

Check if image exists on server using JavaScript?

Asked 5 years, 10 months ago Active 2 months ago Viewed 143k times



Using javascript is there a way to tell if a resource is available on the server? For instance I have images 1.jpg - 5.jpg loaded into the html page. I'd like to call a JavaScript function every minute or so that would roughly do the following scratch code...

104



```
if "../imgs/6.jpg" exists:
    var nImg = document.createElement("img6");
    nImg.src = "../imgs/6.jpg";
```



42

Thoughts? Thanks!

javascript

edited Sep 16 '13 at 21:42



ajtrichards

21.3k 11 75 85

asked Sep 16 '13 at 21:36



0xhughes

1,131 4 18 35

10 Answers



You could use something like:

177



```
function imageExists(image_url){
    var http = new XMLHttpRequest();

    http.open('HEAD', image_url, false);
    http.send();

    return http.status != 404;
}
```

Obviously you could use jQuery/similar to perform your HTTP request.

```
$.get(image_url)
  .done(function() {
    // Do something now you know the image exists.

  }).fail(function() {
    // Image doesn't exist - do something else.

  })
```

answered Sep 16 '13 at 21:38



ajtrichards

21.3k 11 75 85

-
- 3 But shouldn't function name be fileExists because this works on .js .css .html or any other publicly available file. – [CoR](#) Jun 11 '15 at 14:56
-
- 1 Function is awesome. I put it in my collection :) I thought `fileExists` would be better name because this function does not check if image exists on server. It check if file is accessible from server. There is no check if that file actually is an image. It could be .pdf, .html, some random file renamed to *.jpg or *.png. If something ends with .jpg it doesn't mean it's 100% image :) – [CoR](#) Jun 16 '15 at 9:53
-
- 6 This will fail unless accessing the resource is permitted under CORS rules. Using an `Image` object doesn't suffer that limitation. The only advantage of this is that it won't actually download the image. – [Alnitak](#) Jul 13 '16 at 18:28 ✎
-
- 5 cross-domain issue. – [Amit Kumar](#) Aug 25 '16 at 10:41
-
- 1 This does work, but keep in mind that it takes longer to process the request and isn't the best solution. Also it doesn't work on cross origin (definitely in chrome) so you can't use it on `file:///` protocol which means no local usage. – [Cameron Samuels](#) May 20 '17 at 22:40
-

▲ You can use the basic way image preloaders work to test if an image exists.

91

```
function checkImage(imageSrc, good, bad) {
  var img = new Image();
  img.onload = good;
  img.onerror = bad;
  img.src = imageSrc;
}

checkImage("foo.gif", function(){ alert("good"); }, function(){ alert("bad"); } );
```

[JSFiddle](#)

edited May 10 '15 at 22:49



Mikael Engver

3,219 3 33 45

answered Sep 16 '13 at 21:43



epascarello

158k 15 143 190

Hey, cool function! As a side note, this is an interesting way of returning the results, direct into an anonymous function. I would normally do this with a `return` statement like @ajtrichards has in the first part of his answer. – [Sablefoste](#) Jan 18 '17 at 20:44

3 @Sablefoste but synchronous requests are a bad idea... – [epascarello](#) Jan 18 '17 at 21:01

Absolutely; but I never really thought about the other way being synchronous. I am coming from a long history of procedural coding, and sometimes miss the "other" way of looking at things... I'll bet I'm not the only one. ;-) – [Sablefoste](#) Jan 18 '17 at 22:03

1 For future googlers having a problem this solution, try moving the `img.src` definition to immediately after the new `Image` line. – [Gavin](#) Jul 26 '17 at 8:36

4 @Gavin that would not matter at all.... – [epascarello](#) Jul 26 '17 at 13:03

You can just check if the image loads or not by using the built in events that is provided for all images.

44

The `onload` and `onerror` events will tell you if the image loaded successfully or if an error occurred :

```
var image = new Image();

image.onload = function() {
    // image exists and is loaded
    document.body.appendChild(image);
}
image.onerror = function() {
    // image did not load

    var err = new Image();
    err.src = '/error.png';

    document.body.appendChild(err);
}

image.src = "../imgs/6.jpg";
```

edited Feb 2 '18 at 11:25

answered Sep 16 '13 at 21:42



adeneo

270k 21 301 326

2 The best answer for me, as it works in every case (connexion problem, external server...) +1 – [Reign.85](#) Sep 30 '14 at 17:18

1 Compared the performance of this answer with the AJAX.get alternative. This answer performs much faster! – [Samuel](#) Feb 16 '16 at 19:38

I like this solution, anyhow it introduces events in both cases (asynchronous execution), which can cause problems in some situations: I would suggest to append the image in any case and then substitute it only in the case of errors. – [Giorgio Tempesta](#) Jan 31 '18 at 11:01

