Detección De Dispositivos Del Lado Del Servidor Con JavaScript

Durante el último par de años, el Diseño Responsive de la Web y herramientas como Modernizr han vuelto muy populares. Recientemente, las técnicas de combinación (a menudo llamadas RESS), donde la optimización se realiza tanto del lado del servidor y en el cliente, se ha convertido en una tendencia. La herramienta WURFL.js recientemente lanzada, encaja de maravilla en esta categoría.

En este artículo, vamos a ver algunos casos de uso básicos de cómo utilizar **WURFL.js** para optimizar la experiencia del usuario tanto en HTML y CSS, y un ejemplo de la forma de elegir los anuncios correctos para visualizar en diferentes dispositivos. También veremos cómo **WURFL.js** es diferente, sino que complementa, la popular biblioteca de **Modernizr**.

Existia Una Vez, La Detección De

Dispositivos

Si estamos usando alguna técnica, con la cual podamos aprovechar la detección del dispositivo, ya sea con expresiones regulares en JavaScript, Modernizr ó algún otro repositorio; donde se use para dar una mejor experiencia para los usuarios. Esto suele ocurrir en dos niveles:

- Presentación del contenido y la interacción con el servicio.
- El análisis del comportamiento del usuario para determinar los patrones de uso.

El desafío es hacer esto en formas que sean escalables, fáciles de mantener y, tanto como sea posible, fácil de implementar. Para algunos proyectos, el costo y la complejidad de la implementación de herramientas de terceros en servidores es demasiado alto. Sin embargo, una solución de bajo mantenimiento que permite que un sitio web se vea bien y funcionar bien es posible, a pesar de la constante diversificación de los dispositivos. Aquí es donde **WURFL.js juega un papel bastante importante**, al proporcionar una alternativa escalable para la **detección tradicional dispositivo del lado del servidor**, a la vez que complementa otras técnicas y herramienta del lado del cliente.
Antes de continuar con lo mas interesante, echemos un vistazo a lo básico ;)

Copiar, Pegar, Listo!

No es necesario estar registrados, o comprar una version para usar WURFL.js, It’s Free! Por lo tanto, lo primero que debe hacer es copiar y pegar la siguiente línea de código **HTML** en su página:

[html]<script type=’text/javascript’ src=“//wurfl.io/wurfl.js"></script>[/html]

Tanto **HTTP** y **HTTPS** son compatibles. Si vas a utilizar la información proporcionada por el dispositivo por el guión para hacer la prestación, entonces es posible que desee incluir el **script** en el elemento <head>. De lo contrario, puede cargar de forma asincrónica.

Ahora que el **script** está en su página HTML, puede acceder al **objeto WURFL en JavaScript**. El objeto WURFL se parece a esto y está listo para usar:

[js]

```
{
complete_device_name:"Apple iPhone 5",
form_factor:"Smartphone",
is_mobile:true
}
```

[/js]

El objeto tiene tres propiedades:

1. complete_device_name: Este es el nombre con el que se

conoce el dispositivo. Por lo general, obtenemos la marca y modelo o categoría de dispositivos o una definición más genérica.

2. Form_factor:

- desktop
- app
- tablet
- smartphone
- feature phone
- smart TV
- robot
- other non-mobile
- other mobile

3. is_mobile: Esto es true o false, true si el dispositivo es una tableta u otro dispositivo móvil.

Por supuesto, podemos tener todo de la siguiente manera:

```
[js]

console.log(WURFL);

//o esto

alert(WURFL.complete_device_name);

[/js]
```

Bajo El Capó

Debido a que WURFL.js detecta el dispositivo basado en la cadena de agente de usuario y otra información proporcionada

en la cabecera **HTTP**, el contenido del archivo JavaScript **dependerá del dispositivo**. Por lo tanto, no se puede simplemente copiar el contenido del archivo y **ponerlo en línea en el código HTML ó combinarlo** con otro recurso de JavaScript.

