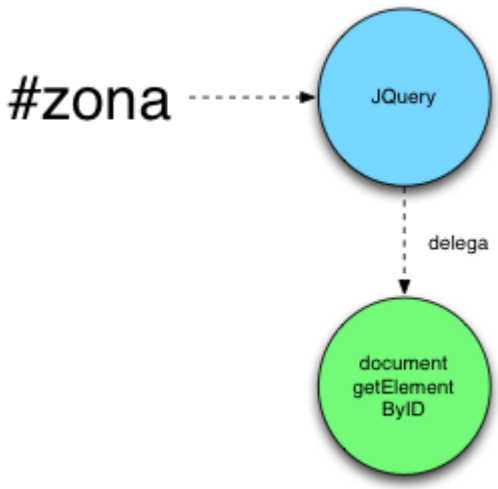
Hoy voy a hablar de como funciona el motor de selectores de JQuery que se denomina JQuery Sizzle Engine . Este motor se encarga de a partir de un selector dado seleccionar las diferentes selecciones en la página HTML de la forma mas óptima posible. Para entenderlo mejor vamos a revisar el siguiente bloque de código.

```
$(document).ready(function () {

$("#zona").css("color","blue");
 });
```

Selectores de ID y Clase

Es un código muy sencillo que JQuery tiene que procesar .¿Ahora bien como lo hace exactamente? . Bueno en este primer ejemplo todo es sencillo JQuery analiza la expresión y al darse cuenta que es un selector de Id (utiliza expresiones regulares para ello). Delega en el método document.getElementById("zona") de Javascript que como todos sabemos selecciona un único nodo por Id del documento.

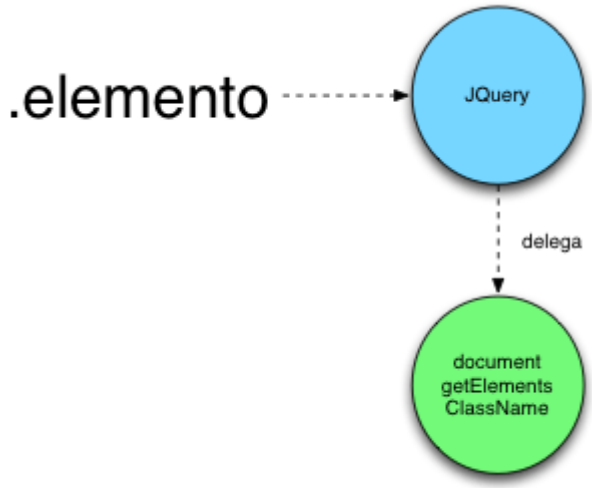


Algo similar sucederá si utilizamos el siguiente selector :

```
$(document).ready(function () {

$(".elemento").css("color","blue");
 });
```

En este caso se trata de un selector de clase . Por lo tanto JQuery después de analizarlo podrá delegar en el método document.getElementsByClassName().

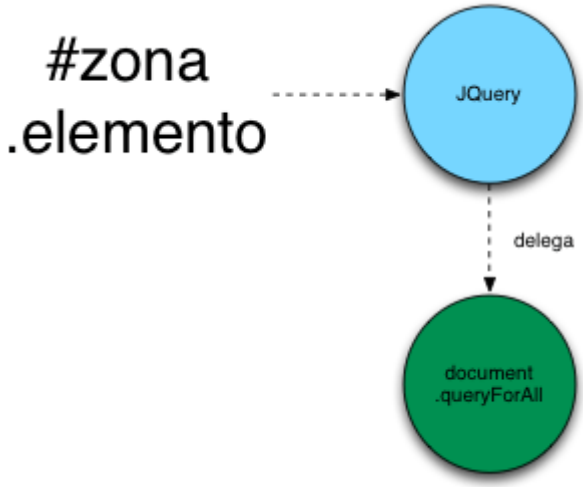


Algo similar ocurre con document.getElementsByTagName(). ¿Ahora bien que pasará si lo que tenemos que seleccionar es algo mas complejo como por ejemplo lo siguiente? .

```
$(document).ready(function () {
$("#zona .elemento").css("color","blue");
 });
```

El método querySelectorAll

Bueno si revisamos las funciones anteriores rapidamente nos damos cuenta que no encajan .La primera selecciona por ID y la segunda por ClassName. Ninguno de los dos métodos soporta una expresión compuesta. Para solventar este problema JQuery delega en otro método de Javascript document.querySelectorAll("#zona .elemento"). Este método soporta que se le pase cualquier selector válido de CSS. De esta forma solventamos el problema.

