CSS

ref. Dorian Desings, Curso básico de CSS. Disponible en Internet, https://www.youtube.com/channel/UCzuwt7Pi_VB8cP5q5UE4u-A. [Consultado octubre de 2020].

Introducción

Es el único lenguaje de estilos que existe, para cualquier web en la que se le quiera dar estilo se utiliza CSS. Sus siglas significa Cascade Style Sheet que traduce hojas de estilos en cascada.

Aunque no es un lenguaje de programación cuenta con funciones propias y variables.

Existen 4 formas para añadir CSS a un documento HTML

1. En el head añadir <style> código CSS </style>
2. En línea con la etiqueta <p style =”código CSS”>
3. Añadir o asociando una hoja de estilos externa al HTML <link rel=”stylesheet” href=”style.css”>
4. Utilizando @import dentro de las etiquetas <style> @import url(“style.css”)</style>

Sintaxis

```
Selector{
propiedad: valor;
}

body{
background: black;
}
```

A este conjunto entero se le llama regla, una regla puede tener tantas modificaciones del selector como se necesiten. Siempre cerrar una modificación con ; (punto y coma).

Tipos de selectores

Selectores básicos y selectores combinadores, son los más utilizados a la hora de realizar modificaciones y aplicar reglas.
Luego están los selectores de pseudo-elementos y selectores de pseudo-clases, se revisarán más adelante.

Selectores básicos

Existen de varios tipos, entre ellos:

- Selectores de etiqueta como por ejemplo:

```
p{
color: red;
}

h1{
font-size: 60px;
}
```

El tema a tener en cuenta aquí es que un selector de etiqueta se aplicará a todos las etiquetas de ese tipo en el documento.

- Selectores de clase: Son los selectores de las clases que se asignan a los elementos de html, por ejemplo:

```
.title{
color: blue;
}

.text{
font-size: 12px;
}
```

- Selectores de id: no se recomienda para dar estilos, se utilizan más para diferenciar componentes y ser utilizados en HTML para las anclas (referencias a diferentes partes del mismo documento) y en JS para identificar específicamente un elemento en el DOM, recalcando que no se debe usar para estilos, un ejemplo sería:

```
#title{
color: blue;
}
```

- Selectores universales: Son selectores que se aplican a todos los elementos del documento, tampoco se recomienda su uso a no ser que se quiera resetear el navegador, un ejemplo sería:

```
*{
color: red;
}
```

- Selectores de atributos: Se selecciona a través de un atributo como href por ejemplo:

```
[href]{
color: green;
}
```

una modificación de este tipo de selector de atributo, sería un selector de atributo-valor que tiene las siguiente variantes.

- Un atributo apuntando a un valor específico:

```
[href=#]{
color: green;
}
```

- Un atributo que tenga un valor al menos una vez

```
[class~="title"]{
      color: red;
}
```

- Un atributo que tenga exactamente el valor indicado o inicie con el valor con guión

```
[class|="text"]{
      color: blue;
}
```

- Un atributo que inicie con un valor indicado:

```
[class ^= "title"]{
      font-size: 60px;
}
```

- Un atributo que finalice con el valor indicado:

```
[class $= "text"]{
      font-size: 12px;
}
```

- Un atributo que contenga el valor:

```
[class *= "title"]
{
      color: blue;
}
```

Selectores combinadores

- Selector de hermano adyacente: selecciona a un hermano justo debajo

```
h1 + h2 {
color: red;
}
```

- Selector de hermano general: buscar todos los hermanos que compartan el mismo padre

```
h1 ~ h3 {
color: green;
}
```

- Selectores descendentes: Aplica el estilo a todos los elementos que sean hijos

```
div span{
color: blue;
}
```

- Selector de hijo directo: Se aplica al hijo directo del elemento seleccionado, el hijo directo es el que está en nivel 1 descendente:

```
p > span{
color: red;
}
```

Herencia

Se obliga a un elemento a heredar la propiedad de su elemento cercano

```
p{
color: red;
}

a{
color: inherit;
}
```

Cascada

La manera como se leen las hojas de estilos y la forma en cómo se sobreescriben los estilos a medida que se baja en el documento.

```
p{
color: red;
}

a{
color: inherit;
}
```

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

Especificidad

Peso de los estilos cuando hay conflicto de estilos

Etiqueta = 1
Clases y pseudo clases = 10
ID = 100
Estilos en línea = 1000
!important = sobre escribe cualquiera anterior

```
/*style.css*/

#parrafo{
color: blueviolet;
}

.parrafo{
color: red;
}

a{
color: inherit;
}

p{
color: green !important;
}

.parrafo-2{
color: purple;
}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Especificidad, herencia y cascada</title>
  <link rel="stylesheet" href="styles-dist.css">
```

```
</head>
<body>

<p>Estamos viendo la <a href="#">herencia</a> en CSS</p>

<p class="parrafo">Estamos viendo la <a href="#">cascada</a> en CSS</p>

<p id="parrafo">Estamos viendo la <a href="#">cascada</a> en CSS</p>

<p class="parrafo-2">El párrafo que queremos cambiar y no podremos</p>
</body>
</html>
```