Para entender esto en detalle, vamos a ver la ilustración anterior. El navegador hace una petición de example.com (**1**). Las etiquetas HTML que devuelve el servidor Web como respuesta(**2**) contiene la referencia a <script> WURFL.js. A continuación, el explorador representa el código HTML y empieza a ir a buscar activos – entre ellos, wurfl.io / wurfl.js (**3**). Cuando la solicitud llega WURFL.io, la solicitud HTTP es analizada por WURFL. Por lo general, a partir de dicha solicitud, habrá un éxito instantáneo, y el dispositivo se identifica sin más preámbulos, y se devuelve un único objeto JavaScript de WURFL. Sin embargo, en ciertos casos, cuando el dispositivo no se puede identificar en el lado servidor (en particular, en el caso particular de los dispositivos iOS), el archivo **JavaScript** contendrá unas cuantas comprobaciones para determinar el dispositivo. El navegador entonces evalúa el JavaScript, y el objeto WURFL está listo para usar (**4**).

WURFL.js es capaz de distinguir entre un **iPhone 5 y un iPhone 5S**.

Casos De Uso

Asumiendo que tiene **WURFL.js** en marcha y funcionando, echemos un vistazo a algunos casos en los que el **uso de WURFL.js** tiene más sentido, ya sea por sí sola o conjuntamente con **Modernizr y/o otras herramientas**. Para mostrar esto, nos referiremos a la página web WURFL.io sí, que utiliza WURFL.js de múltiples maneras.

Optimización De La Experiencia Del Usuario

Cuando se trata de diseño móvil, responsive y adaptable y todo eso, la cosa más común de hacerlo en un sitio web es mejorar la experiencia del usuario para ciertas familias de dispositivos. Mucho puede ser manejado por las consultas de los medios de comunicación, por supuesto, pero a veces se necesita la ayuda de un poco de JavaScript.

Cuando visita **WURFL.io** en su computadora portátil, la parte superior de la página tiene un fondo de vídeo, algunas simples scroll parallax y el texto que cambia de forma dinámica en función del dispositivo o navegador. Se ve muy bien en un ordenador portátil, pero los fondos de vídeo, por no hablar de scroll parallax, no sería ideal en una tableta o un teléfono inteligente, por decirlo suavemente.

Podríamos utilizar Modernizr, por supuesto, o decidir si implementar estas características en otras formas. Pero en muchos casos, a sabiendas de que el dispositivo físico es tan importante como saber si el navegador soporta una función. Podríamos encontrar un problema por el que el navegador

necesita soporte, pero el soporte no es realmente lo suficientemente bueno para hacer una gran experiencia de usuario.

Para evitar estas situaciones, debe utilizar WURFL.js y Modernizr juntos. Tenga en cuenta también que la comparación WURFL.js y Modernizr directamente no es del todo justo. **Modernizr detecta características** reivindicadas por el navegador, mientras que **WURFL.js clasifica el dispositivo de diferentes maneras**. Por lo tanto, si usted no conoce si un dispositivo en particular es compatible con cierta función de navegador detectable.

[js]

```
/*video background*/
if(!WURFL.is_mobile){
$('#vid').videoBG({
mp4:'assets/Birds_Animation.mp4.mp4´,
ogv:'assets/Birds_Animation.oggtheora.ogv',
webm:'assets/Birds_Animation.webmhd.webm'
});
}

/*The parallax scrolling*/
window.onscroll = function () {
if (!WURFL.is_mobile){
heroImage.style[prefixedTransform] = "translate3d(0px," +
window.scrollY / 2.3 + "px, 0px)";
herovid.style[prefixedTransform] = "translate3d(0px," +
window.scrollY / 1.1 + "px, 0px)";
heroText.style["opacity"] = (1 - ((window.scrollY / 6) /
100));
}
}
```

[/js]

El ejemplo anterior simplemente comprueba si el dispositivo es

móvil (SmartPhone o tableta) y presenta características consecuencia.

¿Poner Mas En El CSS?

Los ejemplos anteriores muestran cómo hacer uso de los **datos del dispositivo desde JavaScript**. Sin embargo, podemos hacer que la información del dispositivo este disponible en CSS. Podemos asignar diferentes estilos dependiendo del dispositivo. La primera técnica vamos a ver es similar a cómo funciona **Modernizr**.

