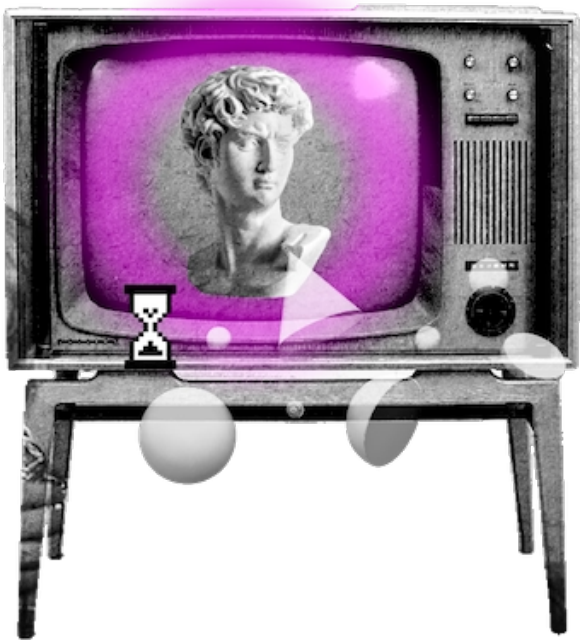
NUCBA

DISEÑO WEB

NUCBA NFT - FLEXBOX



APUNTES

Livecoding: Nucba NFT

Objetivo de la clase

Hasta ahora, veníamos viendo todas las clases temas nuevos de Css. Durante estas dos clases vamos a cambiar el enfoque: Vamos a trabajar en el maquetado de una Landing Page utilizando los conceptos aprendidos hasta ahora de **HTML** y **CSS** y que continuaremos iterando durante el módulo a medida que veamos temas nuevos.
El tema del proyecto es un **e-commerce** de Nft , al cual hemos dado en llamar **Nucba NFT.**

El objetivo, en esta primera iteración , es maquetar la versión de escritorio de dicha página, con el fin de afianzar los temas vistos hasta ahora, haciendo especial foco en el uso de **flexbox** y sus diferentes propiedades para ubicar correctamente los elementos de nuestra página.

Es super importante que presten mucha atención al mentor mientras va desarrollando las distintas secciones del proyecto, y que luego ustedes se apoyen en la clase grabada para tratar de volver a maquetar la página a modo de práctica o bien realizando su proyecto propio.

La práctica será fundamental tanto para dominar estos temas como cualquier otro tema que se vea durante la carrera, por eso les recomendamos dedicarle tiempo en lo posible todos los días (aunque sea media hora / una hora) a la práctica de los temas aprendidos en clase.

Propiedades nuevas

En este proyecto vamos a ver algunas propiedades que no hemos visto hasta ahora en clase. Vamos a ver cuáles son, cómo funcionan y qué rol cumplen en este proyecto para ejemplificar su uso.

Variables en CSS

Las variables en CSS son valores que se pueden asignar a un nombre específico y luego usar en todo el documento.

Las variables en CSS tienen varias ventajas. En primer lugar, **permiten una mayor flexibilidad y mantenibilidad del código**, ya que si se necesita cambiar un valor en todo el documento, solo se necesita **cambiar el valor de la variable en una sola línea de código**. En segundo lugar, las variables **permiten escribir un código más limpio y legible**, ya que se pueden usar **nombres significativos** para las variables en lugar de valores duros y repetitivos.

En este proyecto, usamos las variables para definir todos los colores que vamos a necesitar en el proyecto, para no tener que recordar el código **hexadecimal** exacto cada vez que tengamos que usarlo.

No obstante, las variables no se limitan solo a colores, también se pueden utilizar para tipografías, tamaños, y demás valores que se puedan reutilizar.

Analicemos ahora su sintaxis:

```
:root {
  --background: #02050e;
}

main {
  background: var(--background);
    /*Otras propiedades*/
}
```

Las variables suelen declararse en el elemento raíz **“:root”** del proyecto para que todo el resto del entorno del documento tenga acceso a ellas. Se definen colocando **dos guiones medios** y el nombre de la variable, para luego colocar el valor que deseamos almacenar en dicha variable.

Luego, para utilizarlas, utilizamos la palabra clave **var** y **paréntesis**, dentro de los cuales pondremos el nombre de la variable, precedido aquí también por los **dos guiones medios.**

Z-index

La propiedad CSS **z-index** se utiliza para controlar la posición vertical de los elementos HTML en la página web. El valor de la propiedad z-index **define la capa en la que se encuentra el elemento**, con un **valor más alto** que indica que el elemento está en una **capa superior**.

