Feature Queries en CSS ¿Cómo funcionan?

Diseño Web

Tiempo de lectura 14 minutos.
Publicado el 8 marzo, 2018. Actualizado el 8 marzo, 2018

CSS

Alfonso Serrano

Dejar un comentario

★★★★★ 5 / 5 (1 votos)

En los últimos meses, estamos viviendo ciertos momentos de cambio en el diseño web gracias a CSS Grid. **Pero como siempre que surgen este tipo de avances, a veces tenemos el miedo de que, si empezamos a usarlos, estamos dejando atrás a todos aquellos usuarios que no pueden disfrutar de ellos** por varios motivos: navegadores desactualizados, sistemas operativos antiguos, etc. De hecho, incluso en nuestro artículos aparecen comentarios que, con razón, rehúsan de meter estas nuevas herramientas en su flujo de trabajo. A pesar de saber que optimizarían de forma clara su tiempo de trabajo, además de muchas otras bondades. Y por todos estos motivos,

queríamos hacer una pequeña parada para ver el uso de Feature Query, una herramienta CSS que quizás algunos no conozcáis. Pero que seguro que os dará muchas satisfacciones en el futuro.

Pero, ¿para que sirve Feature Query? Pues sencillamente **le dice al navegador que si soporta cierta propiedad o valor CSS, lo aplique**. Si no la soporta, pues no leerá absolutamente nada de lo que le diga el Feature Query. Por lo que se ahorra el verificar toda esa cascada de información inútil.

Muy importante antes de seguir. **En este artículo vamos a ver ciertas partes con código CSS.** Si no estás familizarizado con este lenguaje, te recomiendo que te pases antes por este artículo de introducción a CSS que escribimos hace un tiempo.

Compatibilidad de Feature Query

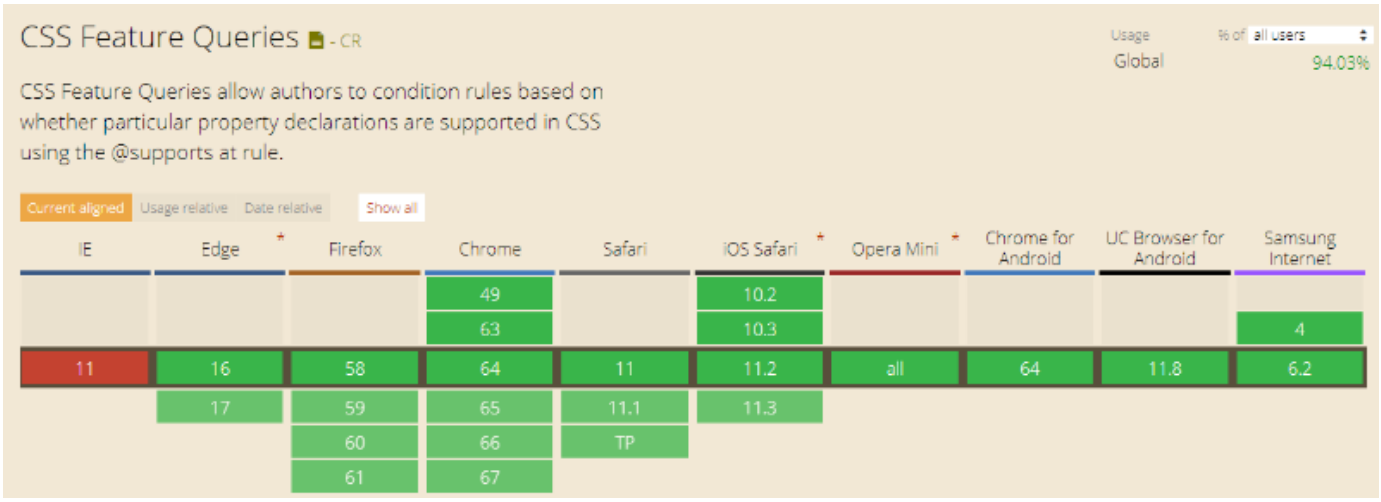


Tabla de compatibilidad de Feature Query en los diferentes navegadores web.