Conflictos de estilos: important -> especificidad -> cascada

Reescribir estilos de navegadores

Normalize -> https://necolas.github.io/normalize.css/

Metodología BEM

Es una forma de estructurar clases CSS de forma sencilla, escalable y reutilizable, es una de las metodologías más utilizadas en el mundo.
BEM viene de sus siglas Block, Elemento y Modifier:

- Un bloque (Block) es cualquier elemento autónomo y aislado del documento HTML, puede ser un menú, una galería, un formulario, una sección.
  Se nombra con una palabra o una palabra seguida de un guión medio

```
.menu
.menu-principal
```

- Un elemento (Element) es cada uno de los elementos del bloque, en un menú es un enlace, en una galería es una foto, en un formulario es un campo de texto o botón, en una sección puede ser el título.
  Se nombra con el bloque al que pertenece seguido de dos guiones bajos y el nombre del elemento

```
.menu__item
.menu-principal__link
```

- Un modificador (modifier) es cuando un bloque se repite en algún otro lugar de la web pero con modificación como color de texto, tamaño de la fuente, decoración.
  Se nombra con el nombre del elemento seguido de dos guiones medios y el nombre del modificador

```
.menu__item--active
.menu-principal__link--big
```

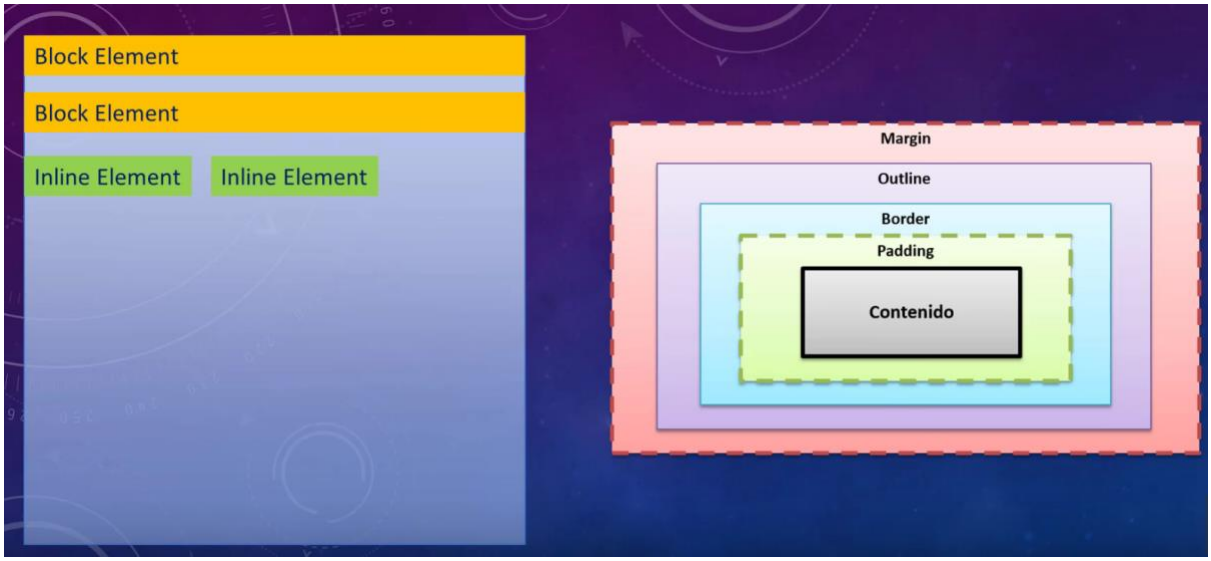
Box-Model

En la web todos los elementos se pueden representar como cajas, una manera de verlo es inspeccionando el código, lo que se resalta al momento de pararse sobre un elemento es un rectángulo, esta forma en la que el navegador dibuja estas cajas se llama el LAYOUT, que hace referencia a la geometría de la web, la posición de un elemento con respecto a.

Las propiedades principales de cada caja es el ancho (width) y el alto (height).

Existen dos tipos de elementos HTML:

- Elementos inline que solo ocupan su contenido y no se puede modificar su ancho o alto.
- Elementos block que ocupan todo el ancho disponible y se les puede asignar un ancho y alto. ViewPort. por defecto el ancho es todo lo que puedan ocupar y el alto es lo que ocupe su contenido



Margin y Padding

Margin es la separación entre una caja y las cajas adyacentes, mientras que el padding es la separación entre una caja y su borde o límite.

```
.Caja-1{
margin: 1em; /* Se aplica a todos los bordes top, right, bottom, left*/
margin: 10px;
}
```

```
.Caja-1{
margin: 1em 1em; /* Se aplica a los ejes X y Y*/
margin: 10px 10px;
}

.Caja-1{
margin: 1em 1em 1em; /* Se aplica al borde top al eje X y al borde
bottom*/
margin: 10px 10px 10px;
}

.Caja-1{
margin: 1em 1em 1em 1em; /* Se aplica a cada uno de los bordes*/
margin: 10px 10px 10px 10px;
}
```

```
.Caja-1{
padding: 1em; /* Se aplica a todos los bordes top, right, bottom, left*/
padding: 10px;
}

.Caja-1{
padding: 1em 1em; /* Se aplica a los ejes X y Y*/
padding: 10px 10px;
}

.Caja-1{
padding: 1em 1em 1em; /* Se aplica al borde top al eje X y al borde
bottom*/
padding: 10px 10px 10px;
}

.Caja-1{
padding: 1em 1em 1em 1em; /* Se aplica a cada uno de los bordes*/
padding: 10px 10px 10px 10px;
}
```

