CSS 3

HOJAS DE ESTILO EN CASACADA

CSS 3

Selectores Avanzados - *Selector de hijos*

">" Este selector, permite seleccionar por herencia directa, es decir, que el elemento de la derecha al signo > tiene que ser hijo directo del elemento de la izquierda.

Sintaxis:

```
elemento1 > elemento2 {
        propiedad: valor;
}
```

CSS 3

Selectores Avanzados - *Selector de hijos*

Ejemplo:

```
<main><section>
        <h2>Selector de hijos: </h2>
</section></main>
```

```
main > section > h2{
            color: #606bf2;
}
```

Seleccionará sólo los h2, que son hijos directos de un section, quien a su vez, es hijo directo de un main.

CSS 3

Selector adyacente: +

Sirve para seleccionar elementos que están dentro de un mismo elemento contenedor y están seguidos en el código html.

Sintaxis:

```
elemento1 + elemento2 {
        propiedad: valor;
}
```

CSS 3

Selector adyacente: +

Sirve para seleccionar elementos que están dentro de un mismo elemento contenedor y están seguidos en el código html.

Ejemplo:

```
<article>
        <h2>Selector adyacente: + </h2>
        <p>Párrafo 1</p>
        <p>Párrafo 2</p>
</article>

h2 + p {
   text-transform: uppercase;
      }
```

para el ejemplo anterior, seleccionará solo el primer h2, ya que el segundo no se encuentra contiguo

CSS 3

Selector general de hermano: ~

Sirve para seleccionar elementos que están dentro de un mismo elemento contenedor. A diferencia del adyacente, **no** necesitan estar seguidos.

Sintaxis:

```
elemento1 ~ elemento2 {
  propiedad: valor;
}
```

CSS 3

Selector general de hermano: ~
Sirve para seleccionar elementos que están dentro de un mismo elemento contenedor. A diferencia del adyacente, **no** necesitan estar seguidos.

Ejemplo:

```
El selector:
h2 ~ p {
    color: #5c5c5c;
}
```

para el ejemplo anterior, seleccionará ambos párrafos.

CSS 3

Ejemplo de Pseudoclases

Una pseudoclase es una palabra clave que se añade a los selectores y que especifica un estado espeifíco del elemento seleccionado.

Sintaxis:

```
selector:pseudoclase {
    propiedad: valor;
              }
```

CSS 3

Pseudo-Clase :nth-child()
La pseudo – clase :nth-child() nos permite seleccionar un elemento específico, en forma numérica.

Por ejemplo: p:nth-child(1) seleccionar la primer etiqueta p de nuestro documento.

```
li:nth-child(2){
     color: #716eb1;
}
```

CSS 3

Pseudo-Clase :nth-child()
La pseudo – clase :nth-child() nos permite seleccionar un elemento específico, en forma numérica.

Además de los valores númericos, podemos utilizar los valores “even” y “odd”. El valor even, selecciona todos los elementos impares y el valor odd, todos los pares:

Por ejemplo:

```
li:nth-child(even){
    background-color:#ebeaea;
}
li:nth-child(odd){
    background-color: #d4d2ff;
}
```

CSS 3

Pseudo-Clase :focus()

Permite darle estilo visual a los input cuando están en foco.

```
input:focus{
     border: 2px solid #ff0404;
 }
```

CSS 3

Pseudo-Clase :first-child()

Selecciona el primer elemento de nuestro documento:

```
p:first-child{
    text-align: center;
}
```

CSS 3

Pseudo-Clase :last-child()

Selecciona el último elemento de nuestro documento:

```
p:last-child{
     font-weight: bold;
}
```

CSS 3

Pseudo-elementos CSS

Los pseudo-elementos, nos permiten seleccionar internamente, una fracción dentro de una etiqueta de contenido. Como por ejemplo, seleccionar la primera letra de un párrafo, o la primera línea, o bien agregar contenido antes o después de la etiqueta.

Sintaxis:

```
selector::pseudoelemento {
  propiedad: valor;
}
```

CSS 3

Pseudo-elemento ::first-letter
Permite seleccionar la primera letra, de la primera línea de texto de un elemento.

Ejemplo:

```
p::first-letter{
    font-size: 2em;
    font-weight: bold;
 }
```

CSS 3

Pseudo-elemento ::first-line
Permite seleccionar la primera línea de texto de un elemento.

