h1, h2, h3 - Agrupación de varios elementos html con una misma regla de estilos

<!DOCTYPE html>

<html>

<head>

<title>Problema</title>

<style>

h1,h2,h3 {

font-family:verdana;

color:#0000ff;

}

</style>

</head>

<body>

<h1>Título de nivel 1</h1>

<h2>Título de nivel 2</h2>

<h3>Título de nivel 3</h3>

</body>

</html>

Añadir reglas para un mismo elemento html

h1,h2,h3,h4,h5,h6 {

font-family:Verdana;

}

h1 {

font-size:40px;

}

h1 tendrá dos propiedades.

Herencia de estilos definidos en elementos html

Hay ciertas propiedades que se heredan como el color, de forma que si definimos una propiedad en un elemento a, y utilizamos otro elemento b dentro de a, entonces b tiene la misma propiedad.

Es posible forzar la herencia de cualquier propiedad, ¿pero cómo?.

Selectores básicos.

Selectores descendentes

p span { ….}

p span { color: red; }

h1 span { color: blue; }

Con las reglas CSS anteriores:

* Los elementos <span> que se encuentran dentro de un elemento <p> se muestran de color rojo.
* Los elementos <span> que se encuentran dentro de un elemento <h1> se muestran de color azul.
* El resto de elementos <span> de la página, se muestran con el color por defecto aplicado por el navegador.

Selectores de clase : .destacado {…}

.destacado { …} / \*.destacado {…}

Definimos la clase destacado:

.destacado { color: red; } -- también: \*.destacado { color: red; }

aplicación:

<body>

<p class="destacado">Lorem ipsum dolor sit amet...</p>

<p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a> accumsan...</p>

<p>Class aptent taciti <em class="destacado">sociosqu ad</em> litora...</p>

</body>

p.destacado{…}

p.destacado {

color:red;

}

con p.destado se aplica esa propiedad a los párrafos con atributo class igual a destacado.

p.aviso { ... }

Todos los elementos de tipo "p" con atributo class="aviso"

p .aviso { ... }

Todos los elementos con atributo class="aviso" que estén dentro de cualquier elemento de tipo "p".

p, .aviso { ... }

Todos los elementos "p" de la página y todos los elementos con atributo class="aviso" de la página.

<p class =”aviso destacado”> … </p>

Para aplicar los estilos de varias clases sobre un mismo elemento html, podemos hacer:

<p class =”aviso destacado”> aviso grave </p>

Selectores Id: #destacado{…}

En ocasiones, es necesario aplicar un estilo a un único elemento de la tabla, hay que utilizar los selectores Id. Así, el atributo Id no se puede utilizar en el resto de elementos de la tabla.

Sintaxis para definir un seletor id:

#destacado { color: red; }

 <p>Primer párrafo</p>

<p id="destacado">Segundo párrafo</p>

<p>Tercer párrafo</p>

usos:

p#aviso { ... }

Todos los elementos de tipo "p" con atributo id="aviso".

p #aviso { ... }

Todos los elementos con atributo id="aviso" que estén dentro de cualquier elemento de tipo "p".

p, #aviso { ... }

Todos los elementos "p" de la página y todos los elementos con atributo id="aviso" de la página.

Configuración de selectores básicos.

Css permite la combinación de uno o más tipos de selectores para restringir el alcance de las reglas.

.aviso .especial { ... }

Para seleccionar aquellos elementos con un class="especial" que se encuentren dentro de cualquier elemento con un class="aviso".

div.aviso span.especial { ... }

Ahora, el selector solamente selecciona aquellos elementos de tipo <span> con un atributo class="especial" que estén dentro de cualquier elemento de tipo <div> que tenga un atributo class="aviso".

ul#menuPrincipal li.destacado a#inicio { ... }

El anterior selector hace referencia al enlace con un atributo id igual a inicio que se encuentra dentro de un elemento de tipo <li> con un atributo class igual a destacado, que forma parte de una lista <ul>con un atributo id igual a menuPrincipal.

