Laboratorio #3: CSS

Antecedentes

El lenguaje de marcado de hipertexto, (HTML) desde su concepción, se ha utilizado para representar la estructura de los documentos HTML, permitiendo al desarrollador identificar cabeceras, párrafos, listas, tablas, imágenes, entre otros. Sin embargo, HTML provee pocas facilidades para especificar la presentación del contenido (letras, colores, espaciado, flujo del texto, etc.).

En los inicios del Web, era suficiente con tener una estructura para el documento HTML, ya que el Web se usaba para publicar artículos y documentos técnicos sin mucho énfasis en cómo se veían. Al aumentar el interés en el Web, tanto la audiencia como los autores requerían funcionalidades que permitieran evolucionar los documentos, los autores de contenido para el web querían ejercer más control sobre la presentación de sus documentos. Esto confirmó la necesidad de separar la estructura y la presentación.

A medida que se diversificaban los navegadores web, en la comunidad Web se presentaron varias propuestas de lenguajes para aplicar estilos y presentación a documentos HTML. De estas propuestas, se unieron dos de las más completas (CHSS y SSP1) para crear la especificación de CSS (Cascading Style Sheets) Hojas de Estilo en Cascada.

Las hojas de estilo en cascada, fueron adoptadas por los navegadores web, cuyos desarrolladores incorporaron los analizadores sintácticos2 para interpretar las hojas de estilo con la especificación CSS del W3C. Las hojas de estilo proveen formas de asociar información de presentación con los elementos estructurales de un documento HTML en una forma que no corrompen su estructura subyacente. [1]

Definición

Lenguaje de hojas de estilo: Un lenguaje de hojas de estilo es uno que expresa la presentación de documentos estructurados. Una característica atractiva de los documentos estructurados manejados con lenguajes de hojas de estilos, es que el contenido puede ser reutilizado en varios contextos y presentados en distintas maneras. Diferentes hojas de estilos pueden ser adjuntadas a la estructura lógica para producir distintas presentaciones. [2]

CSS (Cascading Style Sheets): La hoja de estilo en cascada, es un lenguaje de hoja de estilos usada para describir la semántica de presentación (la vista y

[1] CHSS: Cascading HTML Style Sheets de Håkon Wium Lie; SSP: Style Sheet Proposal de Bert Bos.

[2] Analizador sintáctico: Es un script o programa que convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada. En inglés se denominan *parsers*.

formato) de un documento escrito en un lenguaje de marcado. Su aplicación más común es darle estilos a páginas web escritas en HTML y XHTML, pero el lenguaje puede ser aplicado a cualquier tipo de lenguaje de marcado incluyendo sus distintas variantes.

El lenguaje de estilos, CSS, ha tenido un crecimiento y evolución paulatina que ha permitido tanto dar extensión al lenguaje como permitir a los navegadores adaptarse a las propuestas y recomendaciones emitidas por el W3C sin dejar de responder de manera bastante satisfactoria las necesidades de presentación presentes para cada época. En su primera versión, publicada en Diciembre de 1996, fue bastante elemental pero funcional para la época. La versión 2 en 1998, agrega, no solo nuevos elementos, sino también consideraciones de accesibilidad. Finalmente, la tercera, se dividió en distintos módulos, cuyas recomendaciones formales comienzan en el año 2011 pero que aún posee módulos propuestos por la W3C que se encuentran como borradores de trabajo, últimos llamados o candidatos para recomendación.

La primera versión de CSS, por ser precisamente la primera y respondiendo a las necesidades fundamentales del momento, define la sintaxis, propone la cascada como forma de asignación de prioridades y agrega elementos de estilo relativos a propiedades de fuente, colores de elementos, atributos de texto (espaciado por ejemplo), alineamiento, márgenes, bordes, relleno, posicionamiento. Esta versión ya no es mantenida por la W3C y sufrió su última revisión en el año 2008 para continuar la evolución de las otras versiones de CSS posteriores.