Ejemplo:

```
p::first-line{
 font-style: italic;
}
```

CSS 3

Pseudo-elemento ::before y ::after

Los pseudo-elementos before y after, se usan para agregar contenido antes o después de una etiqueta, en combinación con la propiedad content de css.

Ejemplo:

```
blockquote::before{
    content: "'";
    font-size: 2em;
    font-weight: bold;
}
 blockquote::after{
    content: "'";
    font-size: 2em;
    font-weight: bold;
}
```

CSS 3

Flexbox

Flexbox es un módulo de css, para crear diseños flexibles.
Establece una serie de propiedades que nos permiten posicionar los elementos HTML, tanto horizontal como verticalmente con mayor facilidad que con los flotados.
Requiere que establezcamos un elemento contenedor y dentro de ese elemento contenedor, una serie de ítems que se adaptan, ajustando sus tamaños y disposición en función del espacio disponible de la caja contenedora.
Podemos incluso, desde el css, alterar el orden de los elementos y modificar la forma en que los elementos se ubican en pantalla (sin modificar el html).

CSS 3

Definir flexbox:
Para poder utilizar flexbox, necesitamos crear un elemento contenedor (padre) y una serie de elementos hijos. Como el siguiente ejemplo:

```
<section class="contenedor">
	<article class="item1">1</article>
		<article class="item2">2</article>
	<article class="item3">3</article>
</section>
```

Al elemento contenedor le agregamos la propiedad display, con el valor flex, que establece que se trata de una caja flexible.

```
.contenedor{
	display: flex;
}
```

CSS 3

Propiedades para contenedores:

1.- flex-direction

Esta propiedad establece de qué forma los elementos hijos del contenedor, se van a posicionar.

```
.contenedor{
    flex-direction: row;
}
```

Tiene 4 valores posibles:

CSS 3

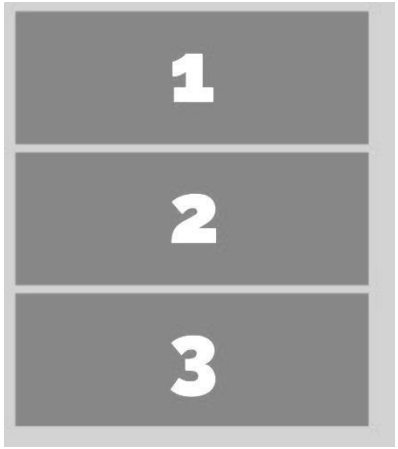
•**flex-direction: row;**
Ordena los elementos horizontalmente de izquierda a derecha en una fila.

•**flex-direction: row-reverse;**
Ordena los elementos horizontalmente de derecha a izquierda en una fila e invierte el orden de los mismos.

CSS 3

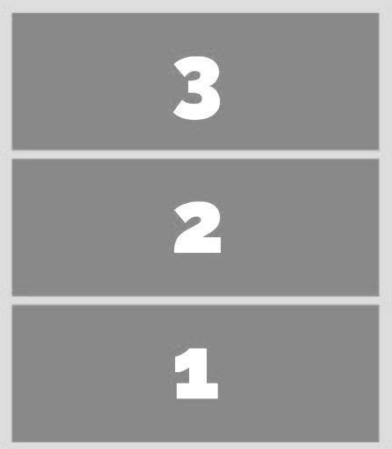
- **flex-direction:column;**

Ordena los elementos verticalmente de arriba abajo en una columna.



CSS 3

•**flex-direction:column-reverse;**
Ordena los elementos verticalmente de abajo arriba en una columna.