Border

Es la línea que rodea la caja o contenedor, similar al margin y padding es un shorthand dado que el borde se puede representar para cada uno de los lados. Los bordes tienen propiedad Width, Style y Color

```
.Caja-1{
border: 5px solid red;
border-top: none;
}

.Caja-2{
border-left-width: 5px;
border-left-style: dotted;
border-left-color: blue;
}
```

Outline

Es una línea que rodea la caja entre el border y el margin, tiene las propiedades Width, Style, Color y Offset

```
.caja-1{
outline: 5px solid red;
outline-offset: -20px;
}
```

Overflow

Se usa cuando un elemento ocupa más del espacio de su contenedor y no es el comportamiento requerido, sus valores son:

```
.selector{
overflow: hidden /*Para ocultar el contenido sobrante*/
overflow: auto /*para mostrar una barra de scroll donde haga falta*/
overflow: scroll /*para mostrar ambas barras de scroll así no hagan
falta*/
}
```

Float

Han sido reemplazado por flexbox, aún se utiliza en casos concretos como que un texto rodee a una imagen

```
.selector{
float: left /*para que el elemento flotado se ubique a la izquierda*/
float: right /*para que el elemento flotado se ubique a la derecha*/
}
```

y como truco, se utiliza overflow: hidden en el contenedor padre para que contenga a los elementos flotados

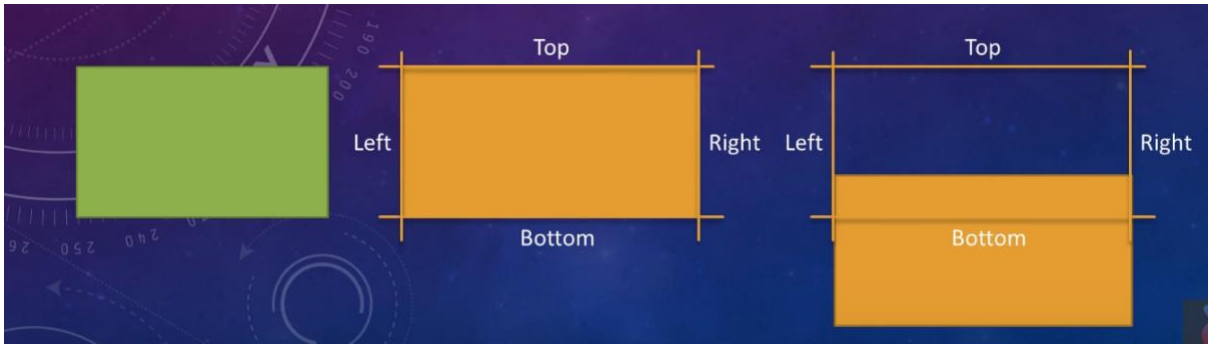
Position

Es la propiedad que permite modificar el flujo del HTML, es decir la manera como visualmente se representan los elementos, los valores de position son: Static, Relative, Absolute, Fixed y Sticky.

Static es el valor por defecto y si no se le indica algún otro valor, no se considera posicionado. Al aplicar otro valor, el elemento se considera posicionado y aparecen 5 nuevos valores que pueden ser modificados, top para mover el elemento con respecto a la parte superior y así para los siguientes 3, right, bottom, left, finalmente se tiene z-index que permite mover el elemento en el eje Z, es decir al frente o atrás.

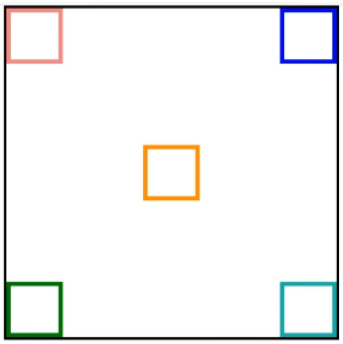
Position relative

Al aplicar este valor de la propiedad la posición del elemento se conserva con respecto al flujo del documento, es decir, no hay un efecto visual, pero como se menciona en el punto anterior, se activan las propiedades indicadas. Tiene mayor peso top y left que right y bottom. Aquí un elemento siempre se va a ocupar o a tener reservado el espacio con el que se definió inicialmente.



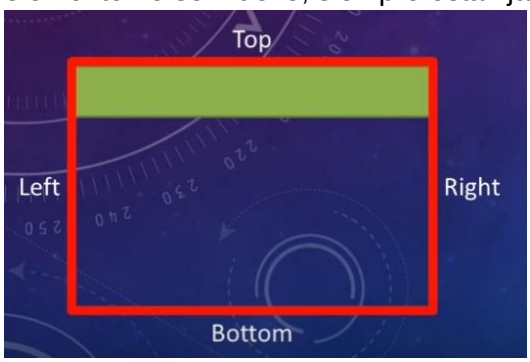
Position Absolute

Este valor hace que el elemento se ubique con respecto a su contenedor posicionado más cercano, si no encuentra ninguno será el viewport o body o espacio disponible de visualización. El elemento no conserva su espacio en el flujo del documento, es decir, es como si no estuviera allí. Al igual que con relative, sus puntos de referencia no cambian y al no tener dimensiones declaradas, se le asignan las que ocupe su contenido.



