

```

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

```

```

<link rel="stylesheet" href="http://localhost/css.css" type="text/css">
<script type="text/javascript" src="http://localhost/javascript.js"></script>
<script type="text/javascript">
(function(){
  onLoaded: function(request) {
    if (request.name == 'log_error') return;
    log.trace("Ajax.Request: " + (request.name || request.url.substr(0, 30)
      ) + "...");
  },
  onComplete: function(request) {
    if (request.name == 'log_error') return;
    log.trace(request, e);
  },
  onException: function(request, e) {
    if (fatal(request.url + ' : ' + e.name + ' | ' + e.message + ' | ' +
      .stack));
  }
})

```

# Índice

<b>1.Scripts en el head.....</b>	<b>3</b>
<b>2.Scripts en el body.....</b>	<b>4</b>
<b>3.Conclusión.....</b>	<b>5</b>
<b>4.Webgrafía.....</b>	<b>8</b>

# 1.Scripts en el head

Es un tema que puede generar bastante controversia. Una buena práctica es integrarlo antes del cierre de la etiqueta `</body>` , pero que a veces sería algo imposible cuando necesitamos modificar el DOM cuando se está cargando la página.

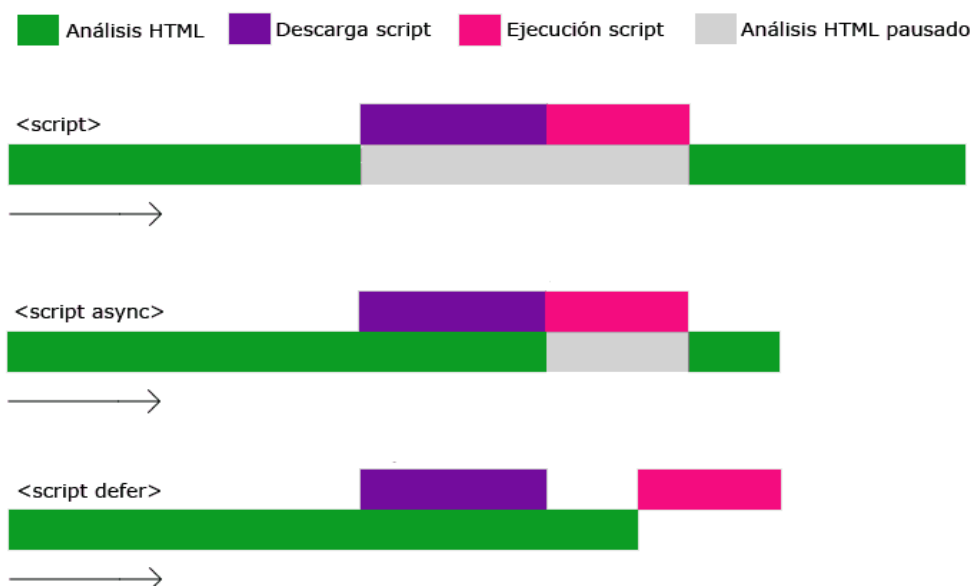
En esos casos siempre es recomendable dejarlo dentro de la etiqueta `<head>` es 100% práctico siempre y cuando le adviertas al script que se cargue después de que el DOM esté totalmente cargado, lo puedes hacer con el evento `load` del objeto `window` en el código JavaScript:

```
// código Js
window.onload = function() {
  /* Aquí la instancia a eventos desde elementos que ya
    están cargados dentro de la página */
}
```

Aunque el código JavaScript esté enlazado en el elemento `<head>`, puedes impedir que penalice el rendimiento si utilizas una técnica como la de Google Analytics, que crea el enlace al código JavaScript dinámicamente y lo carga asíncronamente (mediante el atributo `async`).

Otra técnica consiste en utilizar el atributo `defer`, que indica al navegador que el código JavaScript enlazado se puede descargar y procesar después de que la página se haya cargado completamente.

El problema del atributo `defer` es que su comportamiento no es idéntico en todos los navegadores y obliga a ejecutar los scripts en el mismo orden en el que se definen en la página, por lo que en este caso también se pueden producir bloqueos (algo que no sucede por ejemplo con el atributo `async` de la técnica anterior)