Selectores avanzados:

*https://librosweb.es/libro/css/capitulo\_2/selectores\_avanzados.html*

No soportados por todos los navegadores.

p > span { color: blue; }: Selectores de hijos directos

Aplica a los elementos span que son hijos directos de un elemento p.

h1 + h2 { color: red }: Selectores adyacentes

h2 { color: green; }

h1 + h2 { color: red }

Las reglas CSS anteriores hacen que todos los <h2> de la página se vean de color verde, salvo aquellos <h2> que se encuentran inmediatamente después de cualquier elemento <h1> y que se muestran de color rojo.

Técnicamente, los elementos que forman el selector adyacente deben cumplir las dos siguientes condiciones:

* elemento1 y elemento2 deben ser *elementos hermanos*, por lo que su elemento padre debe ser el mismo.
* elemento2 debe aparecer inmediatamente después de elemento1 en el código HTML de la página.

a[class]: Selector de atributos.

a[class] { color: blue; }

/\* Se muestran de color azul todos los enlaces que tengan

un atributo "class", independientemente de su valor \*/

<a href="http://www.disney.com" target="\_blank">disney.com</a>

a[class="externo"] { color: blue; }

/\* Se muestran de color azul todos los enlaces que tengan

un atributo "class" con el valor "externo" \*/

[class\*="bg"]

Afecta a la clases que contienen en el nombre la cadena “bg”

:not()

Para element que no son div ni span

Body: not(div):not(span){

color: blue;

}

p:not(.diver) {

color: blue;

}

Para elementos p sin la clase diver

a[href="http://www.ejemplo.com"] { color: blue; }

/\* Se muestran de color azul todos los enlaces que apunten

al sitio "http://www.ejemplo.com" \*/

a[class~="externo"] { color: blue; }

/\* Se muestran de color azul todos los enlaces que tengan

un atributo "class" en el que al menos uno de sus valores

sea "externo" \*/

\*[lang=en] { ... }

/\* Selecciona todos los elementos de la página cuyo atributo

"lang" sea igual a "en", es decir, todos los elementos en inglés \*/

\*[lang|="es"] { color : red }

/\* Selecciona todos los elementos de la página cuyo atributo

"lang" empiece por "es", es decir, "es", "es-ES", "es-AR", etc. \*/

input[type="text"]:enabled {…}; - Enabled selector

input[type="text"]:enabled {

background: #ffff00;

}

Establece el fondo de los elementos de tipo “text” que están habilitados.

Colisiones de estilos.

El método seguido por CSS para resolver las colisiones de estilos se muestra a continuación:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su prioridad (palabra clave !important).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Unidades de medida.

Las unidades de medida son obligatorias salvo que sea 0. Si no se indica medida se ignora el valor.

**Unidades absolutas.**

Son poco flexibles y no se suelen utilizar salvo para el medio de impresión, que se suele utilizar el pt.

* in, pulgadas (*"inches"*, en inglés). Una pulgada equivale a 2.54 centímetros.
* cm, centímetros.
* mm, milímetros.
* pt, puntos. Un punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros.
* pc, picas. Una pica equivale a 12 puntos, es decir, unos 4.23 milímetros.

**Unidades relativas.**

* em, (no confundir con la etiqueta <em> de HTML) relativa respecto del tamaño de letra del elemento.
* ex, relativa respecto de la altura de la letra x (*"equis minúscula"*) del tipo y tamaño de letra del elemento.
* px, (píxel) relativa respecto de la resolución de la pantalla del dispositivo en el que se visualiza la página HTML.

Todos los navegadores muestran por defecto el texto de los párrafos con un tamaño de letra de 16 píxel. Por tanto, en este caso el margen de 1em equivale a un margen de anchura 16px.

Si queremos letras de 16px y margen de 8px, tenemos que definir

p { font-size:16px; margin: 0.5em;}

si queremos tamaño de letra de 32px y el mismo margen:

p {font-size:32px; margin:0,25em;)

En el caso:

body { font-size: 10px; }

h1 { font-size: 2.5em; }

En primer lugar, se establece un tamaño de letra base de 10 píxel para toda la página. A continuación, se asigna un tamaño de 2.5em al elemento <h1>, por lo que su tamaño de letra real será de 2.5 x 10px = 25px.

