**Trabajo práctico N°3**

**Diseño web: Hojas de estilo**

Fecha de entrega: 8/5/2019

Catania Franco

Legajo: 137854

[cataniafrane@gmail.com](mailto:cataniafrane@gmail.com)

Zabala Mariano

Legajo: 143116

[Marianol.zabala@gmail.com](mailto:Marianol.zabala@gmail.com)



Universidad Nacional de Luján

Lic. en Sistemas de Información

Programación en Ambiente Web (11086)

**1) ¿Qué significa que los estilos se apliquen en cascada? ¿cómo aplica la herencia de estilos?**

El que los estilos se apliquen en cascada implica que el archivo CSS es procesado de forma top-down. De esta manera se determina qué regla de estilo aplicar si más de una afecta un mismo elemento, siendo siempre el último estilo leído el que tenga mayor prioridad (Rule Order).

La herencia en CSS se basa en la relación entre antecesor y descendiente. La herencia es el mecanismo por el cual las propiedades se aplican no sólo a un elemento especificado, sino también a sus descendientes. Esta se basa en el árbol de documentos, que es la jerarquía de elementos XHTML en una página basada en el anidamiento. Los elementos descendientes pueden heredar valores de propiedad CSS de cualquier elemento antecesor que los encierra.

Los elementos descendientes heredan las propiedades relacionadas con el texto, pero no las propiedades que afectan a la “caja”. Las propiedades que se pueden heredar son color, fuente, espaciado de letras, altura de línea, estilo de lista, alineación de texto, sangría de texto, visibilidad, espacio en blanco y espaciado de palabras. Las propiedades que no se pueden heredar son fondo, borde, display, float y clear, alto y ancho, margen, alto y ancho mínimo y máximo, contorno, overflow, padding, posición, alineación vertical y z-index.

La herencia se puede utilizar para evitar declarar ciertas propiedades una y otra vez en una hoja de estilos, lo que permite un CSS más corto.

**2) ¿Por qué es necesario utilizar un CSS de Reset?**

Un CSS Reset es un conjunto de reglas CSS corto, el cual puede estar comprimido, que restablece el estilo de todos los elementos HTML a una línea de base consistente.

Cada browser tiene su propia hoja de estilo de 'agente de usuario' predeterminada, que utiliza para hacer que los sitios web sin estilo sean más legibles. Esta es un archivo CSS incluido en el browser y se llama cada vez que se procesa una página web y antes de que se apliquen otras reglas CSS. Dado que los estilos predeterminados de un browser pueden interferir con los estilos que el diseñador puede desear aplicar, los CSS Reset se idearon para proporcionar una base consistente entre todos los browsers. Estos son llamados al igual que cualquier otro stylesheet por medio del tag link:

*<head>*

*<link rel="stylesheet" type="text/css" href="reset.css">*

*<link rel="stylesheet" type="text/css" href="Mystyles.css">*

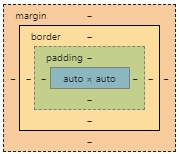
*</head>*

Los CSS Reset son especialmente útiles para normalizar el tamaño del texto y eliminar todos los márgenes. Por ejemplo, la mayoría de los browsers hacen que los links sean azules y los visitados sean violetas, dan a las tablas una cierta cantidad de borde y padding, aplican tamaños de fuente variables a H1, H2, H3, etc. y proporcionan una cierta cantidad de padding a casi todos los elementos.

Sin embargo, al usar un CSS Reset permite a un diseñador forzar a cada browser para que todos sus estilos se restablezcan a NULL, minimizando así las diferencias entre browsers tanto como sea posible.

**3) ¿Qué es el CSS box model?**

Es una caja que envuelve cada elemento HTML. Consiste en: márgenes, bordes, relleno y contenido.



**Contenido**: contiene el contenido "real" del elemento, como texto, una imagen o un reproductor de vídeo. Sus dimensiones son el content width (o el ancho del cuadro de contenido) y el content height (o la altura del cuadro de contenido). A menudo tiene un color de fondo o una imagen de fondo.

**Relleno:** Extiende el área alrededor del contenido la cual es transparente. Sus dimensiones son padding-box width y padding-box height.

**Borde:** Es un borde que rodea el padding y el contenido. Su grosor es determinado por border-width.

**Margen:** Extiende el área alrededor del borde la cual es transparente como el padding. Útil para dejar un espacio vacío respecto a los elementos vecinos. Sus dimensiones se pueden modificar mediante los atributos margin-top, margin-right, margin-bottom, margin-left.