CSS2 agrega mejores opciones de posicionamiento, el concepto de media types, aural style sheet procurando proveer accesibilidad a la presentación en el web a personas con discapacidades (http://www.w3.org/WAI/), texto bidireccional y nuevas propiedades de fuente como sombras al igual que nuevos posibles valores para las definiciones de propiedades. Al igual que la primera versión CSS2 fue sustituido por la revisión 1 de CSS2 por lo cual la especificación de la misma ya no es mantenida por la W3C.

CSS 2.1 viene a corregir algunos errores de CSS2, remueve características pobremente soportadas o no totalmente interoperables y agrega algunas extensiones de navegadores ya implementadas a la especificación. CSS2.1 tuvo una evolución como recomendación algo aparatosa, al pasar en muchas ocasiones desde candidato a recomendación a borrador de trabajo en varias ocasiones, hasta que en Abril del 2011 fue propuesta como propuesta a recomendación para que finalmente, en Junio de ese mismo año fuese aceptada como recomendación. Esta versión se encuentra como recomendada aún desde Junio del 2011 y aún hoy en día es corregida.

CSS3, a diferencia de las versiones anteriores, se desarrolló bajo el esquema de módulos, dado que esta especificación era ya bastante grande, lo cual ha provocado distintos estados entre cada uno de estos módulos mencionados.

Estos módulos vienen a representar nuevas características o extensiones de algunos ya propuestos y al momento se mantienen más de cincuenta módulos correspondientes a CSS3 los cuales han ido alternando entre distintos estados, encontrándose exclusivamente como recomendaciones, cuatro de ellos (media queries, namespaces, selectores de nivel 3 y color). Por motivo de lo reciente que resultan ser las propuestas de CSS3 algunos navegadores no tienen, al momento, implementaciones de todas ellas, por lo cual se debe ser cuidadoso respecto a qué características de CSS3 se desean utilizar si se desea mantener un cierto nivel de uniformidad entre los distintos navegadores.

Semántica y uso de CSS

La semántica de CSS contempla una lista de reglas que consiste en uno o más selectores y un bloque de declaraciones donde se asignan las propiedades a los elementos seleccionados.

Selectores

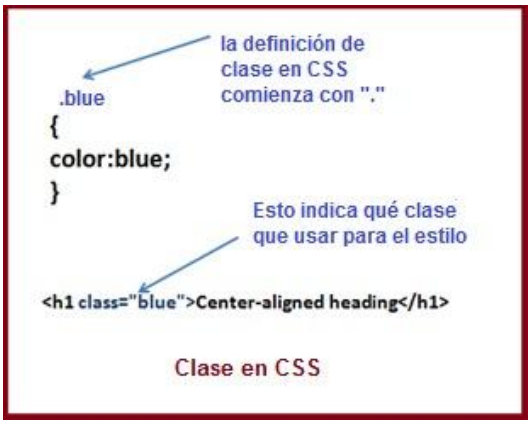
Los selectores permiten al desarrollador indicar para cuáles elementos se deben aplicar un cierto grupo de reglas. Existen cuatro tipos de selectores, siendo el cuarto un derivado de los primeros tres:

Selector de id: Este selector permite aplicar sobre un elemento que cumpla con un identificador único, como es el atributo id de un elemento, el grupo de propiedades que se defina en la regla. La semántica de este identificador es:

```
#<id> {
<propiedades>
}
```

Selector de clase: El concepto de clase en CSS permite definir un estilo, y aplicar este estilo a múltiples elementos en el documento HTML. Para que un elemento pertenezca a una clase se define como un atributo del elemento. El selector de clase permite aplicar la definición del estilo a los elementos con atributo de clase equivalente. La semántica del identificador es:

```
.<clase> {
<propiedades>
}
Y la semántica del elemento para la clase es:  <elemento class=”clase”>
```



Selector de nombre: Este selector permite aplicar sobre un nombre de elemento el grupo de propiedades que se defina en la regla. Cuando hablamos de nombre de elementos esto refiere a las etiquetas, y las distintas clasificaciones que estas involucran. La semántica del identificador es:

```
<nombre> {
<propiedades>
}
```

Selectores agrupados y anidados: Los selectores CSS no solamente pueden ser individuales, sino que pueden ser colocados de forma agrupada o de forma anidada.