Position Fixed

Fixed hace que el elemento se coloque con respecto al viewport, cuando se hace scroll ese elemento no se mueve, siempre está fijado.



Position Sticky

Es una combinación entre relative y fixed, tiene las mismas propiedades que relative hasta que llega a un valor top que se le define, desde este punto se comporta como fixed

Z-index

Permite modificar el orden en el que se presentan los elementos en cuanto a profundidad, esta propiedad admite valores negativos y positivos, los negativos se usan para unos casos muy particulares, los valores positivos se asignan no consecutivos como de 10 en 10 o 100 en 100. Si el padre de un elemento tiene asignado un z-index, el hijo del elemento no se podrá posicionar por encima.

Display

Permite definir el comportamiento de una caja con respecto a sus adyacentes, recordando que existen dos tipos de elementos en HTML, inline y block, lo valores de la propiedad son:

- inline: hace que una caja que se comporte como si fuera un elemento en línea.
- Block: con este valor, la caja se comportaría como un elemento en bloque.
- inline-block: permite añadir a un elemento inline las propiedades width y height
- none: oculta el elemento pero todo lo que utilice el elemento se sigue renderizando
- table: imita el comportamiento de una tabla
- list-item: imita el comportamiento de una lista
- flex: Maquetar componentes como un menú
- grid: sistema de grilla

PseudoElementos

Se utilizan para dar estilo a partes específicas de un elemento, no son componentes reales, son una especie de componentes virtuales que solo existen en los estilos del elemento. La sintaxis de un pseudo elemento es `Selector::pseudo-elemento{ estilos }`

Existen 5 pseudoelementos básicamente:

- ::first-line -> Solo se aplica para elementos en bloque y lo que hace es añadir un comportamiento o estilo a la primera línea del elemento que se indique.
- ::first-letter -> Funciona igual que first-line solo que aquí se aplica para únicamente la primera letra.
- ::before -> añade estilos o contenido justo antes del componente indicado, es obligatorio el uso de la propiedad content, puede ser inicializada en vacío.
- ::after -> añade estilos o contenido justo después del componente indicado, es obligatorio el uso de la propiedad content, puede ser inicializada en vacío.
- ::selection -> añade estilo al momento de seleccionar una parte o todo el elemento.

PseudoClases

Son selectores que reaccionan en tiempo real, principalmente cuando un usuario interactúa con alguno de los elementos, una característica es que funcionan con cualquier tipo de selector e incluso sin ninguno.

```
selector:pseudoclase{
estilos
}

::pseudoclase{
estilos
}
```

Indice de las pseudo-clases estándar

:active	:indeterminate	:only-child
:checked	:in-range	:only-of-type
:default	:invalid	:optional
:dir()	:lang()	:out-of-range
:disabled	:last-child	:read-only
:empty	:last-of-type	:read-write
:enabled	:left	:required
:first	:link	:right
:first-child	:not()	:root
:first-of-type	:nth-child()	:scope
:fullscreen	:nth-last-child()	:target
:focus	:nth-last-of-type()	:valid
:hover	:nth-of-type()	:visited

Fuente: MDN web docs, Pseudo-classes, https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes

Según tipos se pueden dividir en:
Dinámicas:

- link: un enlace no visitado
- visited: un enlace visitado
- active: al momento de dar clic sobre el elemento
- hover: cuando se posiciona el mouse sobre el elemento
- focus: cuando un elemento tiene el foco o está seleccionado, por ejemplo un input

Objetivo:

- target: se aplica cuando un elemento es un marcador o ancla.

Lenguaje

- lang(): Elemento que contiene el identificador de idioma que se le indica, es una función.

UI de estado

- Enable: campo de formulario que estén activados, enable es el valor por defecto de todo input
- Disable: se aplica a los elementos que estén desactivados.
- Checked: se aplica cuando un elemento es marcado o está checked.
- in-range: Se aplica cuando el elemento se encuentra dentro del rango
- out-of-range: se aplica cuando el valor en el elemento se encuentra fuera del rango
- optional: Por defecto todos los campos son opciones, se pueden aplicar efectos a dichos campos.
- required: Aplicar un estilo a los elementos que se marcan como obligatorios.
- valid: aplicar a un elemento cuando el valor del campo es válido
- Invalid: Aplicar estilo cuando el valor del elemento no es válido.

Estructurales

- root: representa la raíz del documento, se le da estilos al HTML
- empty: estilos a los elemento que no tengan contenido
- child: estilos sobre los hijos
    - first-child: primer hijo
    - last-child: último hijo
    - nth-child(): selecciona al n-ésimo hijo
    - nth-last-child(): Selecciona el n-ésimo hijo pero iniciando desde el último
    - only-child: funciona solo cuando hay un único hijo.
        - odd: impares
        - even: pares
        - números enteros
        - ecuaciones

Type

- Al igual que con los child para un tipo de etiqueta o una clase definida

negación

- not(): se aplica a todos los elementos excepto lo que se indique en el not.