**4) ¿Cuál es el código que hay que insertar en una hoja de estilo para poder usar WebFonts?**

Antes, los diseñadores estaban limitados a utilizar fuentes populares como Arial, Helvetica, Verdana, y Times New Roman.

Web fonts permiten a los diseñadores web usar fuentes que no están instaladas en las computadoras de los usuarios que solicitan nuestra página haciendo que estos descarguen el archivo pertinente desde el web server.

Hay varios repositorios que ofrecen una gran variedad de fuentes de código abierto disponibles desde una CDN. Las opciones más populares son:

* Google Fonts: <https://fonts.google.com/>
* Font Library: <https://fontlibrary.org/>
* Adobe Edge: <https://edgewebfonts.adobe.com/>

Para añadir una Fuente en una página se puede usar el tag <link> dentro del tag <head> en el archivo HTML de la siguiente forma:

*<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">*

La Fuente es descargada y procesada en paralelo a nuestra stylesheet.

Alternativamente, un archivo CSS puede usar WebFonts haciendo un @import:

*@import url('https://fonts.googleapis.com/css?family=Open+Sans');*

Esto bloquea el procesamiento de las siguientes stylesheets hasta que los estilos de fuente hayan sido parseados.

La web font puede ser usada en cualquier declaración CSS:

*body {  
 font-family: "Open Sans", sans-serif;  
}*

**5) ¿Qué son y para qué sirven los pseudoElementos?**

Los pseudo-element en CSS son usados para dar estilo a partes específicas de un elemento.

Por ejemplo, pueden ser usados para:

* Dar estilo a la primer letra o línea de un elemento.

p::first-letter {  
   atributo: valor;

}  
  
p::first-line {  
  atributo: valor;

}

* Insertar contenido antes o después de un elemento.

h1::before {  
  content: (Ej: una imagen);  
}

h1::after {  
   content: (Ej: un símbolo);  
}

* Modificar la parte de un elemento que sea seleccionada por el usuario.

::selection {  
  color: value;   
  background: value;

cursor: value;

outline: value;  
}

**6) ¿Cómo se calcula la prioridad de una regla CSS? Expresarlo como una fórmula matemática.**

La prioridad dentro de las reglas CSS determina que regla de estilo aplicar si más de una afecta un mismo atributo de un elemento HTML. Se debe tener en cuenta:

* La prioridad de una propiedad CSS en un documento HTML es determinada de forma top to bottom y left to right, teniendo así más prioridad las reglas que sean definidas luego en el código
* Los valores definidos como !important tendrán la mayor prioridad posible.
* Inline CSS (definido dentro de un tag HTML) tiene mayor prioridad que embedded (dentro de tags <style>) y external (dentro de tags <links>) CSS.
* embedded y external CSS tienen la misma prioridad, pero sobre escribirá las reglas del otro quien sea escrito luego.
* Selectores con orden jerárquico:
  1. IDs
  2. Clases y pseudo-clases
  3. Elementos y pseudo-elementos

Finalmente se obtiene que:

Valor definido como !important > Inline > id nesting > id > class nesting > class > tag nesting > tag.

Sin embargo, pueden darse casos donde las reglas definidas en CSS que afectan un elemento no sean tan simples como la siguiente donde se especifica solo un selector:

*.clase1 {*

*color: red;*

*}*

Sino que pueden definirse múltiples:

*.clase1 section{*

*color: green;*

*}*

En estos casos se tiene en cuenta la especificidad, de forma que la regla más especifica será la tenida en cuenta al definir el estilo de un elemento.

La regla más específica será quien cumpla primero con una de las siguientes consignas:

* Quien tenga mayor número de selectores IDs
* Quien tenga mayor número de selectores clases y pseudo-clases
* Quien tenga mayor número de selectores elementos y pseudo-elementos

**7) ¿Qué es el view port? ¿Cómo se configura? ¿qué problema soluciona?**

El viewport es el área visible por el usuario de la página web.

Este varía con el dispositivo, siendo más chico en un teléfono que en una computadora de escritorio.

Antes del uso masivo de celulares y tablets, las páginas web eran diseñadas solo para computadoras de escritorio, y era común que las páginas web tengan diseños estáticos y tamaños fijos. Estos tamaños fijos resultaban muy grandes para entrar en el viewport de un celular.

