**Formas de incluir código Javascript**

**inline (script)**

<script src="script.js">

*Puede bloquear la fase de parseo del HTML dependiendo de su posición en el HTML*

**inline (como manejador de eventos)**

<button onclick="javascript:alert('Hola');">haz click</button>

**async**

<script async src="script.js">

*Bloquea el evento de carga “onload”*

*La fase de ejecución del código puede bloquear el parseo del HTML*

**defer**

<script defer src="script.js">

*Espera a que acabe el parseo del HTML para ejecutarse.*

**link**

<head>

<link rel="preload" as="script" href="map.js" media="(min-width: 601px)">

</head>

*No bloquea el evento “onload” del Navegador a no ser que sea requerido por otro recurso que sí que bloquee el evento “onload”*

*Comprobar el soporte en los distintos navegadores*

**Términos**

**Parseo**

El runtime transforma el código Javascript a una estructura llamada **AST** (Árbol de Sintaxis abstracta) con el que puede operar.

Consiste en leer un fichero de texto y con una gramática definida (expresadas como una autómata), transformar esa información en un árbol (que al final es un grafo dirigido).

Ej de una gramática BNF

<list> ::= "<" <items> ">" ;

<items> ::= <items> " " <item> | <item> ;

<item> ::= "foo" | "bar" | "baz" ;

**Contexto de ejecución**

Al abrir cualquier pestaña de un navegador, lo primero que se crea es el contexto global (el primer contexto de ejecución **siempre**), creando un **objeto global** (**window** en navegadores, **global** en NodeJS).

A partir de aquí, **cada vez que se ejecuta una llamada, se crea un nuevo contexto de ejecución**.

Para “llevar la cuenta” de la cadena de llamadas (creaciones de contextos de ejecución), el navegador mantiene una pila (**Stack Context**).

Cuando hacemos debugging , y paramos la ejecución de un script en un determinado momento, podemos identificar el contexto de ejecución en el que estamos actualmente : es el contexto **Local**.

La creación de un contexto de ejecución , conlleva internamente dos fases:

**Fase de preparación del contexto de ejecución**

* Se setean todas las variables definidas en el contexto léxico a “undefined”
* se establece el valor de **this**
* se guarda una referencia al contexto del que se viene.

**Fase de ejecución del contexto Local**

* ejecución del código de ese contexto
* una vez acabado, se destruye el contexto de ejecución (se desapila del Stack Context).

**this**

Es una referencia al objeto desde el que se llama a una función.

var greeting = "Hola contexto global!";

console.log(greeting); *// Hola contexto global!*

function greet () {

var greeting = "Saludo desde greet";

console.log(greeting);

var obj = {};

obj.grita = function() {

console.log("TE ESTOY SALUDANDO!");

}

obj.grita();

}

greet() *// Saludo desde greet*

console.log(greeting); *// Hola contexto global!*

El valor de this se establece en la fase de creación del contexto de ejecución.

Puede ser alterado por funciones como **bind, call, apply**

**Lexical this**

Lo veremos en las **funciones “flecha”** , una novedad que se incluyó ES6 o ES2015.

Por el momento sólo diremos que **this** se comporta en las **funciones “flecha”** de manera diferente a cómo hemos comentado

**Referencias interesantes**

[**AST Explorer**](https://astexplorer.net/)  
[**JointJS**](http://resources.jointjs.com/demos/javascript-ast)  
[**MetaJS Javascript Execution visualization Tool**](http://int3.github.io/metajs/)

[**Preload: What’s good for - Smashing Magazine**](https://www.smashingmagazine.com/2016/02/preload-what-is-it-good-for/)

[**Can I use support link rel attribute**](http://caniuse.com/#search=link-rel)

[**Asynchronous vs Deferred JavaScript**](https://bitsofco.de/async-vs-defer/)

[**Marja Hölttä: Parsing JavaScript - better lazy than eager? | JSConf EU 2017**](https://www.youtube.com/watch?v=Fg7niTmNNLg&t=4s)

[**Standard Library - Wikipedia**](https://en.wikipedia.org/wiki/Comparison_of_layout_engines_(ECMAScript)#Standard\_Library)

**EXTRAS**

<https://www.youtube.com/watch?v=PkOBnYxqj3k> .

[You might not need jquery](http://youmightnotneedjquery.com/" \t "_blank)

[Critical Rendering Path](https://www.youtube.com/watch?v=PkOBnYxqj3k" \t "_blank)

[vídeo](https://youtu.be/Bv_5Zv5c-Ts?t=760)

[history api](https://developer.mozilla.org/en-US/docs/Web/API/History)