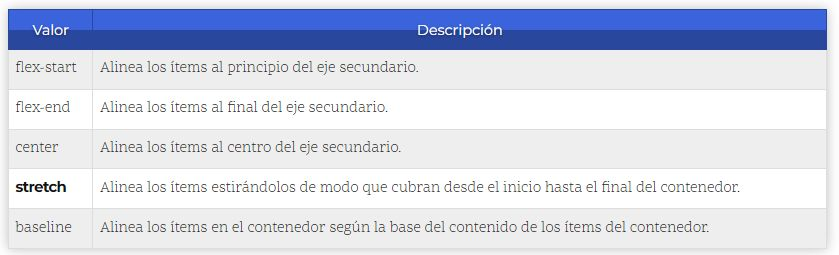
center

space-between space-around

stretch

Flexbox | Eje secundario

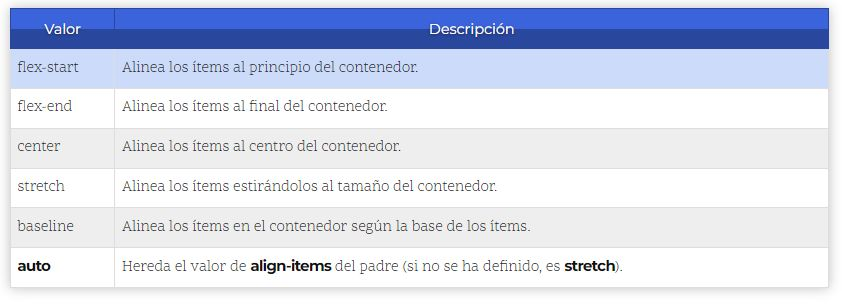
align-items alinea los ítems en el eje secundario del contenedor. A diferencia de align-content, align-items opera sobre el eje secundario. Los valores que puede tomar son los siguientes:



flex-start center

space-between space-around

align-self actúa como align-items, pero sobre un ítem hijo específico, sobreescribiendo su comportamiento. La propiedad puede tomar los siguientes valores:



Si se especifica el valor auto a la propiedad align-self, el navegador le asigna el valor de la propiedad align-items del contenedor padre, y en caso de no existir, el valor por defecto stretch. Ver segundo ejemplo interactivo aquí. Existe un atajo para establecer valores de align-content y de justify-content

a la vez, denominada place-content. Las dos clases siguientes proporcionan las mismas características:

space-around

.container {

display: flex;

place-content: flex-start flex-end;

}

.container {

align-content: flex-start;

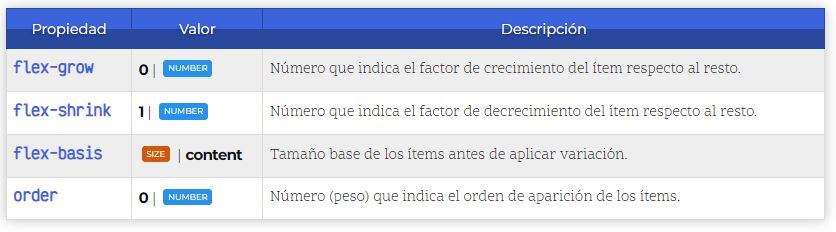
justify-content: flex-end;

}

Si con flex-grow indicamos un valor de 1 a todos los ítems, todos serán del mismo tamaño. Si colocamos un valor de 1 a todos, pero a uno le indicamos 2, ese ítem será más grande que los anteriores. Por defecto tienen un valor de 0.

Propiedades de hijos

Las siguientes propiedades se aplican sobre los ítems hijos:



flex-shrink es opuesta a flex-grow, aplica un factor de decrecimiento. De esta forma, los ítems que tengan un valor numérico más grande, serán más pequeños, mientras que los que tengan un valor numérico más pequeño serán más grandes, justo al contrario de como funciona la propiedad

flex-grow.

Por último, tenemos la propiedad flex-basis, que define el tamaño por defecto (de base) que tendrán los ítems antes de aplicarle la distribución de espacio. Generalmente, se aplica un tamaño (unidades, porcentajes, etc...), pero también se puede aplicar la palabra clave content que ajusta

automáticamente el tamaño al contenido del ítem, que es su valor por defecto.

Atajo: Propiedades de hijos

Existe una propiedad llamada flex que sirve de atajo para estas tres propiedades de los ítems hijos. Funciona de la siguiente forma:

.item {

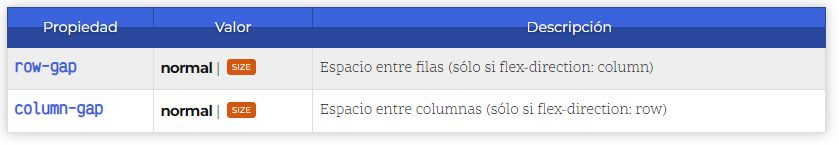
/\* flex: <flex-grow> <flex-shrink> <flex-basis> \*/

flex: 1 3 35%;

}

Huecos (gaps)

Las propiedades row-gap y column-gap establecen el tamaño del «hueco» entre ítems desde el elemento contenedor, sin necesidad de utilizar padding o margin en los elementos hijos.