Variables CSS

Una variable es un espacio en memoria en donde almacenamos un color para poder utilizarlo o modificarlo.
Las variables en CSS necesitan estar dentro de un selector, tener en cuenta el :root.
Tienen herencia y cascada
Existen variables globales y locales

```
/* Para declararla */
:root{
   --nombre-variable: valor;
}

/* Para utilizarla */
.selector{
   propiedad: var(--nombre-variable)
}
```

Background

Explicación background-color, image, attachment (efecto parallax), clip, origin, repeat, position, origin, size multiple

Textos y tipografías

Son los tipos de letra que existen, se dividen en dos grupos

-Familias tipográficas, tienen nombres propios como Arial, Times New Roman, Verdana, Calibri.

- Familias genéricas o seguras, son grupos asociadas por sus características, Serif, sans serif, cursive…

Se recomienda usar fuentes de respaldo

Los modificadores de font son:

- -fammily
- -size
- -weight
- -style

Los modificadores de bloques de text son (aplicar al contenedor del texto):

- -transform
- -align
- -decoration
- -indent

- line-height (truco de centrado)
- letter-spacing

Google fonts

Utilización de google fonts para títulos y párrafos, mencionar también que se pueden descargar fuentes de www.dafont.com/es/ y www.1001fonts.com para importar fuentes descargadas:

```
@font-face{
src: url('./path-to-font/');
font-family: 'the_name_of_my_font';
}
```

Listas

Las listas no ordenadas tienen algunas propiedades como por ejemplo modificar la viñetas de los elementos:

```
.list{
list-style-type: disc, circle, decima, decimal-leading-zero, y algunos
alfabetos;
}
```

Reseteo de la lista para personalizarlas o para hacer menús

```
.list{
margin-top: 0;
margin-bottom: 0;
padding-left: 0;
list-style: 0;
}
```

Tablas

Una tabla tiene su propio layout o su propio formato, hay cosas que no se pueden manejar como en CSS normal.

table-layout:

- automatic: comportamiento por defecto, ocupa lo que puede y colapsa, las columnas con más contenido son más anchas.
- fixed: se establece un width para dividir todas las columnas con un mismo ancho

caption-side: define en donde se coloca el caption de la página web

- top: se posiciona en la parte superior
- bottom: se posiciona en la parte inferior.

border-spacing: controla el espacio entre las celdas

border-collapse: la presentación de las celdas, si son seguidas o no
- spacing: las celdas son separadas y puede modificarse su separación con la propiedad anterior
- collapse: las celdas van juntas similar a como se visualiza en excel.

empty-cells: controla qué hacer con las celdas vacías
- hide: esconde las celdas vacías

Imágenes

- Una imagen responsive requiere una sola propiedad

```
.selector{
max-width: 100%;
}
```

- Dado que las imágenes son elementos inline, es decir, ocupan solo su espacio pero cuenta con un line-height que hace que exista un espacio por debajo en el contenedor de la imagen, se puede corregir utilizando la propiedad display:block en la imagen o el line-height del contenedor a 0.

```
.img{
display: block;
}
.img-container{
line-height: 0;
}
```

- Centrado de las imágenes: lo mejor aquí es cambiar el display de la imagen a block y darle margen auto a ambos lados, así va a estar centrada horizontalmente sin problema, sea cual sea su espacio.

```
.img{
display: block;
margin-left: auto;
margin-right: auto;
}
```

- Centrado vertical: Se puede utilizar flex-blox en display: flex y se pondría align-items: center, pero algunos navegadores viejos no lo soportan, para esos se requeriría hacer display: table para el contenedor y display: table-cell para sus elementos, adicional agregar vertical-align: middle en los elementos.

Otros aspectos importantes a la hora de trabajar con imágenes es object-fit para ajustar la imagen a una ventana o contenedor sin deformar la imagen y object-position que define qué porción o desde que sección de la imagen se va a visualizar.

También se tiene filter, se tiene valores como blur(px) para desenfoque, brightness(0-1 quitar brillo, 1 en adelante sobre-exponer) brillo de la imagen, contrast(%), drop-shadow(h-shadow v-shadow desenfoque y color) para dar sombra al exterior, grayscale(0-1), hue-rotate(deg), invert(0-1) para invertir los colores de la foto, opacity(0-1) para transparencia, saturate(0-1) saturación de la imagen, sepia(0-1).

y para máscaras se utiliza clip-path tiene opciones como circle, ellipse, inset y poligon, recomendación usar https://bennettfeely.com/clippy/

Colores

Plugin para VS Code color highlight

En CSS existen diferentes formas de añadir colores a un fondo o un texto, algunas permiten una manipulación leve del color, como su tonalidad y algunas permiten manipular otros aspectos más avanzados. Las opciones que se tienen en CSS para definir un color son:

- Color name

```
.selector{
color: red;
}
```

- RGB y RGBA

```
.selector{
/*Se crean los colores definiendo su porcentaje de Rojo, Verde y Azul*/
color: rgb(0, 0, 0);
/*Igual al anterior pero el valor final define la transparencia del color*/
color: rgba(0, 0, 0, 1);
}
```