Los selectores agrupados son simplemente listas de selectores a los cuales se aplican las propiedades indicadas por la regla. La semántica es semejante a las anteriormente definidas, con la única excepción que los selectores distintos son separados por medio de comas (,) resultando en la siguiente sintaxis:

```
<selector>,<selector>,…
{
<propiedades>
}
```

Los selectores anidados, permiten definir una jerarquía de aplicación de las propiedades de la regla de forma vertical. Este aplica bajo el esquema de que la regla es válida para aquellos elementos que se encuentren, en la estructura del árbol DOM, como hijos de cualquiera de los elementos que cumplan el atributo más externo o primero y que a su vez se encuentren como hijos de los elementos que cumplan el segundo atributo, y de esa manera con todos los

atributos intermedios hasta que cumplan el atributo más interno o último atributo.

La sintaxis para este tipo de selector es la siguiente

```
<atributo> <atributo> <atributo> <atributo> {
<propiedades>
}
```

```
a { color:red; text-decoration:none; }
    ↓
a.fancy { color:pink; text-decoration:none; }
    ↓
a#example { color:lime; text-decoration:none; }
```

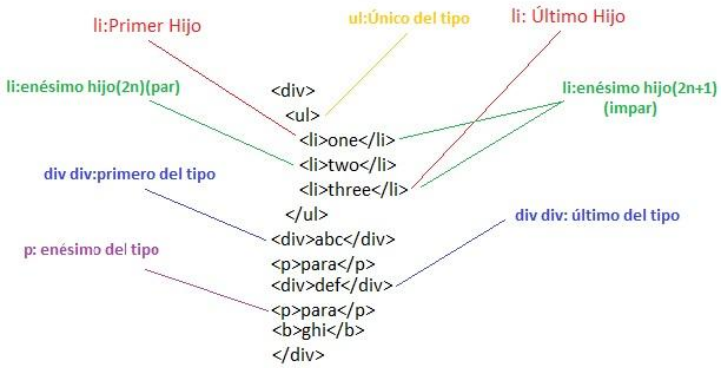

Para comprender de manera más adecuada el funcionamiento de dicho selector se puede ilustrar mediante un ejemplo.

```
.clase p {
…
}
```

En este caso todos aquellos párrafos hijos de los elementos con la clase clase cumplirían esta regla.

Pseudo-clases: Estos elementos permiten realizar formateo en base a información que no se encuentra en el árbol del documento. Existen diversas pseudo-clases en CSS sin embargo una de las más conocidas es :hover que permite especificar que vista css estará disponible si el elemento se encuentra siendo apuntado por el usuario. La sintaxis para el uso de pseudo clases es la siguiente

```
<selector>:<pseudo-clase> {
<propiedades>
}
```



Existe igualmente un selector universal * que permite seleccionar cualquier elemento.

Hay otros diversos selectores más específicos, como pueden ser, por ejemplo especificadores de selectores mediante el uso de atributos con comparación a valores si se desea, nuevas pseudo-clases, pseudo-elementos, e inclusive negadores de selección. Pueden verificar los selectores disponibles en http://www.w3.org/TR/css3-selectors/