El valor de las medidas relativas no se hereda directamente, sino que se hereda su valor real una vez calculado. El siguiente ejemplo muestra este comportamiento:

body {

font-size: 12px;

text-indent: 3em;

}

h1 { font-size: 15px }

La propiedad text-indent, como se verá en los próximos capítulos, se utiliza para tabular la primera línea de un texto. El elemento <body> define un valor para esta propiedad, pero el elemento <h1> no lo hace, por lo que heredará el valor de su elemento padre. Sin embargo, el valor heredado no es 3em, sino 36px (2X1px)y no 45px (3X15px).

En el caso:

body { font-size: 1em; }

h1 { font-size: 200%; }

h2 { font-size: 150%; }

Los tamaños establecidos para los elementos <h1> y <h2> mediante las reglas anteriores, son equivalentes a 2em y 1.5em respectivamente, por lo que es más habitual definirlos mediante em.

En el caso:

div#contenido { width: 600px; }

div.principal { width: 80%; }

<div id="contenido">

<div class="principal">

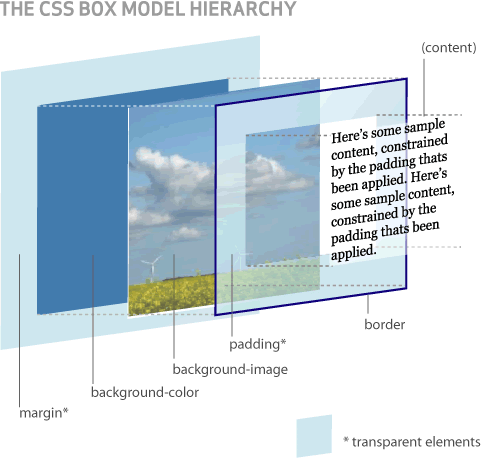
...

</div>

</div>

En el ejemplo anterior, la referencia del valor 80% es la anchura de su elemento padre. Por tanto, el elemento <div> cuyo atributo class vale principal tiene una anchura de 80% x 600px = 480px.

Modelo de cajas



Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

* Contenido (*content*): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
* Relleno (*padding*): espacio libre opcional existente entre el contenido y el borde.
* Borde (*border*): línea que encierra completamente el contenido y su relleno.
* Imagen de fondo (*background image*): imagen que se muestra por detrás del contenido y el espacio de relleno.
* Color de fondo (*background color*): color que se muestra por detrás del contenido y el espacio de relleno.
* Margen (*margin*): separación opcional existente entre la caja y el resto de cajas adyacentes.

div {

width: 300px;

padding-left: 50px;

padding-right: 50px;

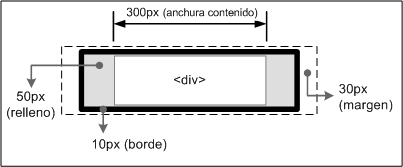
margin-left: 30px;

margin-right: 30px;

border: 10px solid black;

}

La anchura total con la que se muestra el elemento no son los 300 píxel indicados en la propiedad width, sino que también se añaden todos sus márgenes, rellenos y bordes:



**Figura 4.15** La anchura total de un elemento tiene en cuenta los márgenes, rellenos y bordes

De esta forma, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

30px + 10px + 50px + 300px + 50px + 10px + 30px = 480 píxel

Tipos de elementos.

Elementos de bloque y elementos de línea.

Los elementos de bloque empiezan siempre en una línea nueva y ocupan todo el espacio de la línea. Mientras que los elementos de línea no empiezan necesariamente en una línea nueva y ocupan lo necesario en ella, pudiendo haber otros elementos en la línea.

Los elementos en línea definidos por HTML son: a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.

Los elementos de bloque definidos por HTML son: address, blockquote, center, dir, div, dl, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, menu, noframes, noscript, ol, p, pre, table, ul.

Los siguientes elementos también se considera que son de bloque: dd, dt, frameset, li, tbody, td, tfoot, th, thead, tr.

Los siguientes elementos pueden ser en línea y de bloque según las circunstancias: button, del, iframe, ins, map, object, script.

Por sus características, los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

Posicionamiento.