- Hexa

```
.selector{
/*rgb*/
color: #fff;
/*rrggbb*/
color: #ffffff;
}
```

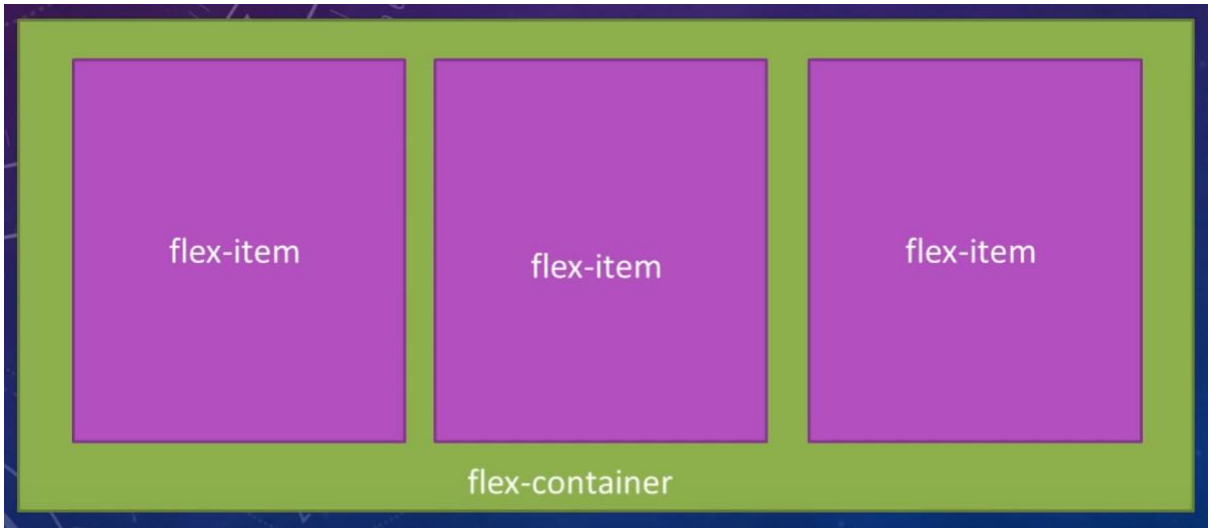
- HSL y HSLA

```
.selector{
/*Color o ángulo en la rueda cromática, saturación, luminosidad*/
color: hsl(120, 100%, 50%);
```

```
/*un valor adicional que indica la transparencia*/
color: hsla(120, 100%, 50%, 1);
}
```

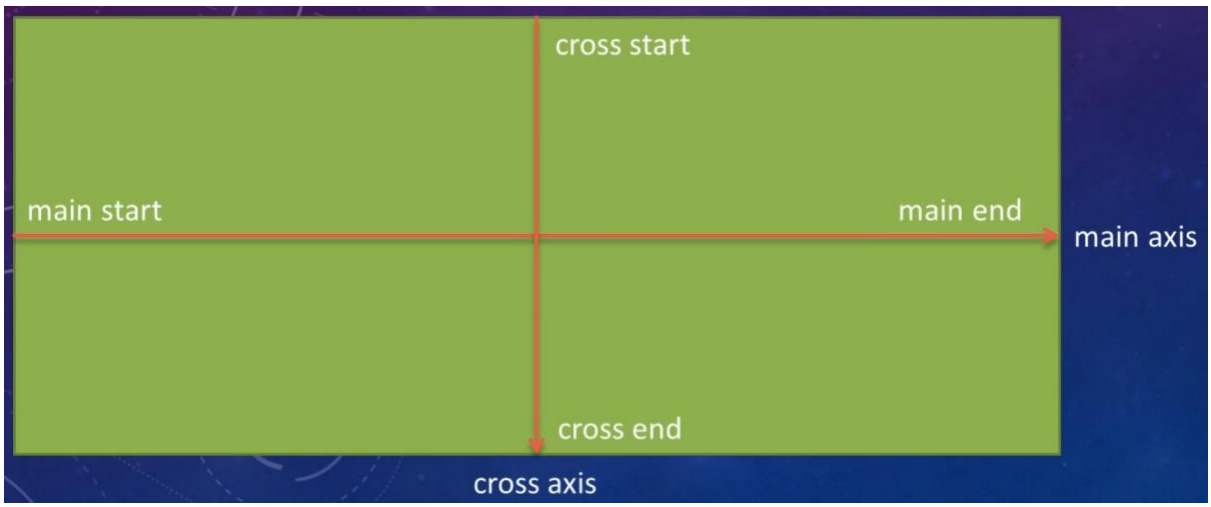
FlexBox

Es un modelo de layout que permite que las cajas sean flexibles, es un valor de la propiedad display y existen display: flex para que el contenedor sea de caja o display: inline:flex para que sea de línea.
Para implementar flex, es necesario un contenedor o flex-container, que sería el contexto de los elementos flex y, adicional, al menos un hijo que será el flex-item.



Solo serán flexibles los hijos directos del contenedor a no ser que se le indique que un hijo es un contenedor flex a la vez.