@adeno, Does it work when Status Code: 404 Not Found – [Mahi](#) Mar 11 at 8:03

@Mahi - The status code shouldn't matter, as long as the 404 page doesn't actually return a valid image, it will fail and trigger the error handler. – [adeneo](#) Mar 21 at 22:39

If anyone comes to this page looking to do this in a **React**-based client, you can do something like the below, which was an answer original provided by Sophia Alpert of the React team [here](#)

13

```
getInitialState: function(event) {
  return {image: "http://example.com/primary_image.jpg"};
},
handleError: function(event) {
  this.setState({image: "http://example.com/failover_image.jpg"});
},
render: function() {
  return (
    <img onError={this.handleError} src={src} />;
  );
}
```

edited Aug 6 '18 at 20:33



[j_quelly](#)

553 1 6 23

answered Jul 13 '16 at 18:25



[Jordan Bonitatis](#)

1,194 10 10

If you create an image tag and add it to the DOM, either its onload or onerror event should fire. If onerror fires, the image doesn't exist on the server.

6

answered Sep 16 '13 at 21:39



[Douglas](#)

25.5k 6 62 86

▲ This works fine:

4 ▼

```
function checkImage(imageSrc) {  
    var img = new Image();  
    try {  
        img.src = imageSrc;  
        return true;  
    } catch(err) {  
        return false;  
    }  
}
```

answered Apr 18 '18 at 11:56



6 Pretty sure this returns true every time? – [Brandon McAlees](#) Jun 20 '18 at 14:52

How is that supposed to work? – [Amin](#) Jul 9 '18 at 11:47

▲ You may call this JS function to check if file exists on the Server:

3 ▼

```
function doesFileExist(urlToFile)  
{  
    var xhr = new XMLHttpRequest();  
    xhr.open('HEAD', urlToFile, false);  
    xhr.send();  
  
    if (xhr.status == "404") {  
        console.log("File doesn't exist");  
        return false;  
    } else {  
        console.log("File exists");  
        return true;  
    }  
}
```

answered Nov 16 '16 at 5:11

[Ani Menon](#)



Good effort. but little bit time consuming when load multiple images at single form. – [Nuwan Withanage](#) Nov 14 '18 at 9:00



2



You can do this with your axios by setting relative path to the corresponding images folder. I have done this for getting a json file. You can try the same method for an image file, you may refer these examples

If you have already set an axios instance with baseurl as a server in different domain, you will have to use the full path of the static file server where you deploy the web application.

```
axios.get('http://localhost:3000/assets/samplepic.png').then((response) => {  
  console.log(response)  
}).catch((error) => {  
  console.log(error)  
})
```

If the image is found the response will be 200 and if not, it will be 404.

Also, if the image file is present in assets folder inside src, you can do a require, get the path and do the above call with that path.

```
var SampleImagePath = require('./assets/samplepic.png');  
axios.get(SampleImagePath).then(...)
```

answered May 30 '18 at 9:59



[Rohith Murali](#)

3,757 2 11 18



2



A better and modern approach is to use ES6 [Fetch API](#) to check if an image exists or not:

```
fetch('https://via.placeholder.com/150', { method: 'HEAD' })  
  .then(res => {  
    if (res.ok) {  
      console.log('Image exists.');    } else {  
      console.log('Image does not exist.');    }  
  })
```

```
}  
}).catch(err => console.log('Error:', err));
```

Make sure you are either making the same-origin requests or CORS is enabled on the server.

answered May 18 at 7:47



[attacomisian](#)

127 1 9

Basically a *promisified* version of @espascarello and @adeneo answers, with a fallback parameter:

1

```
const getImageOrFallback = (path, fallback) => {  
  return new Promise(resolve => {  
    const img = new Image();  
    img.src = path;  
    img.onload = () => resolve(path);  
    img.onerror = () => resolve(fallback);  
  });  
};  
  
// Usage:  
  
const link = getImageOrFallback(  
  'https://www.fillmurray.com/640/360',  
  'https://via.placeholder.com/150'  
)  
.then(result => console.log(result) || result)  
  
// It can be also implemented using the async / await API.
```

Run code snippet

[Expand snippet](#)

Note: I may personally like the `fetch` solution more, but it has a drawback – if your server is configured in a specific way, it can return 200 / 304, even if your file doesn't exist. This, on the other hand, will do the job.


edited May 18 at 8:35

answered Dec 4 '18 at 8:19



[HynekS](#)

562 1 6 18

- 1 For `fetch`, you can use the `ok` property of the `response` object to check whether the request was successful or not: `fetch(url).then(res => {if(res.ok){ /*exist*/} else {/*not exist*/}});` – [attacomsian](#) May 18 at 7:51 
-

That unfortunately does not work for me if I want to check valid background image. `background-image: url('/a/broken/url')` gives me `200` and `ok` (Chrome 74). – [HynekS](#) May 20 at 10:22 