Esta se utiliza comúnmente en combinación con la **posición de CSS**, ya que la propiedad z-index solo funciona en elementos que han sido posicionados de manera explícita. Es decir, si el valor de posición es "static", la propiedad z-index no tendrá ningún efecto.

Además, la propiedad z-index solo funciona en elementos que se superponen. Si dos elementos no se superponen, la propiedad z-index no tendrá ningún efecto en su relación visual.

En este proyecto, en la interacción actual del mismo, lo estamos usando en el header, para asegurarnos de que el mismo se vea por encima de todos los demás elementos de la página cuando comenzamos a scrollear, teniendo en cuenta que el mismo tiene la propiedad **position** con valor **fixed** y por lo tanto nos seguirá a medida que bajemos por la página.

Por defecto, todos los elementos de la página tienen su **z-index** seteado en **“auto”** , lo que significa que según como se coloquen en el HTML es como se determinará la superposición entre elementos en caso de que ocurra.

```
header {
  /*Para anclar en el top de la página*/
  position: fixed;
  top: 0;
  z-index: 2; /* Para que se vea siempre por encima */
}
```

En este ejemplo, vemos que se está anclando al header al extremo superior de la página. Al bajar por la página, cómo fue colocado primero en el html, podría suceder que el header o alguno de los elementos que le vamos a agregar en futuras iteraciones quede por debajo de otros elementos. Como no queremos eso, cambiamos el valor de **z-index** por un valor superior al del resto de los elementos (que , como dijimos antes, están en **auto**).

Tengan en cuenta esta propiedad cuando trabajen con **elementos posicionados** que vean que se superponen con otros.

Si tienen varios elementos posicionados, recuerden que **el elemento con valor más alto en su propiedad z-index es el que se verá por encima.**

Cursor

La propiedad CSS **"cursor"** se utiliza para **especificar el tipo de cursor** que se muestra cuando se mueve el ratón sobre un elemento HTML. Esta propiedad es muy útil para proporcionar retroalimentación visual a los usuarios sobre qué acción se puede realizar en un elemento interactivo.

La propiedad "cursor" acepta varios valores predefinidos, como **"default", "pointer", "text", "wait", "help"**, entre otros, que representan diferentes tipos de cursores. Por ejemplo, "default" especifica el cursor predeterminado, que es generalmente una flecha, mientras que "pointer" especifica el cursor que aparece cuando se mueve el ratón sobre un enlace o un elemento interactivo.

En nuestro proyecto, utilizamos el valor **pointer** para indicar que un elemento es **clickeable.**

```
.boton{
  cursor:pointer;
  /*...Estilos del botón...*/
}
```

Min-width/height y Max-width/height

Muchas veces vamos a trabajar el tamaño de nuestros elementos con unidades de medida relativas. La problemática de esto es que a veces ese valor relativo termina siendo **muy pequeño o muy grande** acorde a lo que nosotros necesitamos para nuestro elemento. Aquí es donde entran en juego las propiedades **min-width, min-height, max-width y max-height.**

La propiedad CSS **min-width** se utiliza para **establecer la anchura mínima** que debe tener un elemento en una página web. Es decir, cuando se establece un valor de **min-width** en un elemento, éste no se puede reducir a un tamaño menor al valor especificado. Más adelante, cuando trabajemos para adaptar nuestras páginas a resoluciones de pantalla más pequeñas, veremos que esto es útil para evitar que un elemento cuyo ancho es definido con medidas relativas termine siendo muy pequeño y no quede bien en la página, y que grácias a esta propiedad se achicará el elemento únicamente hasta el valor especificado. También podría darse lo inverso, y que en pantallas con resoluciones muy grandes una anchura definida en unidades absolutas termine siendo muy pequeña para la pantalla, y utilicemos un **min-width** porcentual para que el elemento ocupe un porcentaje de su elemento padre o de la pantalla en si según corresponda.

La propiedad **min-height** trabaja de manera muy similar, pero **establece la altura mínima** que debe tener un elemento en una página web, y la altura del elemento en cuestión no podrá ser menor que el valor definido en esta propiedad.

Por otro lado, la propiedad CSS **max-width** se utiliza para **establecer la anchura máxima** que puede tener un elemento en una página web. Es decir, cuando se establece un valor de **max-width** en un elemento, éste no puede expandirse más allá del valor especificado.

Funciona a la inversa de la propiedad **min-width** , ya que limita el tamaño máximo de un elemento.

La propiedad **max-height** será el equivalente de la propiedad **max-width** pero para la **establecer la altura máxima de un elemento en una página web.**

En esta iteración del proyecto, utilizaremos principalmente la propiedad **max-width**.

En resoluciones normales de pantalla, nosotros queremos que nuestras secciones ocupen el 100% del ancho de la página y luego mediante el uso de flex acomodaremos los elementos dentro de la misma.