Sólo una de las dos propiedades tendrá efecto, dependiendo de si flex-direction está establecida en column o en row. Es posible usar ambas cuando flex-wrap: wrap, disponiendo de multicolumnas flexbox.

Orden de los ítems

Por último, y quizás una de las propiedades más interesantes, es order, que modifica y establece el orden de los ítems según una secuencia numérica. Por defecto, todos los ítems flex tienen un order: 0 implícito, aunque no se especifique. Si indicamos un order con un valor numérico, se reubican los ítems según su número, colocando primero los ítems con número más pequeño (incluso valores negativos) y después los ítems con números más altos. De esta forma reordenamos fácilmente los ítems, incluso utilizando media queries o responsive design.

**Grid:**

CSS Grid es una muy buena opción para estructurar, organizar y diseñar los elementos de nuestro sitio web, ofreciéndonos un sistema de disposición apropiado de forma nativa en el navegador.

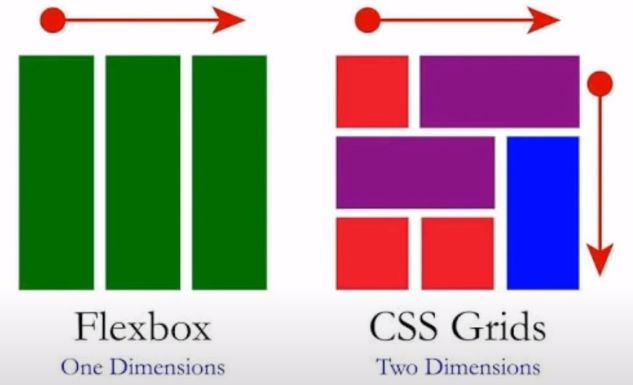
En CSS solemos utilizar múltiples propiedades para estructurar nuestros sitio web, utilizando inline-block, float, relative o hasta flexbox que sólo utiliza una dimensión.

El módulo CSS Grid entra en acción ofreciéndonos un sistema de filas y columnas en dos dimensiones, y total libertad de los ítems para organizar nuestros contenido a gusto.

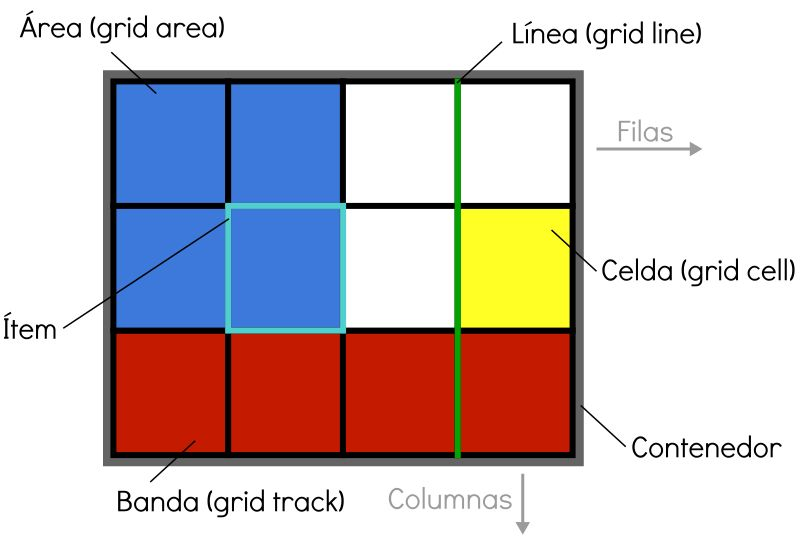
¿Y donde queda el Responsive Design? Con CSS Grid podemos definir filas y columnas como también redefinirlas a nuestro gusto dentro de un media query.

¿Flexbox vs CSS Grid?

CSS Grid puede convivir con Flexbox. No se trata de tecnologías excluyentes, sino complementarias.



El módulo de diseño de CSS Grid ofrece un sistema de diseño basado en cuadrículas, con filas y columnas, lo que facilita el diseño de páginas web sin tener que usar flotadores y posicionamiento.



Conceptos Básicos:

● Grid Container: es nuestro elemento “padre”, donde se asigna un {display:grid;} y nos permitirá colocar otras propiedades para manipular nuestro diseño.

● Grid Item: son los hijos directos de nuestro container. Estos los manejaremos a nuestra voluntad, nuestras filas y columnas que moveremos a nuestro gusto.

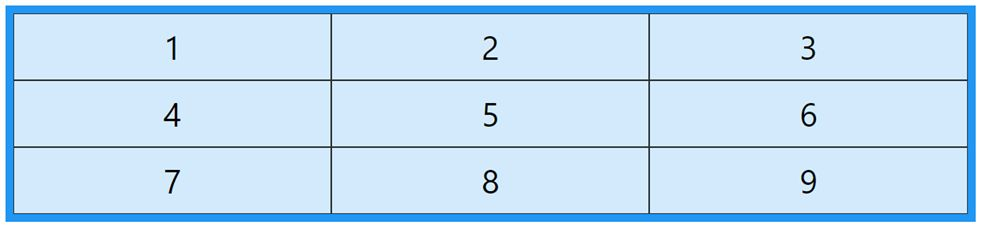
● Grid Line: son las líneas divisorias horizontales y verticales.

