JavaScript Console Profile es un concepto interesante ya que en muchas ocasiones nos encontramos con la necesidad de medir el rendimiento de nuestro código de JavaScript. Si hay un lenguaje de programación que tiene sus trucos a la hora de implementar el código es JavaScript. Vamos a ver un ejemplo sencillo de una situación de este estilo usando JQuery para ello partiremos del siguiente código de HTML y Javascript.

```
<html>
<head>
  <script type="text/javascript" src="jquery-3.3.1.min.js">
  </script>
  <script type="text/javascript" src="001.js">
</script>
</head>
<body>
</body>
</html>
```

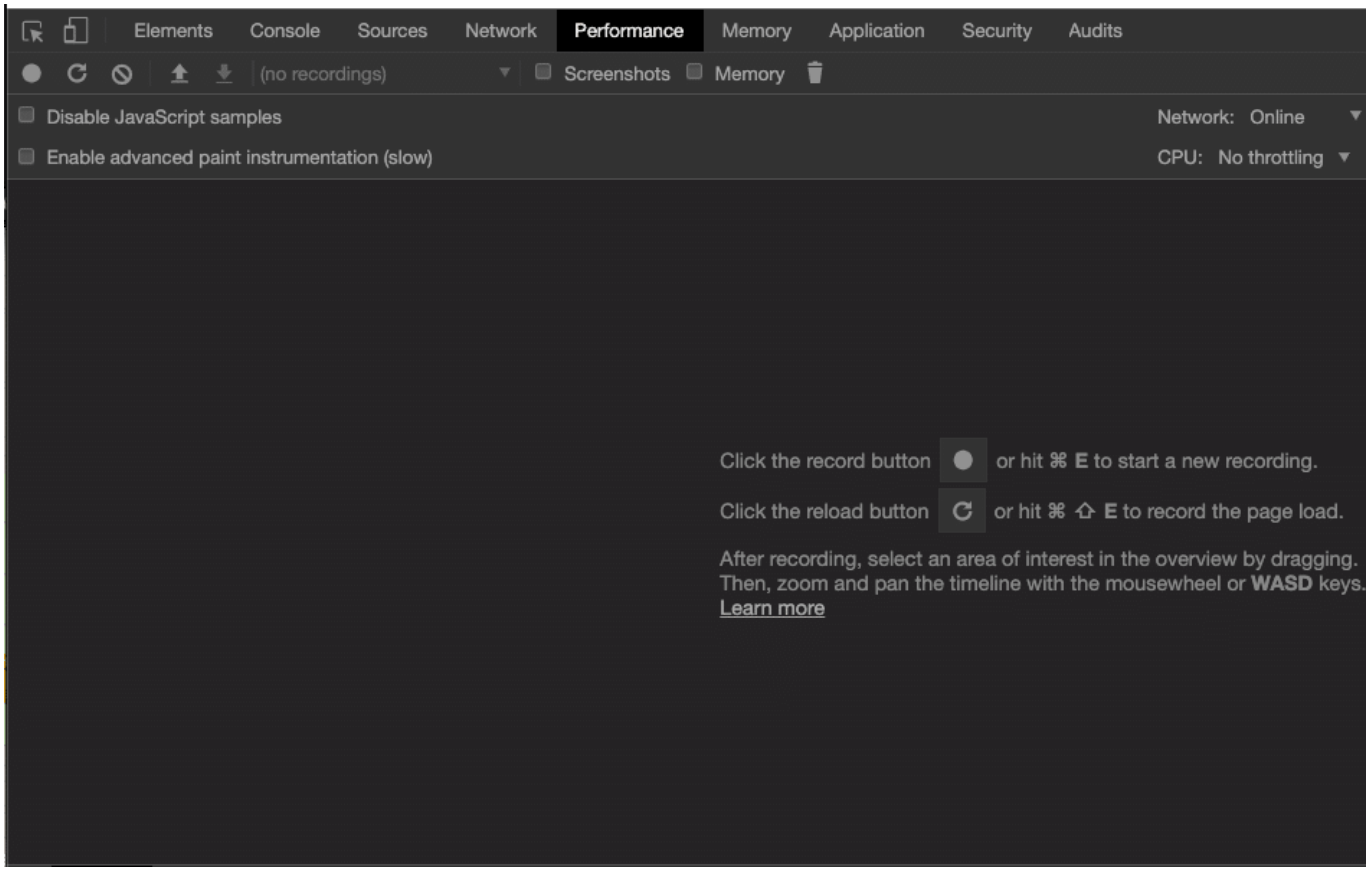
```
$(document).ready(function() {

  for(var i=0;i<10000;i++) {

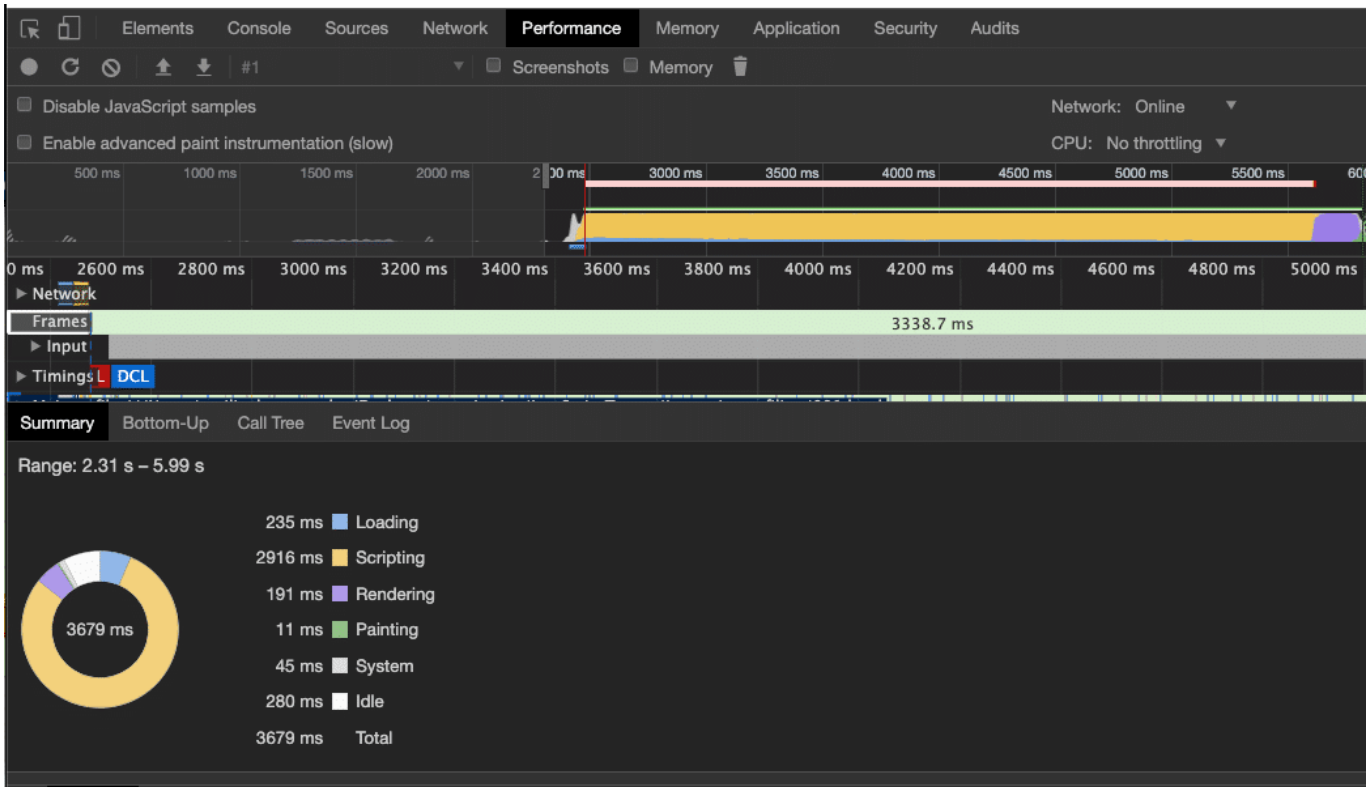
    $("body").append("<p>hola</p>");
  }


})
```

En este caso el código es trivial usa jQuery para generar 10000 párrafos en la página . El código es tan sencillo que no parece que pueda ser objeto de mejora . Vamos cargar esta página en el navegador . Pero vamos a realizar una operación un poco especial nos vamos a situar en la pestaña de chrome de performance.