No obstante, nosotros como desarrolladores tenemos que asegurarnos de que una página se vea bien en la mayor cantidad de dispositivos posibles.

Teniendo en cuenta esto, debemos tener en consideración que quizás nuestras secciones, en una resolución muy amplia de pantalla, pueden quedar muy estiradas a lo ancho y los elementos internos muy separados entre sí.

Para eso, hay una técnica muy común utilizada en la mayoría de las páginas conocidas que busca limitar la ampliación de las secciones. Aquí es donde entra la propiedad **max-width**.

Nosotros tenemos el elemento **main** que abarca todas las secciones principales de contenido de nuestro documento. Las secciones se adaptarán a ese 100% , lo cual, como mencionamos antes, en las pantallas grandes puede ser problemático.

Por este motivo, utilizaremos la propiedad **max-width** para darle un ancho limite a cada una de estas secciones, haciendo que crezcan hasta cierto punto únicamente, y que el resto del espacio lo tome el color de fondo del elemento padre, que en este caso es el elemento **main.**

Es super importante que al utilizar esta técnica nos aseguremos que **las secciones queden centradas en la página y no queden colocadas sobre el extremo izquierdo de la misma.** Para ello, debemos asegurarnos de que el elemento padre esté centrando a sus elementos hijos.

Por último, si quisiéramos que las secciones tuvieran distintos colores de fondo, deberíamos trabajar con divs contenedores individuales para cada sección que cumplan el mismo rol que el elemento **main** en esta página.

```
#hero{
  max-width:1200px;
  /*...Estilos de la sección...*/
}
```

white-space y text-overflow

Otra de las técnicas que aprendemos en este proyecto es la técnica de acortar un texto muy largo usando **puntos suspensivos** , como vemos en la **“card”** de la sección de **portada** o **hero.**

Como vemos, el texto del título de la tarjeta, si no lo trabajamos, al no tener más espacio para crecer, continuaría en la línea siguiente disponible, lo cual haría que se rompa bastante el diseño de nuestra “card”. Entonces, buscamos la manera de hacer que esto no ocurra. Aquí es donde entran en escena las propiedades **white-space , text-overflow** y otra que ya vimos anteriormente, que es la propiedad **overflow.**

La propiedad **white-space** se utiliza para controlar cómo se manejan los espacios en blanco dentro de un elemento. Esta propiedad especifica si se deben conservar los espacios en blanco, colapsarlos o recortarlos, y es útil para controlar la forma en que se presenta el contenido en la página.

En este caso, utilizaremos el valor **nowrap** en esta propiedad, que implica que el texto deberá fluir en una única línea y si se supera el ancho disponible del elemento, se produce un desbordamiento horizontal y el contenido sobrante se ocultará.

Por otro lado, la propiedad CSS **text-overflow** se utiliza para controlar qué sucede cuando el contenido de un elemento es **demasiado grande para caber dentro de su contenedor**. Esta propiedad especifica si se debe truncar el texto y cómo se debe mostrar el texto truncado. En este caso, decidimos utilizar su valor **ellipsis**, que truncará el texto y mostrará puntos suspensivos cuando ya sea necesario truncarlo.

Finalmente, utilizaremos **overflow** con su valor **hidden** para que todo lo que sobresalga del contenedor del texto quede oculto, con lo cual terminamos de realizar la funcionalidad buscada y llegamos al resultado final que buscábamos.

Conclusiones finales

Luego de estas dos clases, habremos terminado el maquetado para la versión “desktop” de nuestra primera página web.

No obstante, esto es solo parte del proceso de realizar una página web , ya que nuestras páginas no deben verse bien únicamente en computadoras.

Como desarrolladores, debemos apuntar a optimizar la experiencia de usuario y por lo tanto debemos adaptar nuestros diseños a otros dispositivos, como tablets y teléfonos móviles, para que todos aquellos que quieran entrar a nuestra web, desde el dispositivo que sea , tengan una experiencia satisfactoria.

Para ello, antes de la próxima iteración de este proyecto, veremos el concepto de **responsive design** , que es un tema central de este módulo.

De nuevo, les recomendamos practicar mucho los temas vistos hasta ahora, intenten hacer su propia página tomando como ejemplo lo realizado en este proyecto y sigan enfocados en el estudio, ya que lo que se viene (**responsive design**) es un tema clave dentro del módulo.

Nucba tip: *Sigan investigando sobre los temas aquí expuestos. Practiquen, apoyense en la documentación oficial y sobre todo mantengan la constancia y la curiosidad a medida que vayan aprendiendo cosas nuevas.*

#HappyCoding 🚀