Fuentes Específicas En CSS Dependiendo Del Sistema Operativo

Como muchos desarrolladores sabrán, en cada Sistema Operativo existe tipografías determinadas, esto es de gran utilidad cuando se desea una mayor velocidad, ya que se omite el cargar tipografías externas. Pero, ¿Cómo cargar las fuentes específicas en CSS dependiendo del Sistema Operativo? A continuación te mostraremos algunos métodos pocos práctico y otro mucho mas sencillo, para que puedas cargas las tipografías del Sistema Operativo donde te encuentres navegando.

Usando User Agent

Primero se podría usar el viejo truco donde pones el **User Agent** en un atributo de datos, sobre el elemento raíz que lo seleccionara. En la parte de Javascript se usaria:

[js]

var doc = document.documentElement;
doc.setAttribute('data-useragent', navigator.userAgent);

[/js]

A continuación, puedes establecer las tipografias, según sea necesario. En este caso, se establece de Lucida Grande para OS X, pero Helvetica Neue para la última versión.

[html]

html {

```
}

html[data-useragent*='Mac OS X'] {
Lucida Grande","Lucida Sans Unicode", Tahoma, sans-serif;
}

html[data-useragent*='Mac OS X 10_10′] {
HelveticaNeue-Light", "Helvetica Neue Light", "Helvetica Neue", Helvetica, Arial, "Lucida Grande", sans-serif;
font-weight: 300;
}

[/html]
```

Esto, básicamente funciona. Lo malo, es el uso de *user agent sniffing*, no estamos haciendo nada en particular que ponga en riesgo nuestra web, puede llegar a ser una muy mala practica.

El método más profesional

Se puede hacer un uso mas avanzado de Javascript, con lo que podemos **crear un iframe** (por lo que no hay estilos de interferencia), inserta un elemento de botón, lo cual debe tener la fuente del sistema en él por defecto.

```
[js]

(function() {
window.addEventListener("DOMContentLoaded", function() {
getSystemFontFamily(function(systemFontFamily) {
var cssText = ".system-font { + systemFontFamily + " }";
var element = createStyleElement(cssText);
document.head.insertBefore(element, document.head.firstChild);
});
});
```

```
function getSystemFontFamily(callback) {
withHiddenFrame(function(document) {
var button = document.createElement("button");
document.body.appendChild(button);
callback(window.getComputedStyle(button).fontFamily);
});
}

function withHiddenFrame(callback) {
var frame = document.createElement("iframe");
frame.style.cssText = "width: 0; height: 0; visibility:
hidden";
frame.onload = function() {
callback(frame.contentDocument);
document.body.removeChild(frame);
}
document.body.appendChild(frame);
}

function createStyleElement(cssText) {
var element = document.createElement("style");
element.type = "text/css";
if (element.styleSheet) {
element.styleSheet.cssText = cssText;
} else {
element.appendChild(document.createTextNode(cssText));
}
return element;
}
})();

[/js]

En Yosemite (10.10.1) obtendrás:

[css]

.system-font {
.HelveticaNeueDeskInterface-Regular’;
```

}

[/css]

Se obtiene un nombre raro, pero funciona. En Mac OS 10.9 obtienes:

[css]

.system-font {
.LucidaGrandeUI’;
}

[/css]

Nuevamente se obtiene un raro nombre, pero aun así, funciona.

El método más simple y elegante

Lo anterior es a nivel profesional, por ende es mas complejo. Pero existe un método aun mas simple para ser usado ;) No es tan simple como en lugar de esto se puede usar @font-face, con valores locales de tipografía, y funciona! Para esto haremos uso de la **propiedad src**.

[css]

@font-face {
MacSystem";
src:
local(".HelveticaNeueDeskInterface-Regular"),
local(".LucidaGrandeUI");
}

body {
MacSystem", sans-serif;

}

[/css]

El truco esta en que hay que escoger el nombre «más raro» de primero. Sin duda podrías extender esta idea para Linux y Windows, y hacer que las sentencia @font-face cubra otras características.