Como ya sabéis, antes de entrar en materia, siempre me gusta mirar antes en la web de Can i use... para comprobar la compatibilidad con los diferentes navegadores de las herramientas de las que vamos a usar. Y si habéis visto la imagen, la compatibilidad es algo mayor al 94%. Una alegría, no lo vamos a negar. Aunque como suele pasar en muchas ocasiones, seguimos dejando fuera a nuestro querido Internet Explorer 11. Pero más adelante tocaremos este tema mediante un pequeño ejemplo.

Uso de Feature Query

La sintaxis de Feature Query es muy parecida a la de Media Query, tal como vemos en la siguiente imagen. Si no sabes que es un Media Query, te recomiendo que eches un vistazo al artículo que escribimos sobre este tema.

No es muy difícil de usar. **Si el navegador entiende el selector que pongamos en el Feature Query, entonces todo el estilo que esté dentro de los corchetes será aplicado.** Pero si no lo entiende, directamente pasará de largo todo ese trozo de código.

También hay que dejar claro que esta herramienta **no sirve para arreglar posibles errores que tengamos en nuestro código** o en una web. Simplemente sirve para preguntarle al navegador si usa o no cierto selector y que, según sea su respuesta, lo use o no.

Pero como solemos decir, que mejor forma de entender el funcionamiento de una herramienta, que usándola directamente.

Ejemplo de Uso de Feature Query

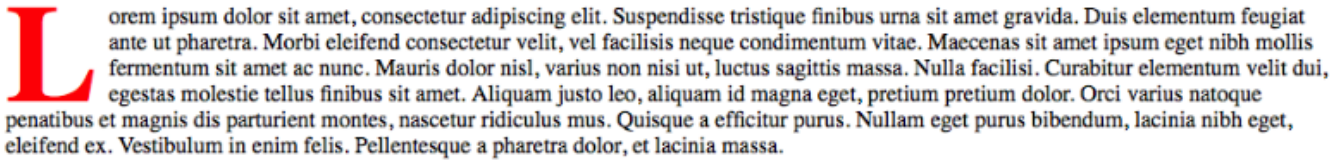
Vamos a hacer un **pequeño ejercicio para ver el funcionamiento de Feature Query**. Queremos usar el selector CSS `initial-letter`. Este selector hace que la primera letra de la primera palabra de un texto se muestre a mayor tamaño. Muy útil si estamos diseñando la web de un periódico, o queremos darle un toque más literario a un texto.

Pero nos enfrentamos a un problema: actualmente, el único navegador web compatible es Safari. Por lo que si escribimos esas líneas de código en nuestra hoja de estilos, solo se mostrará en Safari, y el resto de navegadores, al no conocerlas, pues directamente la obviarán. Así que, vamos a escribir un pequeño Lorem ipsum, y vamos a aplicarle este CSS:

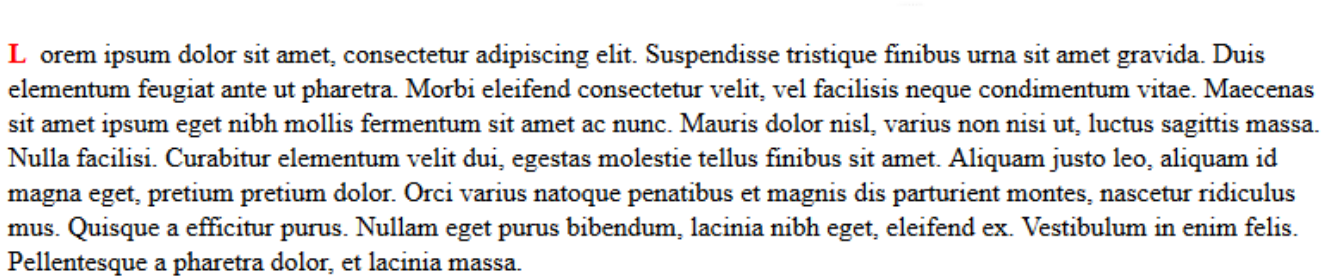
```
p::first-letter {
     initial-letter: 4;
```

```
        color: red;
        font-weight: bold;
        margin-right: 0.5em;
    }
```

Si estamos en Safari, nos deberá de aparecer de la siguiente manera:



Y si estamos en cualquier otro navegador web, como por ejemplo Firefox, veremos esto:



Como podéis ver, no es lo ideal. Firefox aun no es compatible con este selector, por lo que le es imposible aplicarlo. El problema llega cuando si nos aplica parte de lo que hemos escrito, como el color de la fuente o que esté en negrita. Digamos que nos hace la mitad del trabajo. Y esto no es lo que buscamos. **Lo ideal sería que, si un navegador web es compatible con `initial-letter`, lo use con normalidad, pero que si no es compatible, no use ninguno de sus valores.**

Muchos diréis, no sin algo de razón, que para que se muestre de esta manera, mejor no usar `initial-letter`. Pero pensemos de otra manera. Quizás lo idóneo es poder usar desde un primer momento todas las nuevas herramientas que van saliendo, y poder incluirlas en nuestra hoja de estilos. **Y conforme los navegadores se vayan actualizando y siendo compatibles con ellas, que automáticamente se vayan mostrando.** Y aquí es donde entra en funcionamiento Feature Query.

```
@supports (initial-letter: 4) {
  p::first-letter {
    initial-letter: 4;
    color: red;
    font-weight: bold;
```

```
        margin-right: 0.5em;
    }
}
```

Con este trozo de código, le estamos diciendo al navegador que, si soporta `initial-letter`, aplique todo lo que está dentro del Feature Query. Y si no lo soporta, que obvie todo el contenido que está dentro. Con esto ganamos dos cosas. Por un lado, hacemos uso de una nueva herramienta que quizás, por temas de diseño, nos haga falta. Además de que, cuando estemos en un navegador que no lo soporte, no nos muestre cosas raras. Y por otro, **los navegadores que hoy no son compatibles, pero que en el futuro lo sean, utilizarán automáticamente el selector**, sin tenernos que preocupar de cambiar nuestra hoja de estilos.

Como decíamos al principio, todo esto podemos aplicarlo a otros selectores, como CSS Grid. Podemos diseñar nuestra web con sus `floats` como venimos haciendo hasta ahora, asegurándonos que va a funcionar de forma correcta en cualquier navegador. Pero además, añadimos un Feature Query con su Grid. Así, los navegadores modernos se aprovechan de las bondades de estas nuevas herramientas. Los antiguos seguirán teniendo soporte. Y **los navegadores web que se vayan actualizando a lo largo del tiempo, pasarán de usar los clásicos `floats` a funcionar con Grid sin que tengamos que hacer nada extra** en el momento en el que empiecen a ser compatibles.

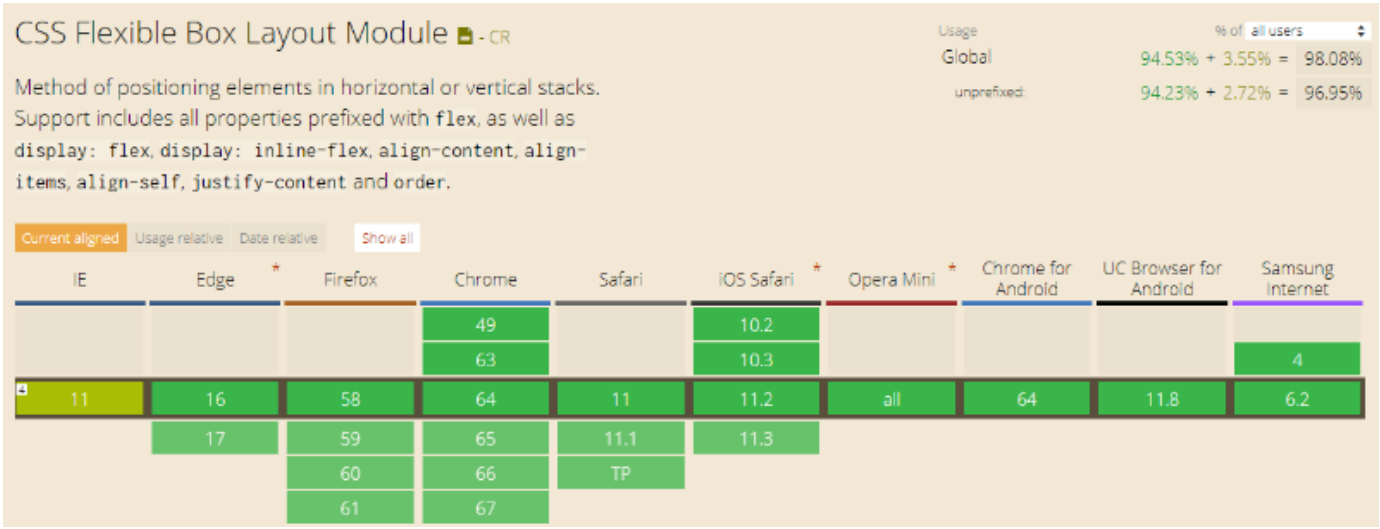
Os dejamos este pequeño ejemplo en Codepen, para que podais ver de primera mano como funciona este Feature Query. Recordad que solo vais a ver la letra inicial roja si entráis desde Safari!