● Grid Track: es el espacio entre dos líneas adyacentes. Filas y columnas.

● Grid Cell: nuestras celdas serán el espacio entre dos filas adyacentes y 2 columnas adyacentes.

● Grid Area: espacio rodeado por 4 grid lines.

Conceptos básicosCSS Grid | Contenedor

<div> <!-- Elemento padre -->

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

CSS Grid | Display

Un elemento HTML se transforma en un contenedor de grilla cuando tiene su propiedad display seteada en grid o inline-grid:

.grid-container{

display: grid;

}

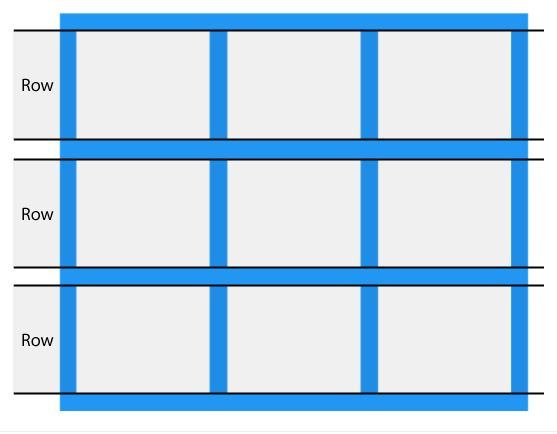
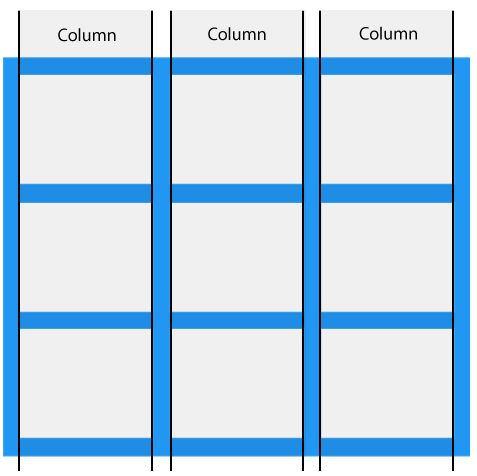
.grid-container{

display: inline-grid;

}

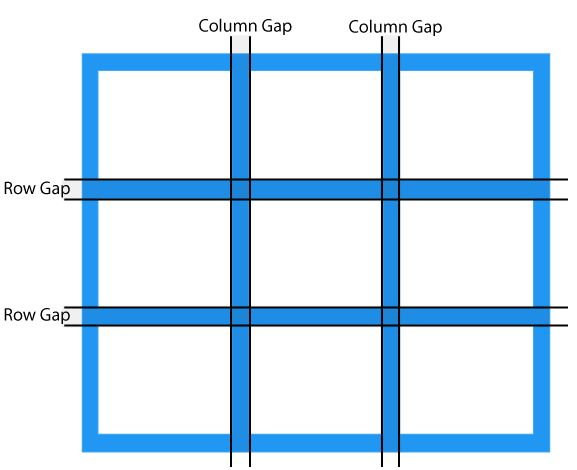
CSS Grid | Grid Items

Se pueden referenciar por fila o por columna, aunque no es la única forma.



CSS Grid | Grid Gaps

Es el espacio entre los ítems. Se pueden ajustar los tamaños de gap con las siguientes propiedades: grid-column-gap; grid-row-gap y grid-gap.



CSS Grid | Grid Lines

Para colocar un grid-ítem en un contenedor se referencian los números de línea:

.item{

Pone un grid item en la línea

de la columna 1, que finaliza

en la línea de la columna 3.

grid-column-start: 1;

grid-column-end: 3;

}

.item{

Pone un grid item en la línea

de la fila 1, que finaliza en la

línea de la fila 3.

grid-row-start: 1;

grid-row-end: 3;

}

CSS Grid | Propiedades

Propiedad Descripción

column-gap Especifica el espacio entre las columnas.

gap Propiedad abreviada. Espacio entre filas y entre columnas .

grid Propiedad abreviada. Filas y columnas, de la cuadrícula, áreas de la cuadrícula, filas y

columnas automáticas y propiedades de flujo automático de cuadrícula.

grid-area Especifica un nombre para el elemento de la cuadrícula. Es una propiedad abreviada

para grid-row-start, grid-column-start, grid-row-end y grid-column-end.

grid-auto-columns Especifica un tamaño de columna predeterminado.

grid-auto-flow Determina cómo se insertan los elementos en la cuadrícula.

grid-auto-rows Especifica un tamaño de fila predeterminado.

grid-column Una propiedad abreviada para las propiedades grid-column-start y grid-column-end.

grid-column-end Especifica dónde termina el elemento de la cuadrícula.

grid-column-gap Especifica el tamaño del espacio entre columnas.

grid-column-start Especifica dónde comienza el elemento de la cuadrícula.