Para resolver este problema, los browsers comenzaron a escalar para abajo la página completa para que entrara en la pantalla.

Esto se puede lograr con HTML5 mediante el tag <meta>.

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

Esto le indica al browser como manejar las dimensiones y escala de la página. El atributo width=device-width hace que el ancho de la página se corresponda con el ancho de la pantalla del dispositivo.

El otro atributo, initial-scale=1.0, establece el nivel de zoom inicial cuando la página es cargada por primera vez por el browser.

**8) ¿Qué son las media querys? Enumere los distintos tipos de medios y las principales características de cada uno de ellos.**

Media query es una técnica CSS introducida en CSS3.

Estas utilizan la regla @media para incluir un bloque de propiedades CSS solo si una condición especifica es verdadera.

Se recomienda usar media querys para aplicar diferentes estilos para pantallas pequeñas y grandes. Esta propuesta surge ya que establecer widths de CSS absolutos muy grandes para los elementos de página hará que el elemento sea demasiado ancho para la ventana en un dispositivo más pequeño. En su lugar, se debe hace uso de valores de ancho relativo, como width: 100%.

Ejemplo:

  /\* Para monitores grandes: \*/

.col-1 {width: 30%;}  
.col-2 {width: 40%;}  
.col-3 {width: 50%;}  
  
@media only screen and (max-width: 768px) {  
  /\* Para teléfonos: \*/  
  [class\*="col-"] {  
    width: 100%;  
  }  
}

Los tipos de medios describen la categoría general de un dispositivo. Excepto cuando se usan los operadores lógicos not u only, el tipo de medio es opcional y el tipo All estará implícito.

* **All:** Apto para todos los dispositivos
* **Print:** Destinado a documentos vistos en una pantalla en modo de vista previa de impresión.
* **Screen:** Destinado principalmente a pantallas de computadoras, tablets, Smart-phones, etc.
* **Speech:** Destinado para lectores de pantalla que leen la página en voz alta.

**9) ¿En qué circunstancias se pueden utilizar las variables css? ¿Qué problemas pueden traer aparejadas? ¿Cuándo consideras que sería bueno utilizarlas?**

Las variables CSS son utilizadas cuando se quiere usar un mismo valor, definido por el diseñador, para múltiples selectores (elementos, clases, IDs, etc.) presentes en el documento CSS. Su ventaja es clara si pensamos en la situación en la que debemos cambiar este valor compartido, ya que solo modificaremos la variable en cuestión y no un valor individual para cada selector.

Además, añade un beneficio semántico debido a que los nombres de las variables pueden identificar mejor que afecto tiene el valor dado.

Las variables son creadas utilizando la siguiente notación:

*:root {*

*--main-color: black;*

*}*

Y accedidas de la siguiente manera:

*p {*

*color: var(--main-color);*

*}*

Al declarar la variable en la pseudo-clase :root se evita la repetición del valor a lo largo del CSS.

Los problemas que pueden traer aparejadas las variables son:

* Dificultan la lectura del documento en el caso de tener numerosas variables y siempre y cuando los nombres no sean muy explicativos.
* Aproximadamente 1 de cada 3 browsers no soporta completamente las variables CSS, impidiendo a sus usuarios ver la página de forma correcta.

<https://caniuse.com/#search=css%20var>

**10) CSS Grid Layout ¿Qué es? Explicar las reglas que intervienen en el armado de una grilla. ¿Qué ventajas y desventajas tiene frente a otros Layouts?**

CSS Grid Layout (también conocido como "Grid"), es un sistema de diseño basado en una tabla bidimensional que tiene como objetivo cambiar la forma en que diseñamos interfaces de usuario basadas en grillas (filas y columnas). CSS siempre ha sido utilizado para diseñar nuestras páginas web, pero nunca ha hecho un muy buen trabajo. En primer lugar usamos tablas (<table>), luego floats, posicionamiento e inline-block, pero todos estos métodos eran esencialmente hacks y dejaron fuera funcionalidades importantes como, por ejemplo, centrado vertical.

Flexbox ayudó, pero está diseñado para diseños unidimensionales más simples, no para complejos bidimensionales. Grid es el primer módulo CSS creado específicamente para resolver los problemas de diseño presentes al diseñar un sitio web.

Para armar un grid primero se debe definir un elemento contenedor con *display: grid*, establecer los tamaños de columna y fila con *grid-template-columns* y *grid-template-rows*, y luego colocar sus elementos hijos en la cuadrícula con *grid-column* y *grid-row*. De forma similar a Flexbox, el orden de los elementos del grid no importa. CSS puede colocarlos en cualquier orden, lo que hace que sea muy fácil reorganizar su grid con media queries.