Al utilizar esta propiedad, se activan dos ejes para colocar y alinear elementos, que por defecto se comportan de la siguiente manera:



Flex-direction para modificar la orientación de los ejes y flex-wrap para controlar el salto de línea de los elementos.

```
/* La dirección del texto se presenta en una línea */
flex-direction: row;

/* Como <row>, pero al revés */
flex-direction: row-reverse;

/* La dirección en la que se apilas las líneas de texto */
flex-direction: column;

/* Como <column>, pero al revés */
flex-direction: column-reverse;
```

```
flex-wrap: nowrap
flex-wrap: wrap
flex-wrap: wrap-reverse
```

Cuenta con algunas propiedades para marcar en qué punto un elemento se hace flexible, es decir, cuánto ocupa antes de hacer un salto o desbordarse

```
/* Tres valores: flex-grow | flex-shrink | flex-basis */
flex: 2 2 10%;
```

flex-grow: Esta propiedad indica que tanto del espacio restante en el contenedor debe ser asignado al ítem o elemento.

flex-shrink: Esta propiedad setea el factor de encogimiento del ítem o elemento flex, si la suma de todos los elementos supera el tamaño del contenedor, la propiedad activada permitirá que los elementos se encojan y se ajusten.

flex-basis: Esta propiedad define el tamaño inicial de un elemento.

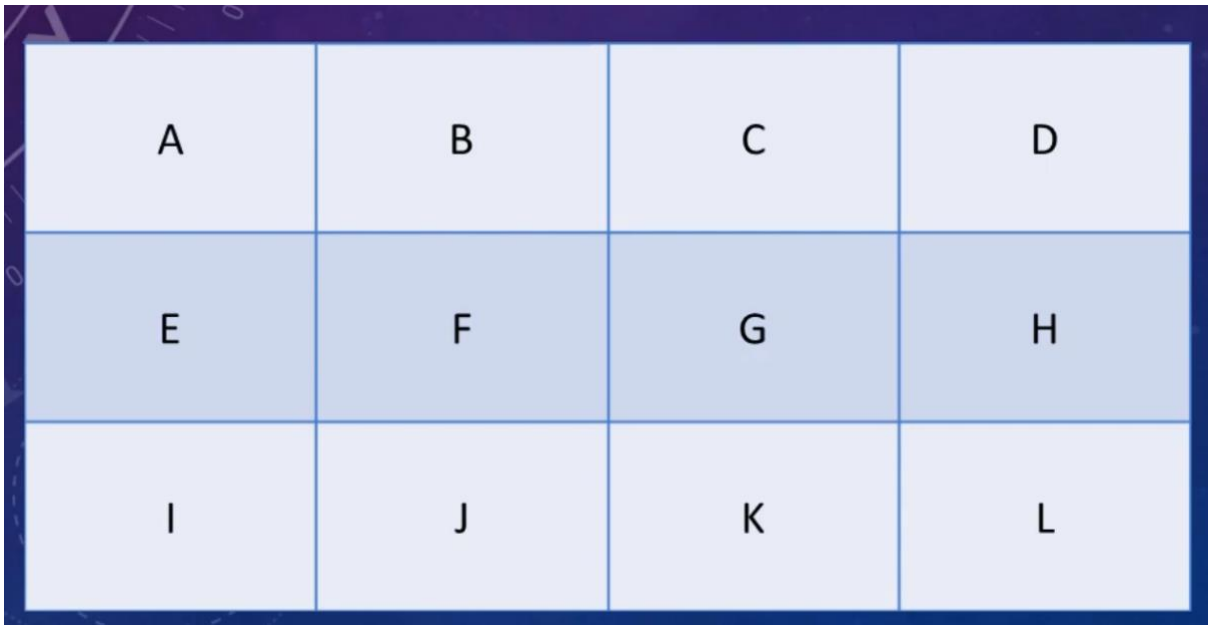
```
/* Specify <'width'> */
flex-basis: 10em;
flex-basis: 3px;
flex-basis: auto;

/* Intrinsic sizing keywords */
flex-basis: fill;
flex-basis: max-content;
flex-basis: min-content;
flex-basis: fit-content;
```

```
/* Automatically size based on the flex item's content */
flex-basis: content;
```

Grid

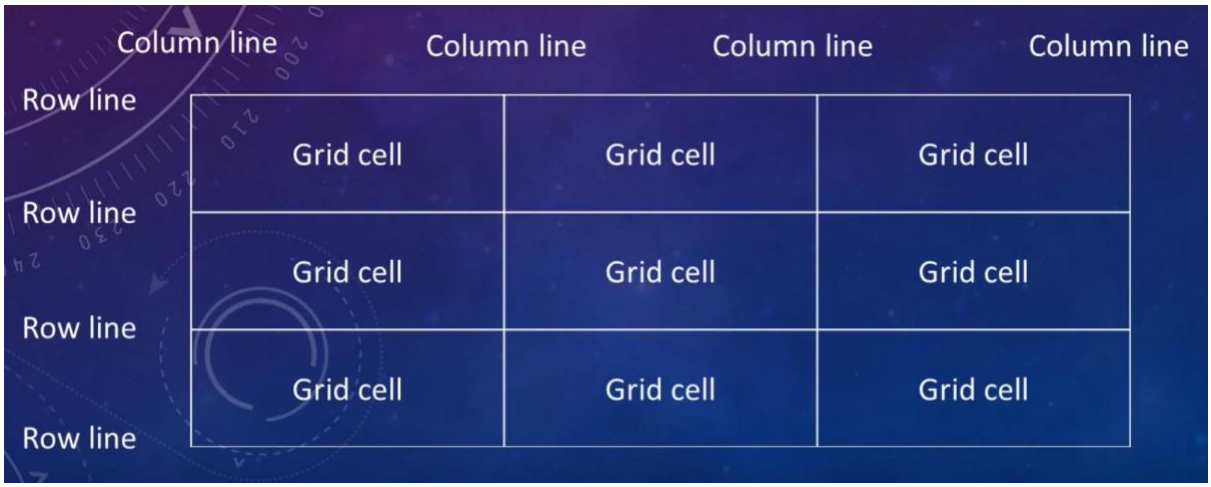
Modelo de layout que permite construir grillas o cuadrículas dinámicas, funcione igual que flex.