Grid-gap Una propiedad abreviada para las propiedades grid-row-gap y grid-column-gap.

grid-row Una propiedad abreviada para las propiedades grid-row-start y grid-row-end.

grid-row-end Especifica dónde termina el elemento de la cuadrícula.

grid-row-gap Especifica el tamaño del espacio entre filas.

grid-row-start Especifica dónde comienza el elemento de la cuadrícula.

grid-template Una propiedad abreviada para las propiedades de las filas de plantilla de

cuadrícula, columnas de plantilla de cuadrícula y áreas de cuadrícula.

grid-template-areas Especifica cómo mostrar columnas y filas, utilizando elementos de cuadrícula con nombre.

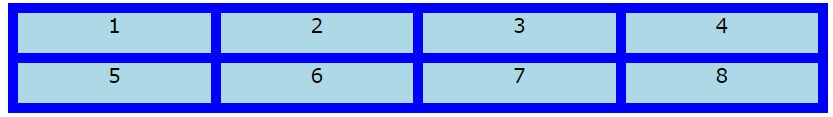
Grid-template-columns Especifica el tamaño de las columnas y cuántas columnas en un diseño de cuadrícula.

grid-template-rows Especifica el tamaño de las filas en un diseño de cuadrícula.

row-gap Especifica el espacio entre las filas de la cuadrícula.

CSS Grid | Grid Container

Para que un elemento HTML se comporte como un contenedor de cuadrícula, debemos establecer la propiedad display en grid (cuadrícula) o inline-grid (cuadrícula en línea). Los contenedores de cuadrícula consisten en elementos de cuadrícula, colocados dentro de columnas y filas.



CSS Grid | Grid Container

Este es el código que genera el cuadro de la diapositiva anterior:

<div class="grid-container">

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

<div>6</div>

<div>7</div>

<div>8</div>

</div>

.grid-container {

width: 800px;

display: grid;

grid-template-columns: repeat(4, auto);

grid-gap: 10px;

background-color: blue;

padding: 10px;

}

.grid-container > div {

background-color: lightblue;

text-align: center;

font-size: 20px;

height: 40px;

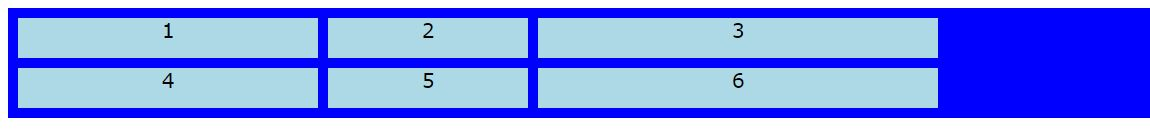
}

CSS Grid | grid-template-columns

La propiedad grid-template-columns define el número de columnas (y el ancho) de la cuadrícula. Se colocan los valores separados por espacios, y cada uno define el ancho de la columna respectiva. Se pueden establecer anchos en px, unidades relativas o %, aunque es recomendable utilizar la medida fr.

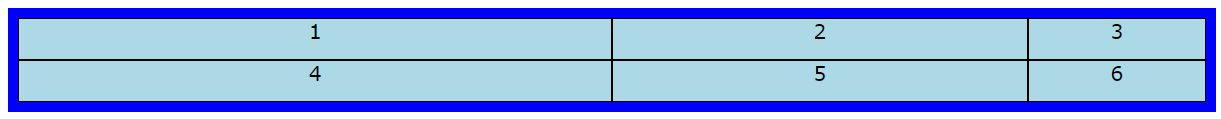
/\*Medidas en px:\*/

grid-template-columns: 300px 200px 400px;CSS Grid | grid-template-columns



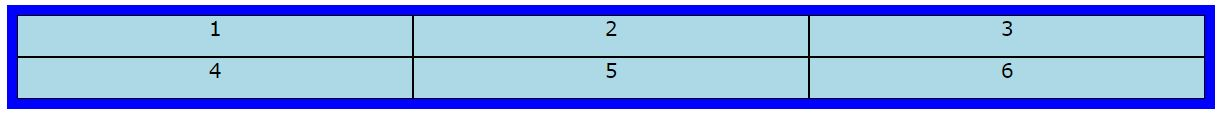
/\*Medidas en %:\*/

grid-template-columns: 50% 35% 15%;



/\*Medida automática:\*/

grid-template-columns: auto auto auto;



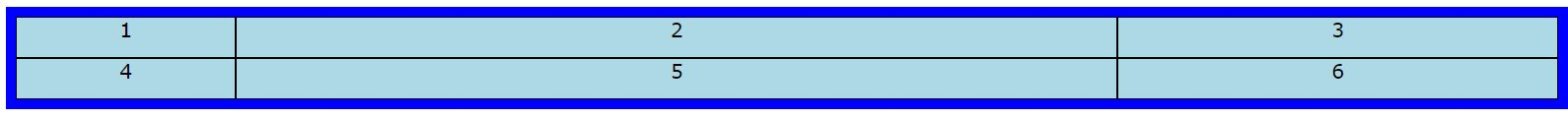
Si una cuadrícula de 4 columnas tiene más de 4 elementos, se agrega automáticamente una nueva fila para colocar los elementos extra.