Este pestaña tiene un botón redondo en la parte izquierda superior que nos permite grabar una sesión de ejecución de código . Vamos a pulsarlos y solicitar que se recargue la página. Una vez la página se haya cargado y los párrafos se generen pulsaremos sobre stop y pararemos la grabación. Automáticamente recibiremos un gráfico de resultado:



Este gráfico nos muestra en que operaciones se ha gastado el tiempo a nivel de JavaScript y podemos ver con claridad que el scripting ha consumido prácticamente el 90 %. Por lo tanto si queremos que esta aplicación funcione mejor tendremos que ver si el código de JavaScript es mejorable así de sencillo . Vamos a probar con esta versión:

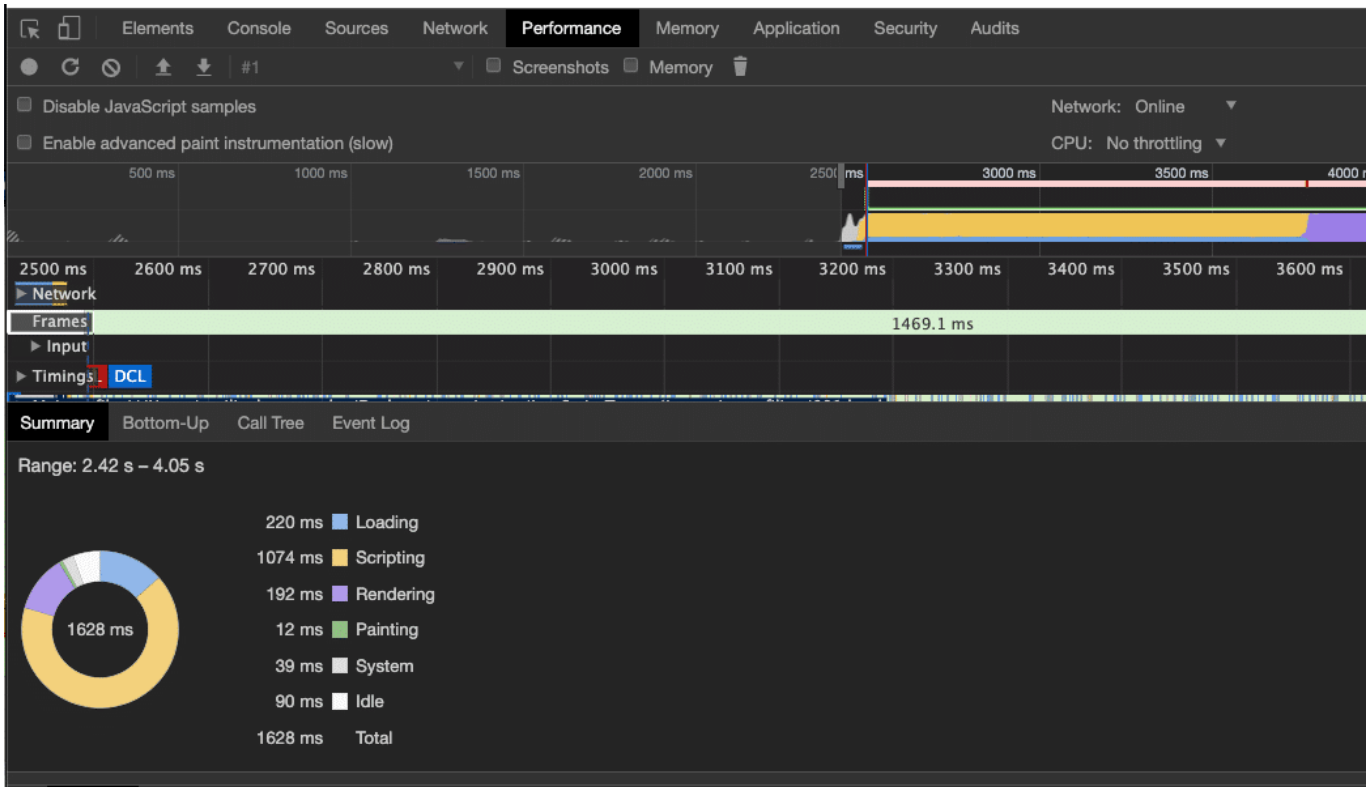
```
$(document).ready(function() {
  var $body=$("body")
  for(var i=0;i<10000;i++) {

    $body.append("<p>hola</p>");
  }


})
```