Fuente: Dorian Desings, **Grid - Fundamentos - Curso básico de CSS 2019**

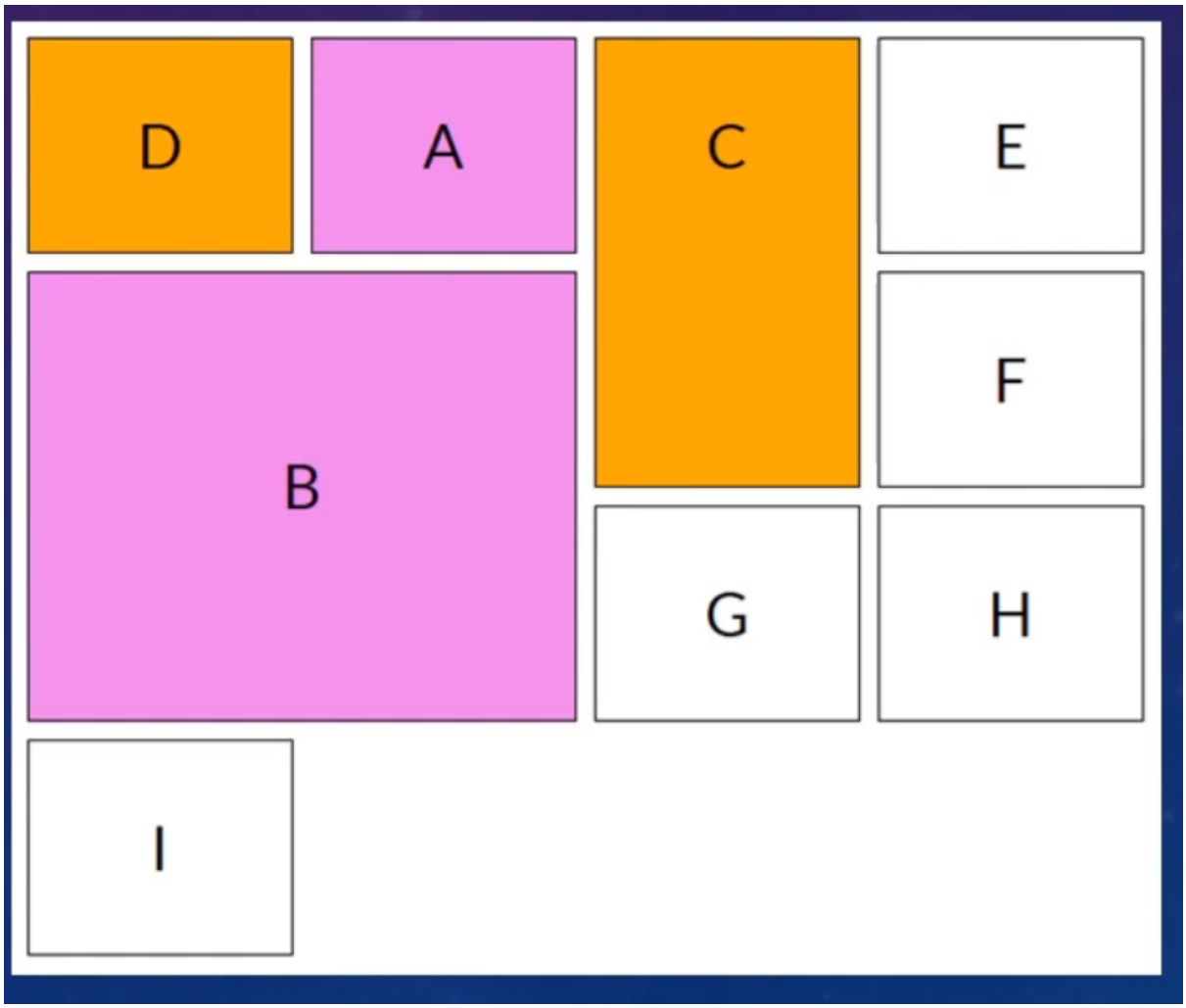
Se compone de:

- grid column: cada columna de la cuadrícula
- grid row: cada fila de la cuadrícula
- grid-cell: cada celda de la cuadrícula
- grid gap: separación entre las celdas
- grid line: las líneas que delimitan las filas y columnas



Fuente: Dorian Desings, **Grid - Fundamentos - Curso básico de CSS 2019**

En grid cada celda es dinámica y el resto de la cuadrícula se adapta, se puede indicar en dónde empieza y termina cada celda, especificando celda y columna



Fuente: Dorian Desings, **Grid - Fundamentos - Curso básico de CSS 2019**