CSS Grid | grid-template-columns

La unidad especial de Grid fr (fraction) representa una fracción de espacio restante en el grid.

Unidad fracción restante (fr):

grid-template-columns: 0.5fr 2fr 1fr;



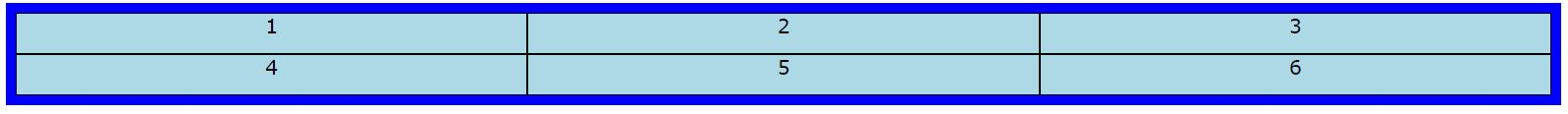
CSS Grid | grid-template-columns

Se puede utilizar la expresión repeat() para indicar repetición de valores, indicando el número de veces que se repiten y el tamaño en cuestión. La expresión a utilizar es la siguiente:

repeat([núm de veces], [valor o valores]):

La expresión repeat():

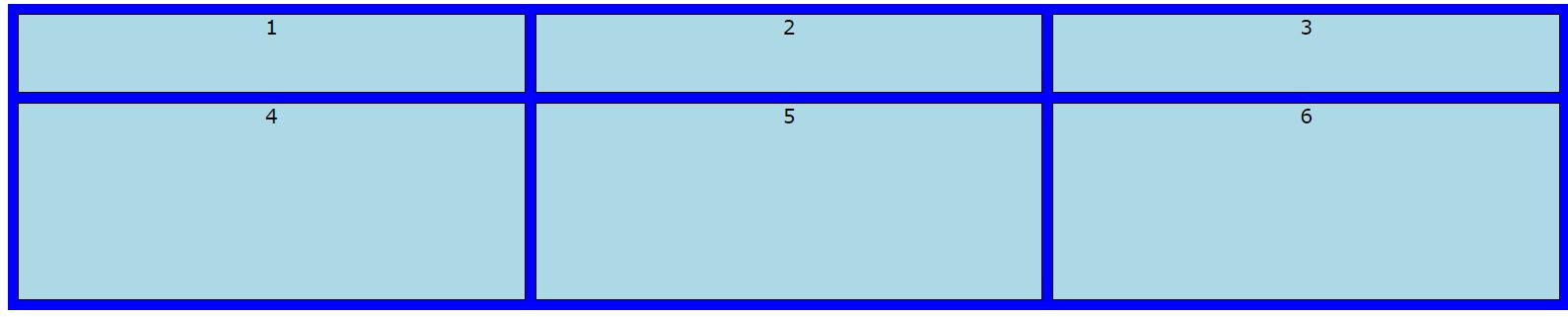
grid-template-columns: repeat(3, 1fr);



CSS Grid | grid-template-rows

La propiedad grid-template-rows define la altura de cada fila. El valor es una lista separada por espacios, donde cada valor define el alto de la fila respectiva.

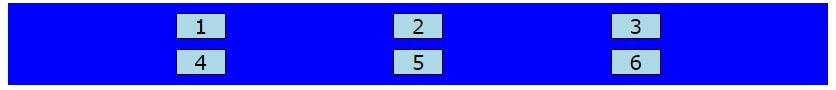
grid-template-rows: 80px 200px;



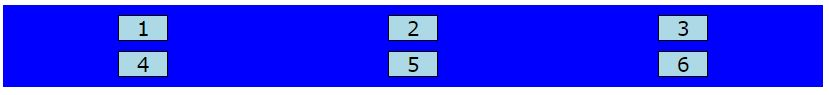
CSS Grid | Propiedad justify-content

La propiedad justify-content se utiliza para alinear toda la cuadrícula dentro del contenedor. El ancho total de la cuadrícula debe ser menor que el ancho del contenedor para que la propiedad tenga efecto.

