Get protocol, domain, and port from URL

Asked 7 years, 11 months ago Active 21 days ago Viewed 291k times

273

53

I need to extract the full protocol, domain, and port from a given URL. For example:

```
https://localhost:8181/ContactUs-1.0/contact?lang=it&report_type=consumer
>>>
https://localhost:8181
```

javascript url dns protocols port

edited Feb 16 '17 at 5:57
codeforester
20.1k 8 44 75

asked Aug 4 '11 at 12:38
yelo3
2,023 5 19 21

9 For those readers looking for an answer where the URL is not the current location, look below the accepted answer – Guy Schalnat Jun 8 '15 at 19:56

16 Answers

135

first get the current address

```
var url = window.location.href
```

Then just parse that string

```
var arr = url.split("/");
```

your url is:

```
var result = arr[0] + "//" + arr[2]
```

Hope this helps

answered Aug 4 '11 at 12:43

wezzy
4,790 3 25 39

191 -1, location.protocol is less hacky – naugtur May 28 '13 at 7:59

8 This works with URL string where `location` object is not available(js outside browser!) – Thamme Gowda Nov 26 '14 at 6:03

David Calhoun's answer uses the built-in parser (like `location`) but can be used for **any** url. Check it out it's neat. – Stijn de Witt Nov 14 '16 at 21:33

4 Or just turn it into a one-liner: `window.location.href.split('/').slice(0, 3).join('/')` – Ryan McGeary May 16 '17 at 19:35

3 window.location.origin – int soumen Nov 30 '18 at 19:03

527

```
var full = location.protocol+'//'+location.hostname+(location.port ? ':'+location.port:
'');
```

answered Aug 4 '11 at 12:45

Shef
38.2k 12 70 85

3 @Randomblue What about it? You will get `about://` . However, I am curious to know, what would be the use case for `about:blank` ? I am not sure if any browser injects plugin resources in `about:blank` , but seems like that could be the only use case. – Shef Sep 2 '12 at 6:27

37 I find my javascript never loads when I visit about:blank... – toxaq Nov 12 '12 at 22:55

2 This doesn't work at all if you have a URL string, right? (i.e. you need to be *at* `location` for this to work) – Nick T Oct 5 '15 at 22:57

1 Sorry for the late reply, @NickT. Yes, it does't do that. Please, use the **nice solution provided by David** for that. – Shef Oct 7 '15 at 16:37

13 Can't you use `location.host` instead of `location.hostname + location.port` ? – c24w Sep 20 '16 at 14:28

163

None of these answers seem to completely address the question, which calls for an arbitrary url, not specifically the url of the current page.

Method 1: Use the URL API (caveat: no IE11 support)

You can use the URL API (not supported by IE11, but available everywhere else).

This also makes it easy to access search params. Another bonus: it can be used in a Web Worker since it doesn't depend on the DOM.

```
const url = new URL('http://example.com:12345/blog/foo/bar?startIndex=1&pageSize=10');
```

Method 2 (old way): Use the browser's built-in parser in the DOM

Use this if you need this to work on older browsers as well.

```
//  Create an anchor element (note: no need to append this element to the document)
const url = document.createElement('a');
//  Set href to any path
url.setAttribute('href', 'http://example.com:12345/blog/foo/bar?
startIndex=1&pageSize=10');
```

That's it!

The browser's built-in parser has already done its job. Now you can just grab the parts you need (note that this works for both methods above):

```
//  Get any piece of the url you're interested in
url.hostname;  //  'example.com'
url.port;      //  12345
url.search;    //  '?startIndex=1&pageSize=10'
url.pathname;  //  '/blog/foo/bar'
url.protocol;  //  'http:'
```

Bonus: Search params

Chances are you'll probably want to break apart the search url params as well, since '?startIndex=1&pageSize=10' isn't too useable on its own.

If you used Method 1 (URL API) above, you simply use the searchParams getters:

```
url.searchParams.get('startIndex');  // '1'
```

Or to get all parameters:

```
Array
    .from(url.searchParams)
    .reduce((accum, [key, val]) => {
        accum[key] = val;
        return accum;
    }, {});
// -> { startIndex: '1', pageSize: '10' }
```

If you used Method 2 (the old way), you can use something like this:

```
// Simple object output (note: does NOT preserve duplicate keys).
var params = url.search.substr(1); // remove '?' prefix
params
    .split('&')
    .reduce((accum, keyval) => {
        const [key, val] = keyval.split('=');
        accum[key] = val;
        return accum;
    }, {});
// -> { startIndex: '1', pageSize: '10' }
```

edited Jun 26 at 13:40

answered Oct 17 '14 at 22:12

David Calhoun
4,506 3 20 19

link.protocol gets me a "http:" if i inspect a anker with "google.com" :-(`var link = document.createElement('a'); link.setAttribute('href', 'google.com'); console.log(link.protocol)` – eXe Sep 26 '16 at 12:35