El estándar de CSS define cinco modelos diferentes para posicionar una caja:

* Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.
* Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
* Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
* Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
* Posicionamiento flotante: se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

El posicionamiento de una caja se establece mediante la propiedad position:

| **Propiedad** | **position** |
| --- | --- |
| **Valores** | static | relative | absolute | fixed | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos |
| **Valor inicial** | static |
| **Descripción** | Selecciona el posicionamiento con el que se mostrará el elemento |

El significado de cada uno de los posibles valores de la propiedad position es el siguiente:

* static: corresponde al posicionamiento normal o estático. Si se utiliza este valor, se ignoran los valores de las propiedades top, right, bottom y left que se verán a continuación.
* relative: corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
* absolute: corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
* fixed: corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad position no permite controlar el posicionamiento flotante, que se establece con otra propiedad llamada float y que se explica más adelante. Además, la propiedad position sólo indica cómo se posiciona una caja, pero no la desplaza.

Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas top, right, bottom y left para controlar el desplazamiento de las cajas posicionadas:

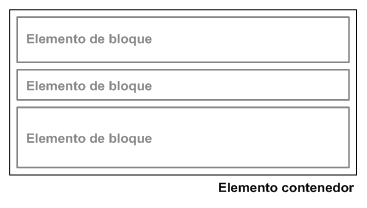
| **Propiedades** | **top**, **right**, **bottom**, **left** |
| --- | --- |
| **Valores** | [unidad de medida](https://librosweb.es/referencia/css/unidades_de_medida.html) | [porcentaje](https://librosweb.es/referencia/css/porcentajes.html) | auto | [inherit](https://librosweb.es/referencia/css/el_valor_inherit.html) |
| **Se aplica a** | Todos los elementos posicionados |
| **Valor inicial** | auto |
| **Descripción** | Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original |

En el caso del posicionamiento relativo, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el posicionamiento es absoluto, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.

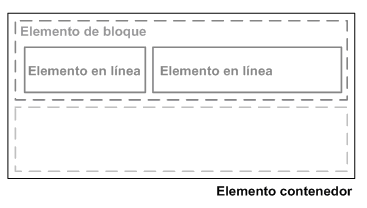
En cualquiera de los dos casos, si el desplazamiento se indica en forma de porcentaje, se refiere al porcentaje sobre la anchura (propiedades right y left) o altura (propiedades top y bottom) del elemento.

**Posicionamiento normal o static.**

Contexto de formato en bloque:



Contexto de formato en bloque:



Si las cajas en línea ocupan más espacio del disponible en su propia línea, el resto de cajas se muestran en las líneas inferiores. Si las cajas en línea ocupan un espacio menor que su propia línea, se puede controlar la distribución de las cajas mediante la propiedad text-align para centrarlas, alinearlas a la derecha o justificarlas. El cambio de propiedades como right, top o bottom, no alteran la posición de las cajas. Si lo hacen atributos como margin-left o padding-left.

**Posición relativa.**

El posicionamiento relativo desplaza una caja respecto de su posición original establecida mediante el posicionamiento normal. El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.

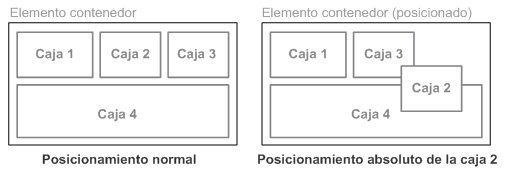
El valor de la propiedad top se interpreta como el desplazamiento entre el borde superior de la caja en su posición final y el borde superior de la misma caja en su posición original.

De la misma forma, el valor de las propiedades left, right y bottom indica respectivamente el desplazamiento entre el borde izquierdo/derecho/inferior de la caja en su posición final y el borde izquierdo/derecho/inferior de la caja original.

**Posición absoluta.**

A diferencia del posicionamiento relativo, la interpretación de los valores de estas propiedades depende del elemento contenedor de referencia de la caja posicionada.

El estándar de CSS 2.1 indica que las cajas posicionadas de forma absoluta "salen del flujo normal de la página", lo que provoca que el resto de elementos de la página se muevan y en ocasiones, ocupen la posición original en la que se encontraba la caja.



El desplazamiento de una caja posicionada de forma absoluta se controla mediante las propiedades top, right, bottom y left.

Busqueda del elemento contenedor de referencia del elemento posicionado:

Se busca sobre los elementos contenedores más próximos, aquel cuyo posicionamientos es distinto de static, en el caso de llegar a body. Si no se encuentra ninguno, se tomará como referencia la ventana del navegador no el body.