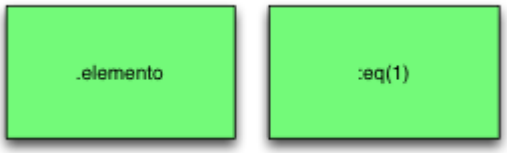


El motor Sizzle

Una vez hemos explicado como JQuery delega en Javascript para el uso de los distintos métodos del API nos queda un problema . ¿Que sucede con selectores mas complejos que no estan soportados por Javascript pero si por JQuery?. Un ejemplo de esta casuistica es el siguiente selector eq(1).

```
$(document).ready(function () {
$(".elemento :eq(1)).css("color","blue");
 });
```

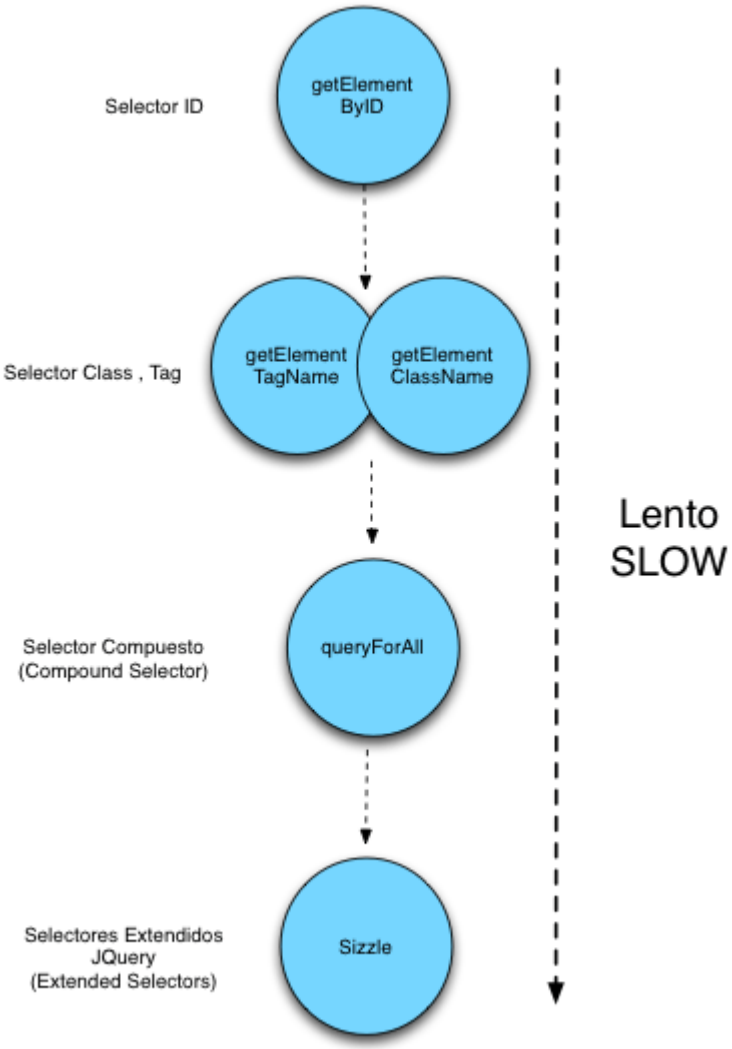
Es en este caso en el que JQuery no puede usar ninguna de las funciones del objeto document de forma directa .No le queda mas remedio que delegar en un motor especial que tiene diseñado y se denomina Sizzle. Este motor evaluará el selector por complejo que sea y lo partirá en bloques para poder abordar la busqueda que le hemos pedidos como se muestra a continuación



Una vez hecho esto JQuery utilizará su método find del motor Sizzle para leer la expresión de derecha a izquierda y ir localizando nodos del arbol DOM que encajen . Así pues en este caso buscará todos los nodos que están en la segunda posición (recordemos eq comienza en 0) de un grupo de nodos.Una vez los tiene seleccionados buscará cuales de estos tienen como padre directo un class ".elemento" y así sucesivamente. Sin embargo esta busqueda será mas lenta que las anteriores.

JQuery Rendimiento

Una vez que hemos comentado estas cosas sobre como JQuery funciona , es mas facil encajar las ideas habituales de que entre los selectores pueden existir gran diferencia a nivel de rendimiento y que hay que andarse con ojo a la hora de usarlos ya que de definir un selector bueno basado en #id a otro similar pero que JQuery tenga que bajar a nivel de motor Sizzle hay mucha diferencia .La siguiente imagen resume un poco lo que hemos comentado.



Para ampliar información sobre estos temas recomiendo los siguientes enlaces en Ingles .

Links interesantes sobre Sizzle

http://blog.jquery.com/2012/07/04/the-new-sizzle/
http://blog.bigbinary.com/2010/02/15/how-jquery-selects-elements-using-sizzle.html

Selectores extendidos de JQuery donde Sizzle se aplica

http://api.jquery.com/category/selectors/jquery-selector-extensions/