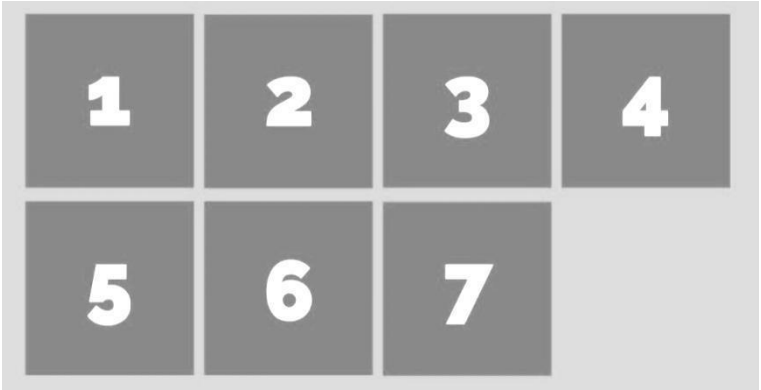
CSS 3

2.- flex-wrap
Redistribuye los elementos para que entren una o más líneas.\

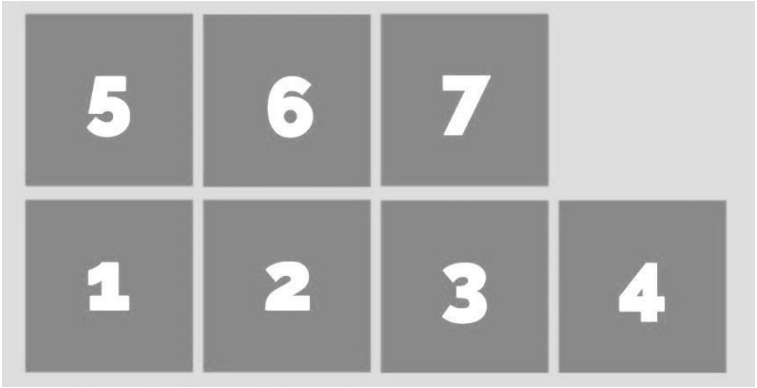
•**flex-wrap: no-wrap;**
Redistribuye los elementos para que entren en una sola línea, achicándolos si es necesario.

CSS 3

•**flex-wrap: wrap;**
Redistribuye los elementos en varias líneas de izquierda a derecha y de arriba a abajo.



•**flex-wrap: wrap-reverse;**
Redistribuye los elementos en varias líneas de derecha a izquierda y de abajo a arriba.



CSS 3

3.- justify-content
Distribuye la posición de los elementos y la disposición que tendrán en el eje horizontal.

•**Justify-content: flex-start;**
Este es el valor por defecto, alinea los elementos a la izquierda.

•**Justify-content: flex-end;**
Alinea los elementos a la derecha.

CSS 3

- **Justify-content: center;**
  Centra horizontalmente los elementos.

- **Justify-content: space-around;**
  Distribuye los elementos, dejando el mismo espaciado entre ellos.

- **Justify-content: space-between;**
  Distribuye los elementos ubicando el primero pegado a la izquierda del contenedor (sin espacio hacia la izquierda) y el último sin espacio a la derecha.

CSS 3

4.- align-items
Distribuye la posición de los elementos y la disposición que tendrán en el eje vertical.

•**align-items: stretch;**
Los elementos hijos, ocupan todo el ancho (o alto) del elemento contenedor.

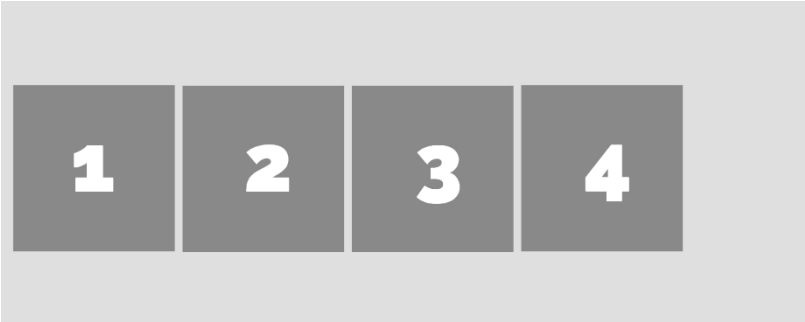
CSS 3

•**align-items: flex-start;**
Los elementos se ubican desde el inicio del borde superior del contenedor.

•**align-items: flex-end;**
Los elementos se ubican desde el inicio del borde inferior del contenedor.

CSS 3

•**align-items: center;**
Los elementos se ubican centrados verticalmente.

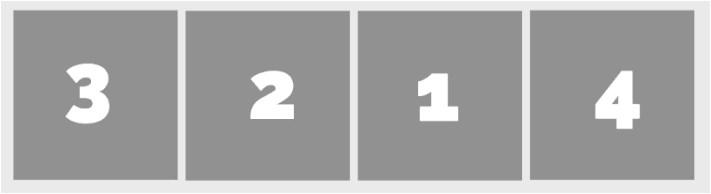


•**align-items: baseline;**
Alinea los elementos tomando como línea base la altura del texto.