Digamos que desea algún comportamiento específico definido en el CSS para dispositivos móviles. Usted tendría que agregar el siguiente fragmento de código JavaScript a la página:

[js]document.documentElement.className += ' ' + (WURFL.is_mobile ? » : 'no-') + "mobile";[/js]

Esto agregará una clase para el elemento html. Para dispositivos móviles, diría <html class=»is_mobile»>; para otros dispositivos, diría <html class=»no-is_mobile»>.
Si conoces Modernizr, entonces probablemente está familiarizado con este enfoque. Su **CSS** podría tener el siguiente contenido:

[css]

```
.mobile #menu a{
padding .5em;
}
```

```
.no-mobile #menu a{
padding .1em;
}
```

[/css]

En este sencillo ejemplo, hemos aumentado el padding en los elementos de menú de manera que sean fáciles de tocar con un dedo gordo.

Este método se puede utilizar para todas las capacidades WURFL.js. Sin embargo, debido a que **complete_device_name y form_factor** no son valores booleanos (como is_mobile), la parte CSS puede convertirse en un dolor de cabeza. Un poco más de flexibilidad puede ser útil. Aquí hay un ejemplo usando ficha atributos:

[js]

```
document.documentElement.setAttribute('data-device_name', WURFL.complete_device_name);
document.documentElement.setAttribute('data-form_factor', WURFL.form_factor );
```

[/js]

Esto pondrá a los atributos de datos de WURFL en el elemento html. Tenemos varias características interesantes con este método: Podemos apuntar a dispositivos específicos, los factores de formularios, e incluso grupos de dispositivos combinados con factores de formularios mediante el uso de selectores CSS:

[css]

```
html[data-form_factor = 'Smartphone'] #menu a{
background: green;
}
```

[/css]

Gracias al selector de atributos, incluso puede hacer coincidir cadenas:

[css]

html[data-device_name*=’Nokia’] [data-form_factor = ‘Feature Phone’] {
background: yellow;
}

[/css]

El CSS de arriba comparará los teléfonos Nokia de cualquier modelo.

Trabajando Con Anuncios En Banner

Muchos diferentes redes de anuncios están ahí fuera, cada uno con su propia especialidad. Algunos son buenos para móviles, otros para usuarios de PC. Algunos anuncios de texto flexible, otros tienen los anuncios de tamaño fijo. Si está más allá del nivel de un principiante en las redes de anuncios, entonces es posible que desee asumir algún control sobre esto. WURFL.js pueden ayudarle a tomar sus propias decisiones o influir en la red para tomar las decisiones correctas para usted.

La solución obvia es pedir WURFL.is_mobile para elegir las redes o los anuncios que son buenos para móviles y otras que

son buenas para no móvil.

[js]

```
if(WURFL.is_mobile){
displayMobileAd();
}else{
displayDesktopAd();
}
```

[/js]

Por otra parte, desde una perspectiva de diseño, siendo capaz de adaptarse a los tamaños y las proporciones de los anuncios a los puntos de interrupción y de diseñar para diferentes factores de forma de los anuncios es agradable. En el extremo, se podría hacer algo como esto:

[js]

```
switch(WURFL.form_factor){
case "Smartphone":
if(WURFL.complete_device_name.indexOf("Apple") !=-1){
showAppStoreAds();
}else(
showWebAds();
)
break;
case "Tablet":
showSpecificProportionAds();
break;
case "Feature Phone":
showTextAds();
break;
default:
showGoogleAdwords();
break;
}
```

[/js]

Finalmente

Es posible que desee considerar WURFL.js o bibliotecas similares para el análisis, la optimización de la experiencia del usuario o de la publicidad, y la biblioteca puede complementarse bastante bien con Modernizr. Mientras Modernizr detecta el soporte para ciertas funciones del navegador, WURFL.js proporciona información acerca del dispositivo físico del usuario.

WURFL.js es un puente entre el servidor y el cliente, por lo que es más fácil para los desarrolladores web front-end para tomar ventaja de la funcionalidad que puede brinda el servidor. También se puede utilizar para los sitios web actuales que han sido diseñados bajo el concepto responsive.