Feature Queries en CSS - initial-letter
A PEN BY fonso_sm

Run Pen

@supports not, o por qué no deberíamos usarlo

Veamos con más detenimiento porque deberíamos obviar el uso de `@support not`, cuando a primera vista puede parecer una buena idea. Con `@support not`, lo que **le estamos diciendo al navegador es que, si no soporta cierta propiedad CSS, utilice el código que tenemos dentro del Feature Query**. Perfecto, ya que podemos agrupar todo el código CSS que sabemos que van a leer hasta los navegadores menos actualizados, y el resto, lo tenemos fuera. Pero claro, no hemos tenido en cuenta una variable en esta ecuación.



Vamos a imaginar un pequeño ejemplo. Como vemos en la imagen de arriba, **la propiedad CSS Flexbox es compatible con prácticamente todos los navegadores**, incluido IE11, Blackberry y IE Mobile. Por lo que perfectamente podríamos decidir en usar CSS Grid en el diseño de una web, y hacer uso de Flexbox para todos aquellos navegadores que se nos escapen. Si quieres más info sobre Flexbox, no te pierdas el artículo que escribimos sobre él.

```
body {
    display: grid;
}

@supports not (display: grid) {
    body {display: flexbox;}
}
```

En principio parece todo correcto. Si mi navegador es compatible con CSS Grid, va a aplicarlo de forma correcta. Y si no es compatible con CSS Grid, pues aplicará Flexbox. **Pero, ¿que pasa si nuestro navegador es compatible con Flexbox, pero no es compatible con Feature Querie?** Es el caso de IE11, por ejemplo. Pues tan sencillo que al ir leyendo nuestra hoja de estilos, IE11 no va a aplicar CSS Grid, como ya sabemos. Pero el problema es que, como tampoco es compatible con los Feature Queries, tampoco va a poder aplicarlo. Por lo que ni se va a usar CSS Grid, como ya hemos

previsto, pero es que tampoco va a poder aplicar Flexbox, ya que *no sabe leer* todo lo que hay dentro del Feature Query. Y este es el problema. **Hemos creado unas líneas de código pensando específicamente en los navegadores menos actualizados, y quizás por un incorrecto uso del Feature Query, no se va a aplicar.** Fracaso absoluto.

Por lo que la disposición correcta que deberíamos tener es:

```
body {
    display: flexbox;
}

@supports (display: grid) {
    body {
              // código para nuevos navegadores
             }
}
```

Con esto lo que queremos mostrar son las cuatro situaciones a las que nos enfrentamos al usar Feature Queries.