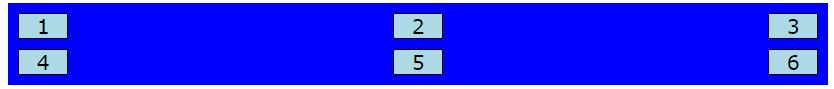
Space-evenly



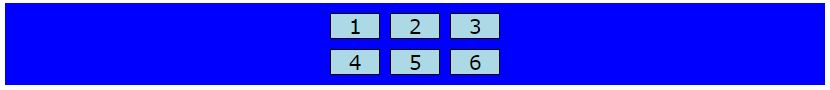
space-around



space-between



center



start



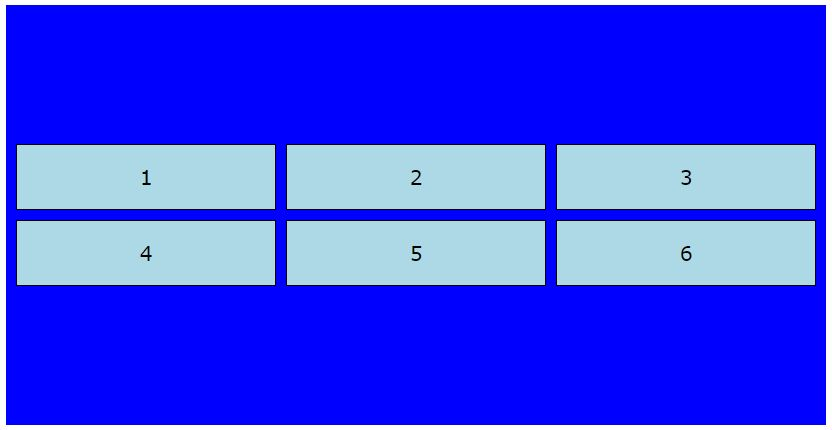
end



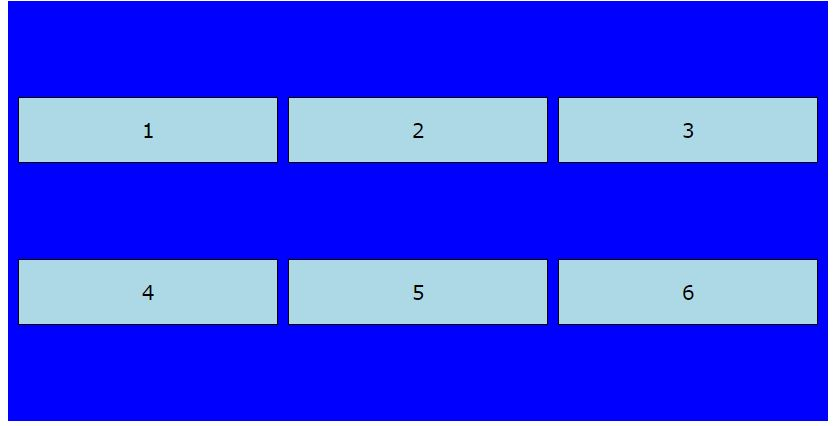
CSS Grid | Propiedad align-content

La propiedad align-content se usa para alinear verticalmente toda la cuadrícula dentro del contenedor. La altura total de la cuadrícula debe ser menor que la altura del contenedor para que tenga efecto.