## 2.Scripts en el Body

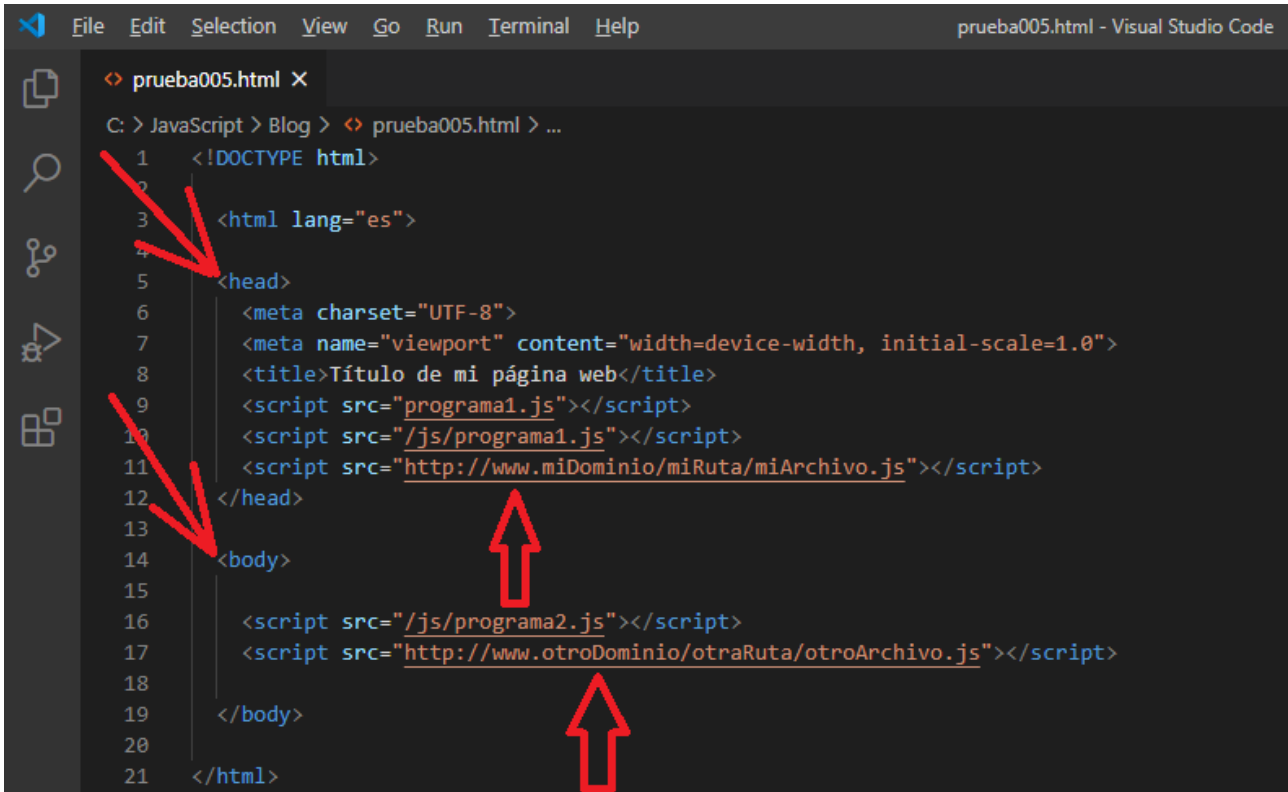
Esto se hace así porque generalmente el Javascript tiene que ponerse en marcha cuando el navegador ha procesado el HTML. No sirve de mucho que el navegador procese el Javascript cuando el usuario todavía no ha podido ver el contenido de la página. Además, el Javascript muchas veces debe operar con los elementos de la página, para modificarlos dinámicamente y ésto se tiene que hacer cuando los elementos han sido cargados por el navegador. Por todo ello se suele poner en la mayoría de los casos antes de cerrar la etiqueta `<body>`.

Poner los scripts en el `<head>` puede hacer que el navegador se entretenga descargando Javascript que no necesita ejecutar previamente a la carga del contenido de la página, lo que puede reducir el tiempo de carga y por lo tanto el rendimiento. Dependiendo del tamaño del código Javascript y del número de archivos que se deban cargar, esto puede ser más o menos representativo, pero la optimización de colocar los scripts antes del `</body>` no suele estar de más.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Ejemplo de Uso QUnit</title>
6      <link rel="stylesheet" href="qunit-git.css">
7      <script src="jquery-1.9.1.min.js"></script>
8      <script src="qunit-git.js"></script>
9
10 </head>
11 <body>
12     <div id="qunit"></div>
13     <script type="text/javascript">
14         test("sumar", function(){
15             expect(2);
16             var a = Math.floor((Math.random()*10)+1);
17             var b = Math.floor((Math.random()*10)+1);
18
19             var r = Sumar(a,b);
20
21             ok(r > 10, "Resultado mayor de 10");
22             equal(r, 30, "Es igual 30");
23         });
24
25         function Sumar(a,b){
26             return a+b;
27         }
28     </script>
29 </body>
30 </html>
```

# 3.Conclusión

Si la funcionalidad es crítica en el head, sino en el cierre del body, si el javascript consiste sólo en un par de líneas puede ir inline para optimizar la carga.



```
prueba005.html X
C: > JavaScript > Blog > <> prueba005.html > ...
1  <!DOCTYPE html>
2
3  <html lang="es">
4
5  <head>
6    <meta charset="UTF-8">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>Título de mi página web</title>
9    <script src="programa1.js"></script>
10   <script src="/js/programa1.js"></script>
11   <script src="http://www.miDominio/miRuta/miArchivo.js"></script>
12 </head>
13
14 <body>
15
16   <script src="/js/programa2.js"></script>
17   <script src="http://www.otroDominio/otraRuta/otroArchivo.js"></script>
18
19 </body>
20
21 </html>
```

# 4. Webgrafía

La información usada para realizar este trabajo ha sido sacada de las siguientes páginas web:

- <https://blog.desafiolatam.com/javascript-en-el-head-o-en-el-cierre-del-body-estas-equivocado/>
- <https://cybmeta.com/diferencia-async-y-defer>
- <https://desarrolloweb.com/faq/colocar-scripts-javascript-head-body>
- <https://es.stackoverflow.com/questions/25088/cuál-es-el-mejor-lugar-para-colocar-los-tag-scripts-src-en-html>