Una vez determinada la referencia del posicionamiento absoluto, la interpretación de los valores de las propiedades top, right, bottom y left se realiza como sigue:

* El valor de la propiedad top indica el desplazamiento desde el borde superior de la caja hasta el borde superior del elemento contenedor que se utiliza como referencia.
* El valor de la propiedad right indica el desplazamiento desde el borde derecho de la caja hasta el borde derecho del elemento contenedor que se utiliza como referencia.
* El valor de la propiedad bottom indica el desplazamiento desde el borde inferior de la caja hasta el borde inferior del elemento contenedor que se utiliza como referencia.
* El valor de la propiedad left indica el desplazamiento desde el borde izquierdo de la caja hasta el borde izquierdo del elemento contenedor que se utiliza como referencia.

Ver las demostraciones:

[Absoluto1](file:///D:\LibrosDemos\misNotas\software\CSS\Demos\aptana_proyect\src\absoluto1.html)

[Absoluto2](file:///D:\LibrosDemos\misNotas\software\CSS\Demos\aptana_proyect\src\absoluto2.html)

[Absoluto3](file:///D:\LibrosDemos\misNotas\software\CSS\Demos\aptana_proyect\src\absoluto3.html)

**Posicionamiento fijo.**

El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

SASS genera css a partir de ficheros .scss y .sass

La principal ventaja de SASS es la posibilidad de convertir los CSS en algo dinámico. Permite trabajar mucho más rápido en la creación de código con la posibilidad de crear funciones que realicen ciertas operaciones matemáticas y reutilizar código gracias a los mixins, variables que nos permiten guardar valores. SASS, en definitiva, se convierte en tu mejor ayudante.

Con angular-CLI podemos iniciar un proyecto que utilice SASS así:

ng new My\_New\_Project --style=sass

Si queremos añadir sass a un proyecto existente:

npm install node-sass

Podemos utilizarlo desde el fichero package.json añadiendo en la sección Script:

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"scss": "node-sass -watch scss -o css"

}

node-sass: Se refiere al paquete node-sass.

-watch: Un modificador opcional que significa "vigila todos los archivos .scss en el directorio scss/ y vuelve a compilarlos cada vez que haya un cambio."

scss: El nombre del directorio donde ponemos todos nuestros archivos .scss.

-o css: El directorio de salida para nuestro CSS compilado.

Integración sass en angular-CLI

Podemos generar un proyecto teniendo en cuenta desde el principio que vamos a trabajar con SASS.

ng generate new –style=sass

También podemos indicar –style=less y –style=scss

Si tenemos el proyecto ya generado, podemos indicar a angular-CLI que pasamos a sass con:

ng set defaults.styleExt scss

En fichero .angular-cli.json nos va a quedar:

"defaults": {

"styleExt": "sass",

"component": {

}

}

Si establecemos la siguiente estructura de directorios:

|- src/

|- sass/

|- \_variables.sass

|- \_mixins. sass

|- styles.sass

y en styles. sass importamos varialbes y mixins con:

// src/sass/styles.sass

@import './variables';

@import './mixins';

Entonces para aplicar todo, los establecemos en .angular-CLI.json con.

"styles": [

"sass/styles. sass ",

\*\*// aquí podemos incluir ficheros css

],

Si construimos el proyecto con ng build - - prod, vemos como genera un css con los resultados de la compilación.

Compilar

sass --watch input.scss output.css

You can watch and output to directories by using folder paths as your input and output, and separating them with a colon. In this example:

sass --watch app/sass:public/stylesheets

Sass would watch all files in the app/sass folder for changes, and compile CSS to the public/stylesheets folder.

Variables

Permite reutilizar valores que podemos manejar desde un solo sitio de forma sencilla y centralizada.

$brand: #F98355;

body{

color:$brand;

}

http://sassmeister.com/gist/28f19ef1925812d4be73

No solo guardar colores, también los breakpoints de nuestro site:

$tablet-landscape-desktop: '(min-width: 768px) and (max-width: 979px)';

Mixins

Es un elemento esencial cuando se trabaja bajo la premisa DRY (Don’t Repeat Yourself). Un mixin permite aprovechar un trozo de nuestro código al que podemos llamar repetidamente y que encapsula en una o más líneas de código.