Center

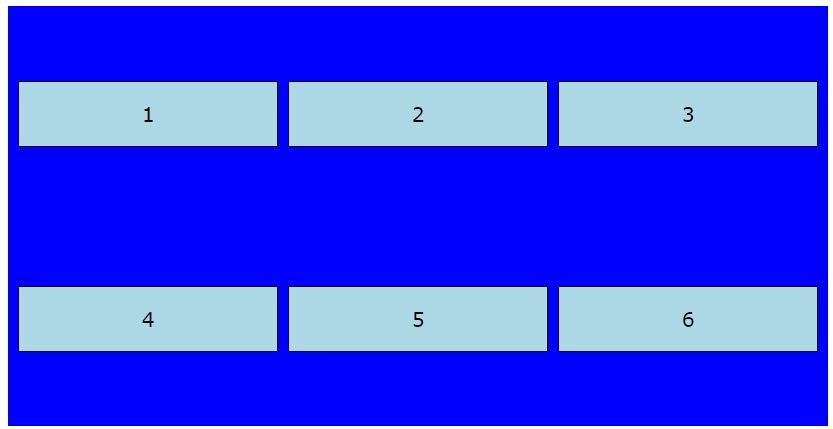


space-evenly

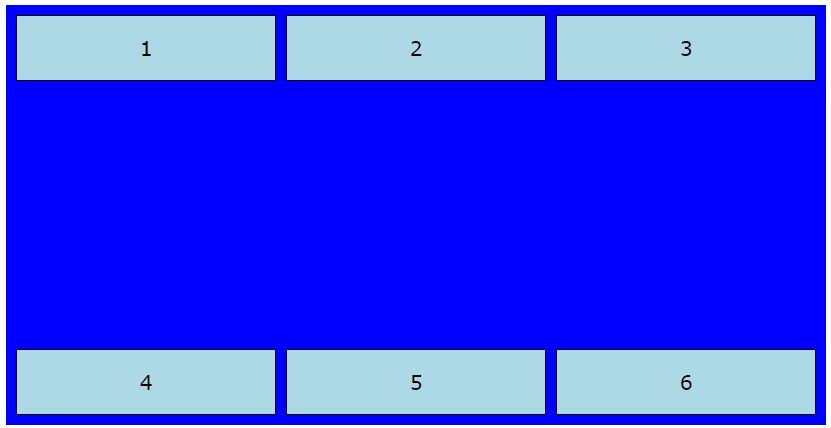


CSS Grid | Propiedad align-content

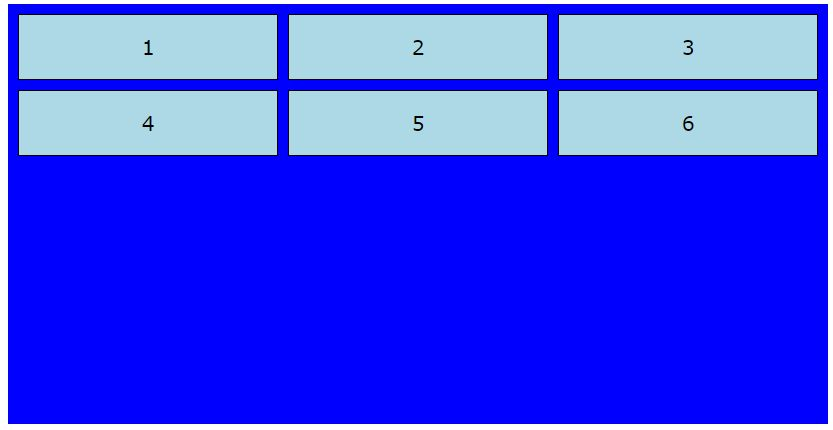
space-around



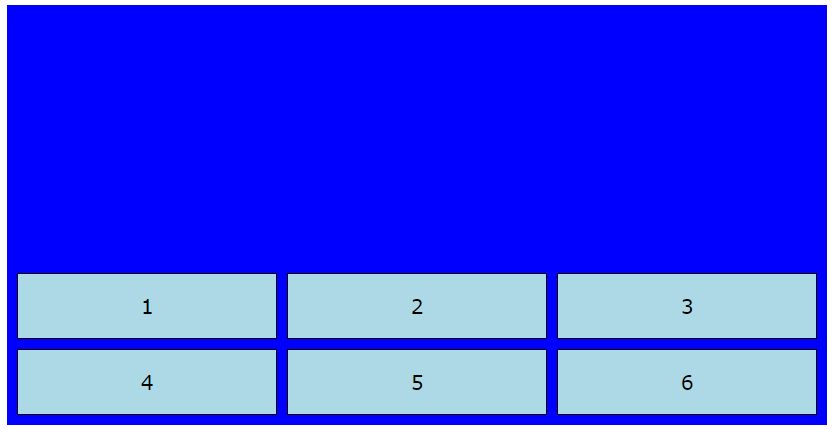
space-between



start



end



CSS Grid | Propiedad grid-area

La propiedad grid-area especifica el tamaño y la ubicación de un elemento de cuadrícula en el diseño, y es una propiedad abreviada para las siguientes

propiedades: grid-row-start, grid-column-start, grid-row-end y

grid-column-end.

La grid-area también se puede utilizar para asignar un nombre a un elemento de la cuadrícula. Y se puede hacer referencia a los elementos de cuadrícula con nombre mediante la propiedad grid-template-areas del contenedor de cuadrícula.

Bootstrap:

Conjunto de herramientas, librerias, convenciones y byenas practicas que encapsulan las tareas repetitivas en modulos genericos facilmente reutilizables. Contiene herramientas y hojas de estilos que nos permiten centrarnos en los elementos centrales de cada diseño.

Permite crear interfaces web con CSS y JavaScript, adaptando la interfaz del sitio web al tamaño del dispositivo en que se visualice.

Ventajas:

Facil de usar

Responsive: permite crear paginas web adaptables a cualquier dispositivo.

Personalizable

Gran comunidad.

Instalacion:

Forma 1: descargar desde bootstrap.com .En este caso los archivos deben estar en la misma carpeta del proyecto y referenciados en el <head> del documento html con <link>.

<link rel="stylesheet" href="css/bootstrap.min.css">

<script src="js/bootstrap.min.js"></script>

Forma 2: incluir bootstrapcdn (content delivery network) en el <head>. Este metodo tiene la ventaja de no necesitar instalacion alguna, pero nuestro sitio va a estar conectado permanentemente con el sitio web bootstrap, proveyendo los estilos.

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstra

p.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83

DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3AhiU" crossorigin="anonymous">

Class Container:

Los contenedores sirven para crear una caja o contenedos dentro de la que se coloca el contenido de una pagina web.

Cuando aplicamos a un elemento HTML la clase container, a ese elemento se le aplica un ancho y un padding determinado y además se centra horizontalmente en la página web.

Bootstrap proporciona 3 tipos de contenedores diferentes, cada uno con sus características distintivas.

.container: establece un ancho maximo o max-width para cualquier tamaño de pantalla o anchos definidos por los breakpoint responsive. Es sensible al dispositivo que utilicemos. Su ancho es determinado por el viewport. Al modificar el tamaño del viewport, el ancho maximo de este contenedor se corresponde con el punto de corte definido.

.container-fluid: establece un width: 100% del viewport en todos los breakpoints. Siempre ocupa el 100% del ancho de la pantalla.

.container- {breakpoint} es similar al container-fluid, pero posee un ancho de 100% solamente hasta el breakpoint definido.

container delimita un ancho máximo de acuerdo al tamaño de la pantalla, row define un grupo horizontal de columnas y col proporciona hasta 12 contenedores por fila.