**Grid Container**

El elemento sobre el cual *display: grid* es aplicado. Es el padre de todos los elementos del Grid. En este ejemplo “container” es el grid container:

*<div class="container">*

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

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

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

*</div>*

**Grid Item**

El hijo (descendiente directo) del grid container. Los elementos “ítem” son Grid ítem, no así “sub-item”:

<div class="container">

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

<div class="item">

<p class="sub-item"></p>

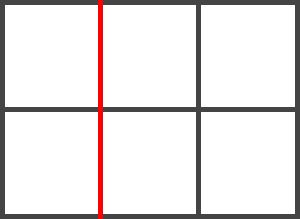
</div>

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

</div>

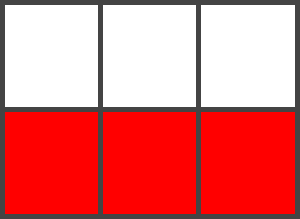
**Grid Line**

Son las líneas divisoras que le dan estructura al Grid. Pueden ser verticales ("column grid lines") u horizontales ("row grid lines") y residir a cada lado de una fila o columna. La línea roja es un ejemplo de una column grid line.



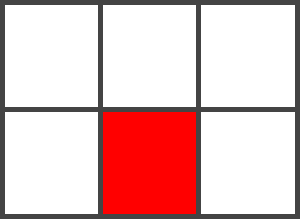
**Grid Track**

Es el espacio entre dos Grid Lines adyacentes. Son las columnas o filas del Grid. La imagen muestra el Grid Track entre la segunda y tercera Grid Line Row.



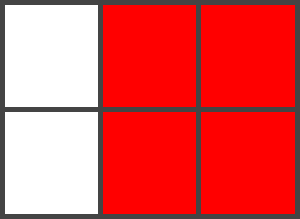
**Grid Cell**

El espacio entre dos Grid Line Rows adyacentes y dos Grid Line Colums adyacentes. Es una sola "unidad" del Grid. La Grid Cell de la imagen se encuentra entre la segunda y tercera Grid Line Column, y entre la segunda y tercera Grid Line Rows.



**Grid Area**

El espacio total rodeado por cuatro Grid Lines. Un Grid Area puede estar formada por cualquier número de Grid Cells. La imagen muestra la Grid Area entre la segunda y cuarta Grid Line Columns, y la primera y tercera Grid Line Rows.



**11) ¿Qué puntos en común y en que se diferencian las Material Design Guidelines de Google y las Human Interface Guidelines de Apple?**

Ambas guías buscan fomentar el desarrollo de aplicaciones funcionales y de alta calidad que sean fáciles de usar por el usuario final.

Las guías se relacionan por buscar claridad: Que el texto sea leíble, uso de iconos representativos de su función y la cantidad justa de decoración. Se debe hacer un uso inteligente con los colores para resaltar aquellos partes funcionales de la interfaz con las que el usuario puede interactuar.

Se debe permitir acceder a otras capas visuales, utilizando transiciones, que evidencien en todo momento donde nos encontramos jerárquicamente dentro de la aplicación para que el usuario no pierda contexto.

Sin embargo, más allá de las buenas prácticas que promueven para mejorar la accesibilidad de la aplicación, se pueden encontrar diferencias en el resultado final de la interfaz de cada uno.

**Apple** es conocida por su diseño *plano*. Evita el uso excesivo de colores o la difuminación de los mismos, texturas, sombras, y cualquier otra cosa que desvíe la atención del usuario.

Tanta simplicidad puede provocar confusión en el usuario respecto a que elementos son clickeables y cuales no.

Por otro lado, **Material Design** también puede parecer plano a simple vista, pero añade *animaciones* y sombras sobre los elementos apropiados para mantener un diseño simple pero más fácil de navegar al mismo tiempo.

Otra diferencia notable dentro de la navegabilidad es que Google diseño una barra de navegación universal en la parte inferior de la pantalla de los dispositivos Android, dando siempre la posibilidad de volver hacia atrás cuando se desee.

Apple no cuenta con tal característica, por lo que las aplicaciones suelen incluirlo en la esquina superior izquierda de la pantalla. Otra opción que ofrece para volver a la pantalla anterior es deslizar el dedo de izquierda a derecha.