CSS 3

Propiedades para items:
1.-order
Con esta propiedad, podemos cambiar el orden de cualquier elemento hijo. Por ejemplo:

```
.item1{
    order:3;
}
```



CSS 3

2.-flex-grow

Es el crecimiento que va a tener un elemento en relación a los demás elementos hermanos. El valor por defecto es 1, si algún elemento tiene un valor de flex-grow mayor, el resto de los elementos, se re distribuyen en ancho.

Por ejemplo:

```
.item2{
     flex-grow:2;
}
```

CSS 3

3.-flex-shrink
Esto define la posibilidad de que un elemento flexible se contraiga si es necesario.

4.-flex-basis
Sirve para establecer un valor inicial de ancho o de alto de los elementos, previo a que ese ancho o alto sea modificado por las otras propiedades.

```
.item1{
     flex-basis:30em;
}
```

5.-align-self
Esta propiedad permite alinear un elemento en forma individual.
Los valores posible son:

```
auto | flex-start | flex-end | center | baseline | stretch
```

CSS 3

3.-flex-shrink
Esto define la posibilidad de que un elemento flexible se contraiga si es necesario.

4.-flex-basis
Sirve para establecer un valor inicial de ancho o de alto de los elementos, previo a que ese ancho o alto sea modificado por las otras propiedades.

```
.item1{
     flex-basis:30em;
}
```

5.-align-self
Esta propiedad permite alinear un elemento en forma individual.
Los valores posible son:

```
auto | flex-start | flex-end | center | baseline | stretch
```

Referencias

Compatibilidad con navegadores:

Como flexbox aún no es compatible con todos los navegadores, debemos agregar los prefijos -ms- , -webkit- a las propiedades de flexbox.
Podemos ver qué navegadores son compatibles en el sitio Can I Use:
http://caniuse.com/#search=flexbox

Web para probar propiedades de flexbox:

https://demos.scotch.io/visual-guide-to-css3-flexbox-flexbox-playground/demos/

CSS 3

CSS3, nos brinda dos posibilidades a la hora de hacer animaciones web: transiciones y animaciones.

Transiciones CSS

Las transiciones permiten animar los cambios de las propiedades CSS, cuando un elemento cambia de estado y que ese cambio no sea brusco, sino que suceda en un intervalo de tiempo en forma fluida. Por ejemplo, si tenemos un cambio en las propiedades de un vínculo en su estado normal y su estado hover:

CSS 3

```
a{
    color: #fff;
    background-color:#66ccff;
}
a:hover{
    background-color:#ff8d35;
}
```

Esto produciría un cambio brusco en el color de fondo del vínculo, que pasaría de tener un color a otro al pasar el mouse.

CSS 3

Para que este cambio sea fluido, podemos agregarle una transición:

```
a{
    color: #fff;
    background-color:#66ccff;
    transition: all 2s;
}
a:hover{
    background-color:#ff8d35;
}
```

Ahora el cambio, queda animado y esa animación dura 2 segundos.

CSS 3

Para que una transición se dispare necesitamos que haya un cambio de estado en el elemento html, que puede ser:

```
:hover
:focus
:visited
:linked
```

Las propiedades que tienen las transiciones son:
transition-property: sirve para indiciar sobre qué propiedad se aplicará la transición. Se pueden especificar varias propiedades separándolas por comas. El valor por defecto es all que indica que la transición se aplicará sobre todas las propiedades que se modifiquen entre un estado y otro.

transition-duration: indica el tiempo de la transición, se expresa en segundos o decimas de segundos separados por puntos. Por ejemplo: 0.5s
El valor por defecto es 0.

CSS 3

transition-delay: podemos establecer un tiempo de espera (delay) para que comience la transición, una vez que el elemento haya cambiado de estado. Ese tiempo, lo agregaríamos con esta propiedad.

transition-timing-function: indica cómo será la progresión de la transición entre un estado y otro.

El valor por defecto es ease, que lo que hace es suavizar ambos extremos de la transición.

linear: este tipo de progresión es una progresión mecánica,
ease o ease-in-out: la transición comienza y acaba lenta.
ease-in: la transición comienza lenta y luego va más rápido. Representa una aceleración
ease-out: la transición comienza rápida y termina lenta. Representa una desaceleración.
cubic-bezier(n,n,n,n): es un tipo de progresión personalizada.

transition: propiedad abreviada que permite unificar todas las propiedades anteriores en una sola.