Declaración

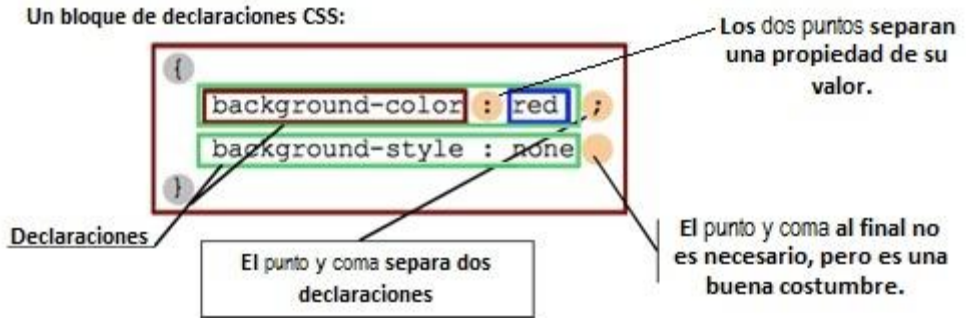
El asignar un valor a una cierta propiedad es la característica y función fundamental de CSS. Y el asignar esos pares propiedad valor es lo que se llama en CSS una declaración. Es el elemento nuclear de definición de propiedades por lo cual se hace sumamente importante. La sintaxis de la declaración es simple, <propiedad>:<valor>, es importante observar los dos puntos que separan cada uno de estos elementos.

Bloque de declaración

El bloque de declaración es el lugar donde se definen las propiedades aplicadas a un selector o grupo de selectores en una cierta regla. Este es quien define como tal qué elementos de presentación se deben aplicar en la vista.

El bloque de declaración se encuentra entre llaves y el mismo consiste de pares propiedad valor separados por punto y coma. Una sintaxis final de cómo se vería la sintaxis de una regla css sería

```
selector [, selector2, ...] [:pseudo-clase] {
 propiedad: valor;
 [propiedad: valor;
 ...]
}
```



Las propiedades en CSS son diversas y afectan distintos tipos de elementos visuales del documento, el comprender su uso individual es una labor detallada y que se escapa del alcance de este documento. Descripciones detalladas a nivel práctico pueden ser encontradas en:

https://developer.mozilla.org/en-US/docs/CSS

https://developer.mozilla.org/en-US/docs/CSS/Getting_Started

http://www.w3schools.com/css/default.asp

Alcance de una regla CSS

Toda regla CSS tiene un alcance de los posibles elementos a los cuales dicha regla puede afectar. Este alcance se define de manera simple: todo elemento al cual un selector aplique una regla y a sus descendientes las propiedades definidas les afectan, siempre y cuando los descendientes no tengan un valor para dicha regla aplicado.

Esto hace que podamos definir reglas CSS con un selector general y que para todos aquellos contenidos dentro de este, la misma regla se aplique, permitiendo un grado de definición general.

Cómo definir e incluir archivos CSS

Al definir las reglas CSS de momento conocemos que se realiza en archivos por separado más no los hemos enlazado entre ellos o con el documento. Para realizar estas inclusiones se definen diferentes métodos, algunos propios de css y otros definidos por HTML o incluso por los navegadores.

@import

Dentro de un archivo CSS se puede realizar la inclusión de las reglas de otro archivo CSS mediante el uso de @import, permitiendo que se puedan incluir reglas dentro de otro documento CSS. La sintaxis es simple

```
@import “ruta/al/archivo.css”;
```

El import, puede incluir luego de la ruta una lista de media queries que permiten identificar para que tipo de medio aplica esa importación.

