Understanding HTTP Requests/Responses regarding the DOM and script execution

Asked 14 years, 3 months ago Modified 14 years, 3 months ago

Viewed 3k times

5

I've been away from web design/development for a long, long now and have recently begun to get back into it. I started off doing things just to make them work, and now that I'm getting back into it, I would like to understand things a bit more clearly - including when the DOM is requested by the browser to when it is fully loaded, and the difference between script placement at the top and bottom of a page.

I realise that this is more a post for http://doctype.com, but I figured I would get a more technical answer from here. I would also like to have made this a community wiki, but I don't have enough points...as yet.

Please feel free to correct me here - My questions/assumptions:

1. When the browser makes a request for the page, the server responds with a Document Object that contains the hierarchy/order of scripts, css and html - correct?

2. Once received, the browser then builds a Document Object Tree - is this when the DOM is ready or when it starts rendering elements on the page in a browser?
3. In that regard, what is the difference between "when the DOM is loaded" and "when the DOM is ready"?
4. Is there any difference between placing (java)script at the top (in the head tag) or at the bottom (before `</body>` tag)?
5. Is there an DOM event that fires when all assets (css, images, javascripts, etc) are **fully** loaded by the browser? I ask this as sometimes I might have a background image still being loaded by the browser, and well before it can complete, my Javascript animations already start executing.

Thank you for taking the time to read this, and i look forward to your responses!

html **dom** **execution**

Share

Improve this question

Follow

asked Aug 30, 2010 at 21:34

magz

175 ● 2 ● 8

2 Answers

Sorted by: Highest score (default)

1. Browsers make conditional and unconditional requests to the server. (The Server Responds to Inquiry and the Client Renders responses...there is a limited throughput of information to and from the user *(Privacy Settings & Etc.)*

Unconditional Requests:

> An unconditional request is made when the client browser does not have a cached copy of the resource available locally. In this case, the server is expected to return the resource with a HTTP/200 OK response. If the response's headers permit it, the client may cache this response in order to reuse it later. If the browser later needs a resource which is in the local cache, that resource's headers are checked to determine if the cached copy is still fresh. If the cached copy is fresh, then no network request is made and the client simply reuses the resource from the cache.

Conditional Requests:

> If the browser later needs a resource which is in the cache, but that response is expired (older than its max-age or past the Expires date), then the client will make a conditional request to the server to determine whether the previously

> cached response is still valid and should be reused. The conditional request contains an If-Modified-Since and/or If-None-Match header that indicates to the server what version of the content the browser already has in its cache. The server can indicate that the client's copy is still fresh by returning HTTP/304 Not Modified headers with no body, or it can indicate that the client's copy is stale by returning a HTTP/200 OK response with the new version of the resource.

2. The Document Object Model - is a model of information as it pertains to a browsers request/response. In many ways ECMA/Javascript was born as a direct relation accessing page elements and became the default DOM Library solution for many of the browser objects (document.frm etc). However, implementation and support is non-unified and sparse across all browsers.
3. Essentially, the DOM Being loaded is just a definition to say that the page is being rendered and the content has/is being invoked. This is something that cannot be counted on in regards to accessing DOM Object so, it is unfavorable method of interaction. Generally, you should use the ready status which means that the DOM is in a ready and waiting state of request.
4. Yep loads of difference - execution order of the client side code means where it will be loaded. Load with

the entire page object or load after all content.

5. Yes and No. :) No Real Guarantee but, with javascript/jquery used you can use the onload event of the body to identify all elements of the page successfully loaded. This is a trick question btw. No real answer to my knowledge unless, I am mistaken to what you asked...

Share Improve this answer Follow

answered Aug 30, 2010 at 22:14

community wiki
user384929

2 Thank u Joe, this does help quite a bit. I feel like a noob going thru all of this again...but I'd rather feel like a noob than to continue on not knowing fundamentals! So with Javascript, if I have script at the top of a page that accesses or modifies some HTML element, does it run when the Page (and all assets are loaded) or while its been fetched? – magz Aug 30, 2010 at 23:56

1 We can only use methods when loaded i dnt know if it hapoens asynchronously – blackHawk Nov 18, 2016 at 5:30

1 For faster page load we put scripts on the bottom of body tags – blackHawk Nov 18, 2016 at 5:30

3

1. Script tags inline with your markup are executed synchronously with the browser's processing of that markup (except, see #2), and so if -- for instance -- those tags reference external files, they tend to slow



down the processing of the page. (This is so the browser can handle document.write statements, which change the markup they're processing.)

2. Script tags with the defer attribute may, on some browsers, not be executed until after the DOM has been fully rendered. Naturally these can't use document.write. (Similarly there's an async attribute that makes the script asynchronous, but I don't know much about it or how well it's supported; details.)
3. Script tags in content you assign to elements after DOM load (via innerHTML and similar) are not executed at all, barring your use of a library like jQuery or Prototype to do it for you. (With one exception pointed out by Andy E: On IE, if they have a defer attribute, it will execute them. Doesn't work in other browsers.)
4. If you append an actual script element to the document via Element#appendChild, the browser executes that script immediately (and you can happily remove the element if you like, the script has already been executed and processed). (You would normally append those to the head element, but in practice it doesn't really matter.)
5. Script inside event handlers on attributes () rather than in a script tag is executed when the relevant event occurs.

Original Author - from: [When does the browser execute Javascript? How does the execution cursor move?]

To Answer your last question - well sometimes...

Share Improve this answer

Follow

edited May 23, 2017 at 12:03

Community Bot

1 • 1

answered Sep 2, 2010 at 14:30

user384929