Are you doing that on a `http` page perhaps? If not specified it will 'inherit' from the current location – Stijn de Witt Nov 14 '16 at 21:24

3 This is a fantastic answer and should get more votes, because this answer is not limited to just the *current* location but works for *any url*, and because this answer utilizes the browser's built-in parser instead of building one ourselves (which we can't hope to do as well or as fast!). – Stijn de Witt Nov 14 '16 at 21:26

Thank you for this clever trick! I would like to add one thing: There is both `host` and `hostname`. The former includes the port (e.g. `localhost:3000`), while the latter is only the host's name (e.g. `localhost`). – codener Mar 31 '17 at 11:30

This works well in case of absolute URL. It fails in case of Relative URL and cross-browser. Any suggestions? – Gururaj Aug 4 '17 at 12:42

118

For some reason all the answers are all overkills. This is all it takes:

```
window.location.origin
```

More details can be found here: https://developer.mozilla.org/en-US/docs/Web/API/window.location#Properties

answered May 30 '13 at 18:35

Pijusn
6,533 7 43 72

19 FYI, I'm sure this will be great in the future when all popular browsers have implemented it, *however*, this isn't the case at present: developer.mozilla.org/en-US/docs/Web/API/… At time of writing only recent versions of Firefox and WebKit browsers support the origin property according to my research. – Zac Seth Jul 12 '13 at 16:58

2 Just to complete: location is defined on HTML5 and it implements the `URLUtils` interface which is defined on WHATWG and includes the `origin` attribute. – Ciro Santilli 新疆改造中心996ICU六四事件 Nov 10 '14 at 9:00

5 Hello from 2015.. unfortunately URLUtils still isn't properly implemented across all browsers, according to this compatibility table on MDN. However it does seem that the origin property is slightly better supported than in 2013, it's still not fit for production as it's not implemented properly in Safari. Sorry guys :(– totallyNotLizards Jun 9 '15 at 8:25

Update: Still not supported for many browsers (safari as well) :(:(– Ahmad hamza Apr 20 '16 at 19:12

It does not work in IE as well, it returns "undefined". – Siddhartha Chowdhury Jul 20 '16 at 14:31

50

As has already been mentioned there is the as yet not fully supported `window.location.origin` but instead of either using it or creating a new variable to use, I prefer to check for it and if it isn't set to set it.

For example;

```
if (!window.location.origin) {
  window.location.origin = window.location.protocol + "//" + window.location.hostname +
(window.location.port ? ':' + window.location.port: '');
}
```

I actually wrote about this a few months back A fix for window.location.origin

edited Sep 9 '15 at 11:14
Jeffrey Knight

answered Oct 10 '13 at 0:26
Toby

4,602 5 32 45 3,581 9 36 62

1 This is first time that I know `window.location.origin` is exists. Thank you. ^^ – EThaizone Jo Apr 8 '17 at 9:34

31

host

```
var url = window.location.host;
```

returns `localhost:2679`

hostname

```
var url = window.location.hostname;
```

returns `localhost`

answered Jul 19 '15 at 7:42
Miroslav Holec
2,435 19 20

15

`window.location.origin` will be enough to get the same.

edited Nov 30 '18 at 19:59
probablyup
3,289 1 15 29

answered Nov 30 '18 at 19:04
int soumen
317 2 12

3 This should be the accepted answer... – ebu_sho Dec 10 '18 at 10:04

Thank you ebu_sho. – int soumen Feb 18 at 7:27

1 That solved my problem easily. Thank you @intsoumen – Turker Tunali Mar 25 at 9:49

1 agree! works like magic – YanivN Apr 11 at 12:37

13

> The protocol property sets or returns the protocol of the current URL, including the colon (:).

This means that if you want to get only the HTTP/HTTPS part you can do something like this:

```
var protocol = window.location.protocol.replace(/:/g,'')
```

For the domain you can use:

```
var domain = window.location.hostname;
```

For the port you can use:

```
var port = window.location.port;
```

Keep in mind that the port will be an empty string if it is not visible in the URL. For example:

- http://example.com/ will return "" for port
- http://example.com:80/ will return 80 for port

If you need to show 80/443 when you have no port use

```
var port = window.location.port || (protocol === 'https' ? '443' : '80');
```

edited Aug 28 '17 at 10:23

answered Mar 19 '14 at 13:39

Дамян Станчев
2,031 2 26 48

6

Indeed, *window.location.origin* works fine in browsers following standards, but guess what. IE isn't following standards.