Inclusión de archivos CSS externo en HTML

Desde un documento HTML se permite la inclusión de una hoja de estilos CSS mediante una etiqueta llamada link. Esta debe ser incluida como hija de la etiqueta head y debe incluir, al menos, dos atributos que permiten establecer la relación con el archivo CSS. Primero, el atributo href que le permite indicar la ruta al archivo css. Luego, rel, que permite indicar que es una hoja de estilos y si se desea se puede incluir el atributo type donde se indica explícitamente que el tipo del documento es CSS. La sintaxis finalmente es de la forma:

```
<link rel="stylesheet" type="text/css" href="ruta/al/archivo.css">
```


Declaración de reglas mediante la etiqueta style

Dentro de un documento HTML se pueden definir explícitamente reglas CSS al encontrarse estos contenidos dentro de las etiquetas style. Hacer uso de esto es tan simple como colocar las etiquetas de apertura y cierre y dentro de ellas definir las reglas CSS.

```
<style type=”text/css”>
/*Reglas CSS*/
</style>
```

Declaración de propiedades mediante el atributo style

Dentro de un documento HTML se pueden definir propiedades que afectan directamente un elemento mediante el atributo style. Toda lista de propiedades que se escribe como valor de este atributo permite que las reglas CSS indicadas apliquen para dicho elemento. Esta técnica es poco recomendada dado que limita el objetivo planteado para CSS, la separación de presentación y estructura, dado que, la presentación se encuentra embebida como un atributo de la estructura ofuscando esta separación mencionada.

```
<nombre_etiqueta style=”propiedad: valor; propiedad1: valor1;”>…</nombre_etiqueta>
```

Otros métodos de aplicación de reglas CSS

Los navegadores suelen proveer de formas que el usuario manipule o coloque CSS propios, en general por motivos de accesibilidad para usuarios con limitaciones.

Adicionalmente existen agregaciones a los navegadores modernos que permiten modificar el CSS al momento, modificando la visual de la web activamente (Por ejemplo Firebug).

Prioridad en CSS: La cascada

El estilo final de un elemento puede ser especificado desde varias fuentes, lo cual hace de CSS una herramienta poderosa al momento de aplicar estilos. Las fuentes de información de estilos forman una cascada, y más de un estilo puede afectar a un mismo elemento, por lo que la aplicación de los estilos tiene el siguiente orden de prioridad:

- Los estilos por defecto del navegador (browser) para el lenguaje de marcado.
- Los estilos especificados por el usuario que está visualizando el documento.
- Los estilos enlazados en el documento en este orden:
    1. En un archivo de estilos externo.
    2. En una definición de estilos al comienzo del documento que aplica a los estilos usados únicamente en el documento.
    3. En un elemento específico del documento.

Buenas prácticas CSS (CSS eficiente para el navegador)

Teniendo en cuenta el comportamiento de la cascada y la forma en la cual el navegador aplica las declaraciones en CSS es de suma importancia considerar cuál es la forma correcta de colocar selectores de forma que el navegador deba invertir el menor esfuerzo posible para deducir cuál regla aplica dónde. En función a esto hay un cierto set de buenas prácticas que permiten definir CSS de forma lo más eficiente posible para el navegador.

1. Evitar reglas universales
2. No anidar selectores de id con selectores de clase o de nombre: Un selector de id es lo más específico que se puede llegar a tener, es único, agregarle nombres o clases a la búsqueda simplemente genera un tiempo de búsqueda completamente innecesario

Errado:

```
button#botonespecifico {…}
.menu-izquierdo#icono {…}
```

Correcto:

```
#botonespecifico {…}
#icono {…}
```

3. No anidar selectores de clase con selectores de nombre: La idea de la recomendación anterior aplica igualmente en este caso. Un selector de clase es, en general aplicado a aquellos elementos (más de uno) que cumplen una cierta característica, si requerimos que ciertos elementos que se encuentren anidados en un tipo de etiqueta cumplan con ciertas reglas, a diferencia de otros que no importa donde se encuentren, es mucho más eficiente simplemente aplicar una clase a dichos elementos anidados y otra para la clase general.