BOOTSTRAP | Grid

Debemos comprender la estructura anidada de Bootstrap: la clase container es la de mayor jerarquía y actúa como un contenedor de filas (rows), que a su vez contienen a las columnas (col).

Podemos establecer el ancho de una columna y hacer que las columnas hermanas tomen una nueva dimensión automáticamente a su alrededor. Las otras columnas cambiarán de tamaño sin importar el ancho de la columna central.

BOOTSTRAP | Layouts

Podemos tener un layout para cada dispositivo. Si tenemos una pantalla pequeña podemos hacer que se muestre con determinado encolumnado, y en un dispositivo con pantalla mayor que se muestre con otro. De esta manera tendremos distintos tipos de maquetados según el dispositivo, trabajando desde el estilo, no desde la estructura HTML.

BOOTSTRAP | Sistema de grillas

El sistema de grillas de Bootstrap permite hasta 12 columnas en la página. Es posible agrupar las columnas para crear columnas más amplias. Este sistema es responsivo, por lo tanto, las columnas se reorganizan automáticamente dependiendo del tamaño de la pantalla. Recordemos que siempre deben sumar 12.

BOOTSTRAP | Grid system

Este ejemplo crea tres columnas iguales utilizando las clases del sistema grid predefinidas. Dichas columnas serán centradas en la página con el componente padre .container. Las columnas de la cuadrícula que no tengan un width específico se distribuirán automáticamente como columnas de igual ancho.

<div class="container">

<div class="row">

<div class="col-sm">Primer col</div>

<div class="col-sm">Segunda col</div>

<div class="col-sm">Tercer col</div>

</div>

</div>

BOOTSTRAP | Clases receptivas

El sistema de cuadrícula Bootstrap tiene seis clases:

● .col- (dispositivos extra pequeños: ancho inferior a 576 px)

● .col-sm- (dispositivos pequeños: ancho igual o superior a 576 px)

● .col-md- (dispositivos medianos: ancho igual o superior a 768 px)

● .col-lg- (dispositivos grandes: ancho igual o superior a 992 px)

● .col-xl- (dispositivos xlarge: ancho igual o superior a 1200 px)

● .col-xxl- (dispositivos xxlarge: igual o superior a 1400 px)

Las clases anteriores se pueden combinar para crear diseños más dinámicos y flexibles.

Cada clase se escala, por lo que si desea establecer los mismos anchos para sm y md,

sólo necesita especificar sm.

BOOTSTRAP | Componentes

Los componentes de Bootstrap permiten acelerar el proceso de diseño. Son soluciones prediseñadas y personalizables.

Navbar: permite crear una barra de navegación o menú. Viene preparado con el típico icono de hamburger (tres líneas horizontales) que aparece en la versión móvil.

Alerts: son como cajas de texto con cierto tipo de diseño. Se suelen usar para proporcionar información puntual al usuario.

Carousel: Utiliza un sistema de slides para recorrer varios elementos. Permite contener fotografías que van pasando dentro del mismo espacio. Es un componente de presentación de diapositivas.

Dropdowns: Sirven para que el usuario pueda escoger una opción en un conjunto de posibilidades. Genera un menú desplegable hacia abajo o hacia a la derecha que permite incluir vínculos. Con el atributo active se puede marcar alguna opción del menú. Se pueden alternar para mostrar listas de enlaces y más.

Breadcrumb (o migas de pan): Sirven para mostrar la situación del usuario dentro de una página. Indica al usuario dónde está y de dónde viene. Se agregan dentro de la etiqueta semántica <nav>. El atributo active es el que indica en qué página estamos ubicados.

Collapse (accordion): Este elemento añade un botón capaz de ocultar o mostrar cierto contenido, es decir, crear elementos colapsables. Son contenidos que se despliegan y su uso es común en la sección “preguntas frecuentes”.

Buttons: Los botones por defecto son elementos inline, pero de ser necesario un comportamiento similar a inline-block podemos aplicar la clase btn-block.

<button type="button" class="btn btn-primary btn-lg btn-block">Block

level button</button>

Cards: Las cards o tarjetas, sirven para agrupar contenido. Se suelen utilizar para crear listas de elementos, por ejemplo, artículos de blog, colecciones de elementos, etc.

Forms: Bootstrap aplica estilos a los elementos de tipo formulario para convertirlos en elementos responsive, mejorar su apariencia y permitirnos crear diferentes alineaciones.

Tables: Boostrap dispone de distintas clases para dar estilo a las tablas, estas son algunas de ellas:

● .table (por defecto)

● .table-hover

● .table-striped

Progress bar (barras de progreso): Otra herramienta que nos presenta la librería Bootstrap son las barras de progreso. Normalmente se las utiliza para indicar cuánto se ha avanzado en una actividad. Para crear una barra de progreso debemos definir un div con la clase "progress" y un div interno al anterior con la clase "progress-bar".:

Ventanas modales: Son ventanas emergentes que se abren cuando el usuario interactúa con algún elemento. Para funcionar, modal usa los atributos data-toggle con el valor “modal” y data-target con el id del modal que se crea. Para cerrar el modal se usa la etiqueta html de data-dismiss=”modal”. Modal es un contenedor.