JavaScript Console Profile

Este versión parece casi idéntica pero cachea el selector de jQuery de tal forma que no tengamos que estar seleccionando el body continuamente parece un cambio sin importancia . Vamos a realizar un profile de este código y ver el resultado:



El resultado es sorprendente simplemente con ese cambio hemos triplicado el rendimiento y la sesión de profile nos ayuda mucho a entender esto de un primer vistazo. Todavía lo podemos afinar más y cambiar el código por :

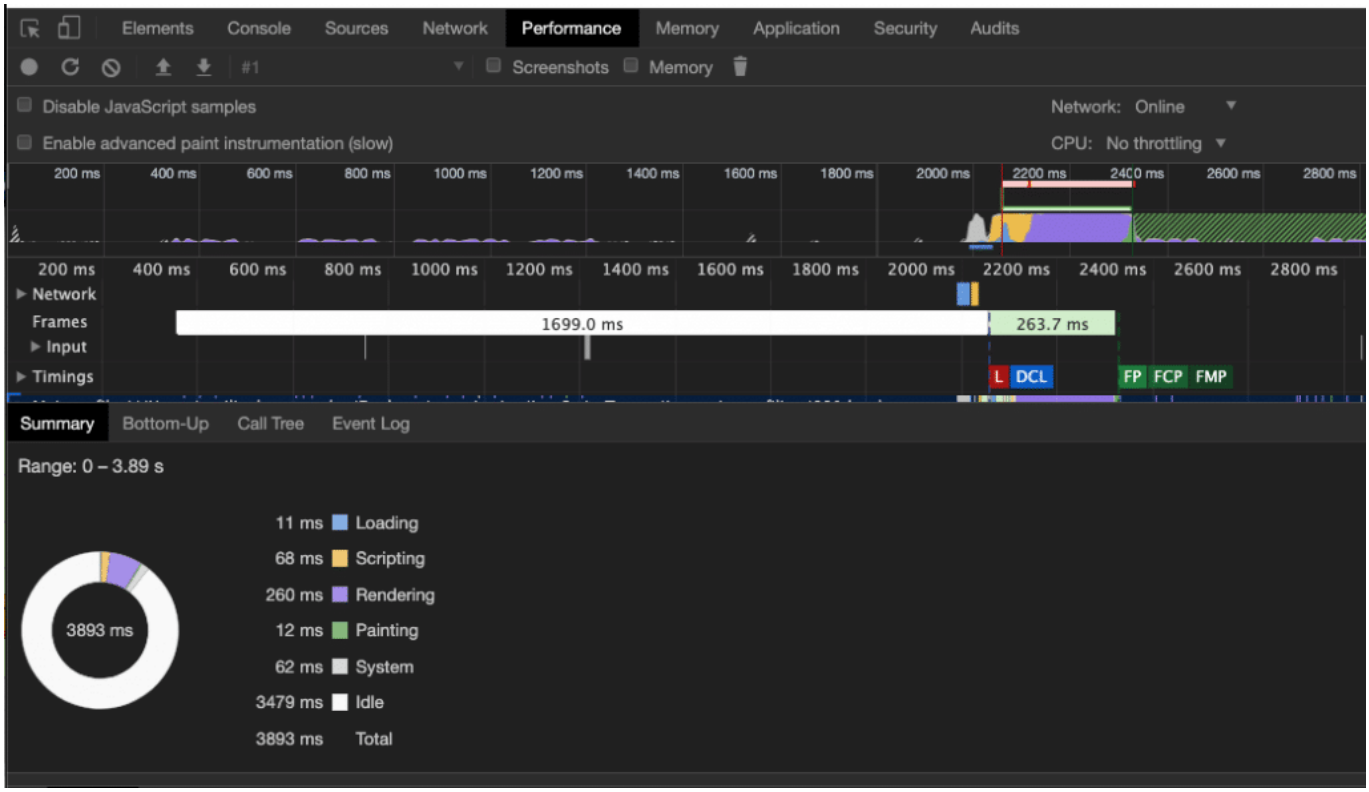
```
var texto="";

$(document).ready(function() {
  var $body=$("body")
  for(var i=0;i<10000;i++) {
```

