<codoacodo/>

FULL STACK JAVA
Clase

CSS (3)

BA

Temas

Pseudoclases
Prefijos de CSS
Transiciones, animaciones y transformaciones
Animate.css
Responsive Web Design, Conceptos y práctica
Trabajo Práctico. Web Responsive.

Pseudoclases

Pseudoclases

Son clases que se activan según una actividad o un estado que sufren en la página, puede ser que uno se sitúe encima de ellas, o haga foco en un elemento, o el elemento se encuentre en determinada posición respecto a su padre, etc.

Podemos ver la extensa lista de ellas aquí

Pseudoelementos

Cuando pasamos sobre un elemento se activa el pseudoelemento **:hover**.

Cuando nos ubicamos en un elemento (como por ejemplo un input) se activa el pseudoelemento **:focus**.

El primer hijo de un contenedor tiene como pseudoelemento **:first-child**, el último **:last-child**, el último de un determinado tipo **:last-of-type**, podemos agarrar un hijo determinado con **:nth-child(n)** haciendo referencia con 'n' al número de hijo al que queremos elegir.

Prefijos

Prefijos

No todos los exploradores trabajan con las mismas tecnologías, cada uno tiene su forma de interpretar nuestro código css, por ello que no todos los atributos están disponibles de la misma manera.

Algunos sí lo están, pero al trabajar con su propia tecnología para renderizarlos quizás necesitan de un prefijo para interpretarlo y adaptarlo en sus renderizados.

Entre tantas páginas que nos mantienen informado podemos ver StateOfCss, que realiza cada año encuestas y hace referencias a los atributos (y herramientas) que aparecen o se empiezan a usar en CSS.

Prefijos

Los veremos de manera frecuente, podemos encontrarlos de la siguiente manera:

-webkit- (utilizada por Chrome, safari y otros navegadores basados en WebKit)
-moz- (Firefox)
-o- (Opera)
-ms- (internet Explorer y Edge)

En VSCode disponemos de la extensión css-auto-prefix, que creará estos prefijos de manera automática cuando utilicemos algún atributo que no tenga una compatibilidad general.

Transición, animaciones y transformaciones

Transición, animaciones y transformaciones

Transiciones: El atributo **transition** permite otorgar un tiempo (***transition-duration***), especificar el o los atributos a modificar (***transition-property***), el retardo de reacción (***transition-delay***) y el timing o la forma de en que actuará la transición (***transition-timing-function***).

Como otros atributos combinados puede recibir todos los atributos en una sola línea, un solo atributo:

```
transition: color 1s ease-in-out .5s, font-size 3s ease 1.5s;
```

Transición, animaciones y transformaciones

Transformaciones: El atributo **transform** permite transformar el contenido de múltiples maneras, como girar, escalar, cambiar de posición, sesgar, etc.

Aquí podemos ver todas las posibilidades que nos ofrece el atributo transform.

Transición, animaciones y transformaciones

Animaciones: El atributo **animation** nos deja crear animaciones con ayuda de la propiedad ***@keyframes***, donde podemos animar cada parte de la misma utilizando todo lo aprendido hasta la fecha y más.

Al igual que transition podremos agrupar las propiedades de animation en una sola línea:

name duration timing-function delay iteration-count direction fill-mode

```
animation: animacion 6s ease .2s infinite normal both;
```

Transición, animaciones y transformaciones

Podemos ver las propiedades acerca de las animaciones en profundidad aquí .

Con la regla css @keyframes daremos todas las cualidades a nuestra animación. @keyframes debe acompañarse con el *name* que le dimos en el atributo, seguido de las instrucciones.

```
.animacion {
    width: 300px;
    height: 300px;
    background-size: contain;
    animation: animacion 6s ease .2s infinite alternate both;
}

@keyframes animacion {
    from {
        background-image: url('/src/img/1.jpg');
    }

    to {
        background-image: url('/src/img/2.jpg');
    }
}
```

Transición, animaciones y transformaciones

Se puede seccionar en porcentajes para dar las instrucciones a cumplir en el transcurso del tiempo dado:
nombramos a la animación 'animación2', la animación durará 6 segundos, la animación tendrá la curva de velocidad 'ease', tendrá un retardo de 0,2 segundos, se repetirá infinitamente, tendrá movimiento alternado de inicio-fin fin-inicio.

```css
.animacion-b {
    width: 300px;
    height: 300px;
    background-size: contain;
    animation: animacion2 6s ease .2s infinite alternate both;
}

@keyframes animacion2 {
    0% {
        background-image: url('/src/img/1.jpg');
    }

    20% {
        background-image: url('/src/img/2.jpg');
    }

    50% {
        background-image: url('/src/img/1.jpg');
    }

    75% {
        background-image: url('/src/img/background.jpg');
    }

    100% {
        background-image: url('/src/img/2.jpg');
    }
}
```

Animate.css

Animate.css

Animate.css es un sitio que nos facilitará el trabajo de hacer algunas animaciones clásicas de una manera simple y customizable.

Sólo debemos copiar el link que nos ofrece su página:

```
<link rel="stylesheet"  href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css  "  />
```

En el head de nuestro html y listo, podremos usar sus clases.

Animate.css

Para hacer uso de sus animaciones sólo debemos llamar en la clase de la etiqueta a animar, primero debemos ingresar ***animate__animated*** para que lo identifique, luego ***animate__<nombre_de_la_animación>***, como vemos aquí tenemos que buscar el nombre de la animación en la página (cada vez que demos un clic en una de ellas nos mostrará un preview de la misma).

Animate.css

Cabe aclarar que todas estas clases se pueden combinar con otras clases propias nuestras, donde podemos dar color, backgrounds, tamaños, transformaciones... Todo lo que ya vimos lo podemos combinar, incluso poner atributos de animaciones para sobreescribir las que vienen con animate.css y modificar las velocidades, keyframes, curvas de velocidad, delays, etc.

Responsive

Finalizamos nuestro recorrido por HTML y CSS viendo nuestro último tema: El diseño Responsive.

Responsive

En CSS vimos la regla @keyframes, ahora veremos @media que nos ayudará a convertir nuestra página en responsive.

Recordemos que con responsive decimos que nuestro sitio se adaptará a distintos tamaños, por lo que nuestra tarea será cambiar valores en aquellos elementos que generen conflictos a la hora de redimensionar nuestro sitio.

Responsive

Lo único que debemos hacer es:

Escribir la regla **@media** seguido de **'screen'** (que representa a la salida de los datos, en nuestro caso nuestro monitor) acompañado de la anidación '**and**' y con ella anidar cualquier regla que queramos, en nuestro ejemplo el ancho máximo en el que funcionarán las clases mencionadas dentro de las llaves (**max-width: 1100px**).

```
@media screen and (max-width: 1100px) {
    header {
        flex-direction: row-reverse;
    }

    header div {
        width: 70%;
    }

    header nav ul {
        flex-direction: column;
    }
}
```

Responsive

Vemos que dentro reescribimos las clases (o selectores) con los valores que queremos modificar en nuestro sitio mientras el ancho de la pantalla no supere los 1100px.

No debemos transcribir todas las clases ni sus atributos, sino sólo las que deseemos ajustar.

```
@media screen and (max-width: 1100px) {
    header {
        flex-direction: row-reverse;
    }

    header div {
        width: 70%;
    }

    header nav ul {
        flex-direction: column;
    }
}
```

Links de interés

Linear gradients
Clippy
Smoothie

Animaciones:
Imagehover.io
Animate.css

Datos:
StateOfCss
Ejemplos
Brand Palettes
SVGrepo

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

Todo en el Aula Virtual.