CSS 3

El valor por defecto es ease, que lo que hace es suavizar ambos extremos de la transición.

linear: este tipo de progresión es una progresión mecánica,
ease o ease-in-out: la transición comienza y acaba lenta.
ease-in: la transición comienza lenta y luego va más rápido. Representa una aceleración
ease-out: la transición comienza rápida y termina lenta. Representa una desaceleración.
cubic-bezier(n,n,n,n): es un tipo de progresión personalizada.

transition: propiedad abreviada que permite unificar todas las propiedades anteriores en una sola.
Un ejemplo sería transition: all 2s;

CSS 3

Animaciones CSS
A diferencia de las transiciones, las animaciones, no dependen de los estados de los elementos HTML para comenzar.

Pueden tener varias posiciones o cuadros claves en el movimiento, denominados keyframes y por ende nos brindan mayor precisión a la hora de hacer una animación más compleja.

Lo primero que tenemos que hacer es crear la animación y establecer los cuadros claves. La animación la declaramos con @keyframes y seguido va el nombre que le asignemos a la animación.

Cada porcentaje representa un cuadro clave. Puedo establecer ilimitados cuadros claves en una animación.

CSS 3

```
@keyframes nombre-animacion {
	0% {
		transform: translateX(0px);
	}

	50% {
		transform: translateX(400px);
	}

	100% {
		transform: translateX(0px);
	}
}
```

CSS 3

Una vez creada la animación, podemos invocarla desde cualquier selector css y nuestro elemento html quedará animado:

```
.circulo{
	animation-name: nombre-animacion;
	animation-timing-function: ease;
	animation-duration: 1s;
	animation-interation-count: infinite;

}
```

En este ejemplo, los elementos de clase circulo quedaran animados moviéndose a derecha y volviendo a la posición inicial. La animación durará 1 segundo y se reproducirá en forma infinita.

CSS 3

Propiedades de las animaciones:
Las propiedades de las animaciones son bastante parecidas a las de las transiciones. Para poder aplicar animaciones sobre nuestros elementos contamos con las siguientes propiedades:
animation-name: este es el nombre que le hayamos puesto a la animación, cuando establecimos los keyframes. La forma de invocarla es a través de su nombre.

animation-duration: el tiempo que tardará la animación. Se expresa en segundos.

CSS 3

animation-timing-function: indica la curva de progresión de la animación (igual que las transiciones).

animation-iteration-count: indica el número de veces que se repetirá la animación. Su valor por defecto es 1. Podemos incrementarlo o bien si queremos que se reproduzca indefinidamente, establecer el valor “infinite”

animation-delay: indica el retardo con el que se iniciará la animación. Por defecto es 0.
animation: forma abreviada, permite unificar las propiedades anteriores en una sola.
Por ejemplo:
animation: nombre-animacion 3s ease 0.5s infinite;

CSS 3

Recursos:
Animate Css
https://daneden.github.io/animate.css/
Es una galería de animaciones css, que podemos utilizar para animar cualquier elemento.
Simplemente, tenemos que vincular el css a nuestra página y agregarle al elemento, la clase con la animación que querramos, más la clase animated, que indica que queremos que ese elemento se anime.

Wow js
https://github.com/matthieua/WOW
Esta librería js detecta el scroll del navegador y puede combinarse con facilidad con anímate.css, disparando las animaciones en el momento que el usuario baja con el scroll por esa sección.

Referencias

Referencias:

W3 Schools Css3 Selectors Reference
https://www.w3schools.com/cssref/css_selectors.asp

W3 Schools Css3 Flexbox
https://www.w3schools.com/css/css3_flexbox.asp
A visual Guide to flexbox properties
Autor: Dimitar Stojanov
https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties

Css-Tricks- A Complete Guide to Flexbox
Autor: Chris Coyier
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Understanding Flexbox: Everything you need to know

Autor: Ohans Emmanuel
https://medium.freecodecamp.com/understanding-flexbox-everything-you-need-to-know-b4013d4dc9af#.mmfc1uaov

Referencias

Referencias:

W3 Schools Css3 - Animations
https://www.w3schools.com/css/css3_animations.asp