// ----

// Sass (v3.4.12)

// Compass (v1.0.3)

// ----

@mixin circle($width) {

width: $width;

height: $width;

border-radius: 100%;

background:black;

}

div{

@include circle(100px);

}

SASS generará el siguiente css:

div {

width: 100px;

height: 100px;

border-radius: 100%;

background: black;

}

Los mixins son muy útiles cuando trabajas con CSS3. Tienes varios prefijos según el navegador para que, solo llamando al mixin y pasando el valor que te interese personalizar, incorpore todas las posibilidades de vendor prefixes: -webkit, -moz, -ms , etc…

Hay gran variedad de mixins que puedes encontrar a partir de proyectos como http://compass-style.org, http://bourbon.io/ o simplemente buscando en github+sass. Puedes crearte tu propia biblioteca de mixins con los que utilizas de forma más recurrente.

extend, la herencia de .sass

Hay ciertos momentos en los que deseamos aprovechar una clase o variarla ligeramente. SASS nos brinda la posibilidad de poder reutilizar este código anterior y añadir algún elemento nuevo, si así lo deseamos.

// ----

// Sass (v3.4.12)

// Compass (v1.0.3)

// ----

.featured{

background:#000;

width:100px;

height:100px;

padding:10px;

}

div{

@extend .featured;

color:orange;

font-family:sans-serif;

}

Que generará el siguiente css:

.featured, div {

background: #000;

width: 100px;

height: 100px;

padding: 10px;

}

div {

color: orange;

font-family: sans-serif;

}

Uso en html:

<div>Hola</div>

Nesting

Se acabó el código de este estilo:

div{}

div a{}

div a:hover{}

SASS nos ayuda a simplificar haciendo uso del nesting o anidación. Conseguimos escribir menos, con la indentación del código CSS podemos reflejar mejor la jerarquía:

indentacion-sass

div {

a {

&:hover {

}

}

}

Funciones

Siguiendo con las posibilidades dinámicas, si solo necesitamos hacer cálculos matemáticos podemos crearnos funciones ad hoc para llamarlas a la hora de compilar:

@function ratio ($var1, $var2)}

@return ($var1 + $var2)

}

@media: CSS media Queries

Nos permite diferentes configuraciones css dependiendo del dispositivo y el tamaño del dispositivos.

@media not|only mediatype and (expressions) {

CSS-Code;

}

Value Description

all Used for all media type devices

print Used for printers

screen Used for computer screens, tablets, smart-phones etc.

speech Used for screenreaders that "reads" the page out loud

Ejemplo:

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: pink;

}

@media screen and (min-width: 480px) {

body {

background-color: lightgreen;

}

}

</style>

</head>

<body>

<h1>Resize the browser window to see the effect!</h1>

<p>The media query will only apply if the media type is screen and the viewport is 480px wide or wider.</p>

</body>

</html>

El caracter &

Selectores de pseudo clase

código scss

a {

color: #0090B2;

&:hover {

color: #FF7A00;

}

&:active {

color: #B25500;

}

}

código css generado:

a {

color: #0090B2;

}

a:hover {

color:#FF7A00;

}

a:active{

color: #B25500;

}

Añadiendo clase o id

.feature-class {

color: #0090B2;

&:hover {

color: #FF7A00;

}

&:active {

color: #B25500;

}

&.style-class {

color: #00CEFF;

&:hover {

color: #0090B2;

}

&:active {

color: #FF7A00;

}

}

}

.feature-class{ color:#0090B2;}

.feature-class:hover {color:#FF7A00;}

.feature-class:active {color:#B25500;}

.feature-class.style-class{color:#00CEFF;}

.feature-class.style-class:hover{color:#0990B2;}

.feature-class.style-class:active{color:#FF7A00;;}

selector padre dentro de selector hijo

.feature-class {

color: #0090B2;

.parent-class & {

color: #00CEFF;

}

}

.feature-class {

color:#0090B2;

}

.parent-class .feature-class {

color:#00CEFF;

}

Con elementos sibling

Aplicado entre elementos hermanos y contiguos.

p {

margin: 0;

line-height: 1.5em;

& + & {

margin-top: 1em;

}

}

p {

margin:0;

line-height:1.5em;

}

p + p {

margin-top:2em;

}