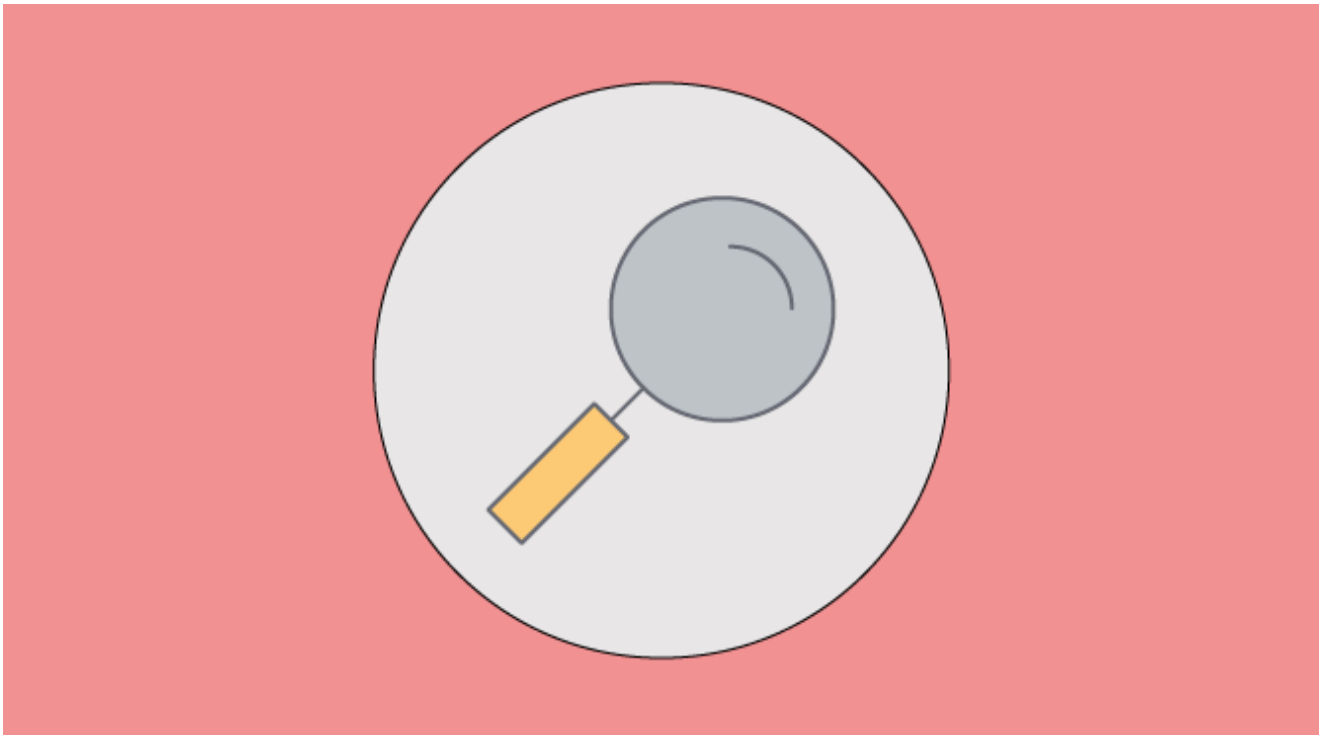
- **Si el navegador es compatible con Feature Query y con el selector**, el código que está dentro se va a aplicar de forma correcta.
- **Si el navegador es compatible con Feature Query pero no con el selector**, el código que está dentro no se va a aplicar. Pero no pasa nada, porque en ese navegador va a usar el código que no está dentro del Feature Query.
- **Si el navegador no es compatible con Feature Query ni con el selector**, de nuevo, el código que tiene dentro no se va a aplicar. Es un caso parecido al anterior.
- **Si el navegador no es compatible con Feature Query pero si con el selector**, el código que está dentro no se va a aplicar. Por lo que estamos creando unos estilos para ese navegador compatible con el selector, pero el Feature Query le impide leerlo. Y como todo lo que está fuera del Feature Query tampoco lo puede leer, definitivamente no estamos consiguiendo nada.

En definitiva, **Feature Query es una propiedad muy útil para estos momentos de transición y actualización de navegadores**, en los que surgen nuevos métodos para trabajar con nuestros diseños web, pero que se van implantando poco a poco. Así, por un lado, estamos usando las herramientas más actuales para trabajar y poder hacer una web mejor. Y por otro, no dejamos de lado a aquellos usuarios que, por los motivos que sean, aun no pueden disfrutar al navegar de estas nuevas opciones.

No te puedes perder...

Media Queries en CSS ¿Cómo funcionan?

Cómo se hace una web: Qué es CSS [Parte 2]



Cómo inspeccionar el CSS de WordPress con tu navegador

Deja un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario *

Nombre *

Email *

Sitio Web

Recibir un email con las últimas noticias y recursos gratuitos de SiloCreativo

Recibir un email con los siguientes comentarios a esta entrada.

Publicar comentario