So because of that, this is what worked for me in IE, FireFox and Chrome:

```
var full = location.protocol+'//'+location.hostname+(location.port ? ':'+location.port:
'');
```

but for possible future enhancements which could cause conflicts, I specified the "window" reference before the "location" object.

```
var full = window.location.protocol+'//'+window.location.hostname+(window.location.port
? ':'+window.location.port: '');
```

edited Jun 28 '13 at 16:27

answered Jun 28 '13 at 16:17

cpu
467 4 6

2

```
var http = location.protocol;
var slashes = http.concat("//");
var host = slashes.concat(window.location.hostname);
```

answered Jul 2 '13 at 10:05

Elankeeran
2,839 7 32 55

2

```
var getBasePath = function(url) {
    var r = ('' + url).match(/^(https?:)?\/\/[^/]+/i);
    return r ? r[0] : '';
};
```

answered Sep 9 '16 at 7:14

haipeng
21 4

2 consider explaining your answer. Don't assume the OP can understand the significance of the different parts of your code. – ADyson Sep 9 '16 at 10:00

2

Try use a regular expression (Regex), which will be quite useful when you want to validate / extract stuff or even do some simple parsing in javascript.

The regex is :

```
/([a-zA-Z]+):\/\/([\-\w\.]+)(?:\:(\d{0,5}))?/
```

Demonstration:

```
function breakURL(url){

     matches = /([a-zA-Z]+):\/\/([\-\w\.]+)(?:\:(\d{0,5}))?/.exec(url);

     foo = new Array();

     if(matches){
          for( i = 1; i < matches.length ; i++){ foo.push(matches[i]); }
     }

     return foo
}

url = "https://www.google.co.uk:55699/search?
q=http%3A%2F%2F&oq=http%3A%2F%2F&aqs=chrome..69i57j69i60l3j69i65l2.2342j0j4&sourceid=chrom
8"


breakURL(url);       // [https, www.google.co.uk, 55699]
breakURL();          // []
breakURL("asf");     // []
breakURL("asd://");  // []
breakURL("asd://a"); // [asd, a, undefined]
```

Now you can do validation as well.

edited Jun 2 '17 at 1:06

answered Jun 2 '17 at 1:00

ed9w2in6
121 4

"A valid RFC 3986 URL scheme must consist of "a letter and followed by any combination of letters, digits, plus ("+"), period ("."), or hyphen ("-")." -- stackoverflow.com/a/9142331/188833 (Here's an urn:ietf:rfc:3897 (URI) / urn:ietf:rfc:3897 (IRI) regex for the scheme: part of a URI/IRI in Python: github.com/dgerber/rfc3987/blob/master/rfc3987.py#L147) – Wes Turner Jun 13 '18 at 15:45

Here is the solution I'm using:

1

```
const result = `${ window.location.protocol }//${ window.location.host }`;
```

EDIT:

To add cross-browser compatibility, use the following:

```
const result = `${ window.location.protocol }//${ window.location.hostname +
(window.location.port ? ':' + window.location.port: '') }`;
```

edited Jul 8 at 20:22

answered Apr 28 at 19:55

JulienRioux
355 4 13

1 Upvoted, but `window.location.host` may not be the best cross-browser – nathanfranke Jul 4 at 7:54

1 Thanks, I've added cross-browser compatibility to my original answer. – JulienRioux Jul 8 at 20:22

0

ES6 style with configurable parameters.

```
/**
 * Get the current URL from `window` context object.
 * Will return the fully qualified URL if neccessary:
 *   getCurrentBaseURL(true, false) // `http://localhost/` - `https://localhost:3000/`
 *   getCurrentBaseURL(true, true) // `http://www.example.com` -
`https://www.example.com:8080`
 *   getCurrentBaseURL(false, true) // `www.example.com` - `localhost:3000`
 *
 * @param {boolean} [includeProtocol=true]
 * @param {boolean} [removeTrailingSlash=false]
 * @returns {string} The current base URL.
 */
export const getCurrentBaseURL = (includeProtocol = true, removeTrailingSlash = false)
=> {
  if (!window || !window.location || !window.location.hostname ||
!window.location.protocol) {
    console.error(
      `The getCurrentBaseURL function must be called from a context in which window
object exists. Yet, window is ${window}`,
      [window, window.location, window.location.hostname, window.location.protocol],
```

```
    )
    throw new TypeError('Whole or part of window is not defined.')
  }

  const URL = `${includeProtocol ? `${window.location.protocol}//` :
''}${window.location.hostname}${
    window.location.port ? `:${window.location.port}` : ''
  }${removeTrailingSlash ? '' : '/'}`

  // console.log(`The URL is ${URL}`)

  return URL
}
```

answered Aug 20 '18 at 9:10

Sébastien
711 5 15

0

window.location.protocol + '//' + window.location.host

answered Sep 9 '18 at 13:22

Code_Worm
1,066 1 17 20

0

Simple answer that works for all browsers:

```
let origin;

if (!window.location.origin) {
  origin = window.location.protocol + "//" + window.location.hostname +
     (window.location.port ? ':' + window.location.port: '');
}

origin = window.location.origin;
```

answered Apr 11 at 2:58

Mike Hawes
227 2 5