   Se podría tomar como convención utilizar dentro del nombre de la clase el nombre de la etiqueta, pero si hay cambios en ese respecto entonces se debe cambiar el nombre de la clase. La mejor práctica es definir una semántica de nombres independiente para no perder flexibilidad

Errado:
p.indentado {…}
Correcto:
.p-indentado {…}
Mejor:
.nombre-semántico {…}

4. Utilizar la categoría más específica posible: Una causa importante de ralentización es colocar muchas reglas aplicadas a un selector de nombre para especificar en particular a cuáles refiere. Al agregar clases a los elementos podemos subdividir esas reglas en categorías y acelerar el tiempo necesario para realizar la búsqueda de dichos elementos.

Errado:
p[attr1=”true”] > b > i {…}
Correcto:
.i-attr1{…}

5. Evitar el selector descendiente: El selector descendiente es el selector más costoso a nivel de tiempo dentro de CSS, en particular aún más si el selector está a nivel de nombres o en un selector universal. Una forma de solucionar esto es utilizar el selector de hijo, que sin embargo, no es totalmente recomendable como veremos en la siguiente recomendación

Errado:
p div table {…}
Solución no más adecuada
p > div > table {…}

6. Evitar el selector de hijo: El selector de hijo, si bien es algo más rápido que el selector descendiente, aumenta de forma considerable el tiempo de búsqueda, por esa razón es recomendable evitarlo. La solución viene a ser la categorización de forma más específica mediante clases.

Errado:
p > div > table {…}
Correcto:
.table-p

7. Confiar en la herencia: Es importante aprender como aplica la herencia y aprovecharla, de esa forma se evita el uso de reglas o selectores innecesarios que aumentan los tiempos de búsqueda sin que haga falta.

Errado:
#item > .izquierdo {…}
Correcto:
#item {…}

Referencias

[1] http://nwalsh.com/docs/articles/css/

[2] https://en.wikipedia.org/wiki/Style_sheet_language

[3] https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[4] http://www.w3.org/TR/CSS21/cascade.html

[5] https://developer.mozilla.org/en-US/docs/CSS

Anexo: Algoritmo específico de la cascada definido por la W3C

Encontrar todas las declaraciones que apliquen para el elemento y propiedad en cuestión, para el tipo de medio que aplique. Se debe recordar que se pueden definir reglas CCS en particular para tipos de medios en lugar de hacerlo de forma general. Al hacerlo de forma general estas reglas aplican para cualquier tipo de medio.

Ordene de acuerdo a la importancia (normal o importante) y origen (autor, usuario o agente de usuario) en orden ascendiente:

Declaraciones del agente de usuario: Aquellas declaraciones CSS generales definidas por los navegadores.

Declaraciones normales de usuario: El usuario puede definir sus propias entradas CSS, sin que estas tengan la directiva !important especificada. Esto suele usarse por motivos de discapacidad de usuario

Declaraciones normales de autor: Estas son las reglas CSS definidas por el desarrollador en cualquiera de las formas especificadas enlazadas con el código HTML o inmersas en este sin que tengan la directiva !important especificada.

Declaraciones importantes de autor: Igualmente que lo indicado para las declaraciones del autor, con la excepción que estas refieren a aquellas que tengan la directiva !important indicada.

Declaraciones importantes de usuario: Al igual que las directivas definidas por el usuario con la excepción que tienen la directiva !important indicada.

Ordene las reglas con la misma importancia y origen por especificidad del selector, selectores más específicos sobrescribirán a otros más generales.

Finalmente, ordene por orden especificado: Si dos o más declaraciones tienen el mismo peso, origen y especificidad, la última indicada gana. Declaraciones en hojas de estilo importadas son consideradas como que se encuentran antes de cualquier declaración de la hoja de estilos en sí misma.

En este algoritmo se habla de la especificidad, que se calcula mediante la siguiente fórmula

Especificidad: abcd

a = 1 si la declaración proviene de un atributo style en la etiqueta, 0 en caso contrario

b = número de atributos id en el selector (#id)

c = número de otros atributos y pseudo-clases en el selector (.clase / :hover)

d = número de nombres de elementos y pseudo-elementos en el selector (div / :first-line)

“I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.” Alan Turing