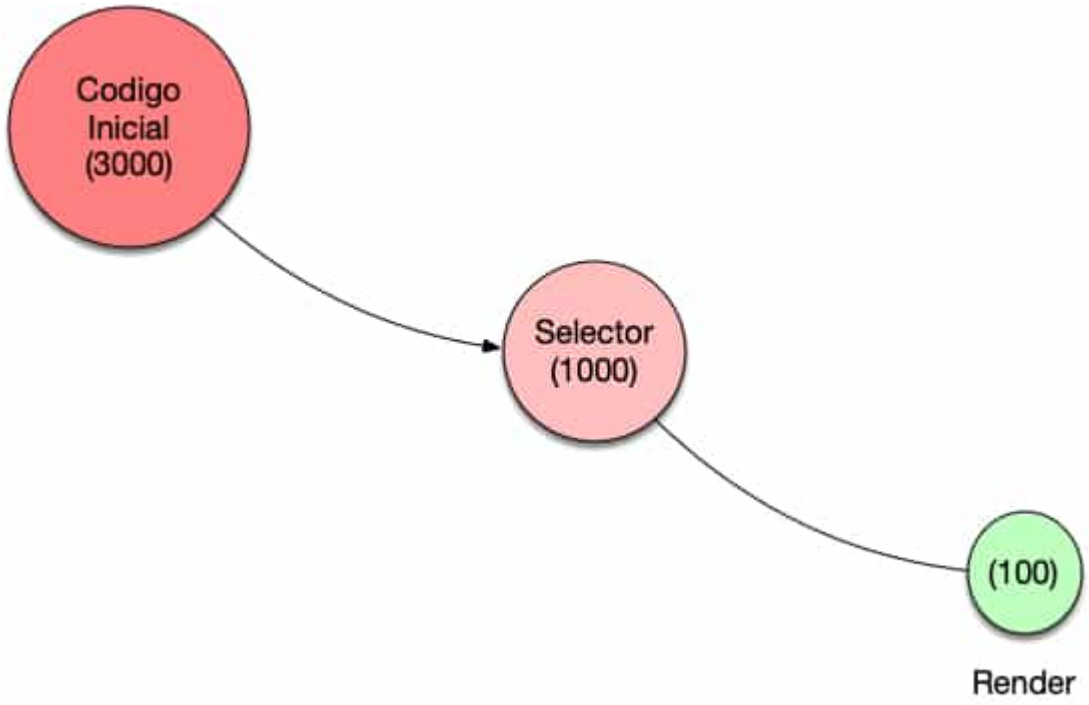
```
    texto+="<p>hola</p>";
  }
$body.append(texto);

})
```

El resultado será demoledor ya que suma todas las cadenas de texto y finalmente renderiza en la página y no esta renderizando continuamente.



Las sesiones de profiling nos pueden ayudar a resolver muchas dudas y gestionar mejor el rendimiento de JavaScript. En este caso hemos realizado 2 cambios con JavaScript Console Profile . El primero evitar que jQuery genere un selector de forma continua. El segundo evitar que jQuery renderize continuamente la página y pase a renderizarla solo al final.



Otros artículos relacionados

1. Promise chaining en JavaScript
2. JavaScript Arrow Function y scopes
3. JavaScript Array Spread Operator y simplificaciones
4. Utilizando JavaScript template String
5. JavaScript Console