

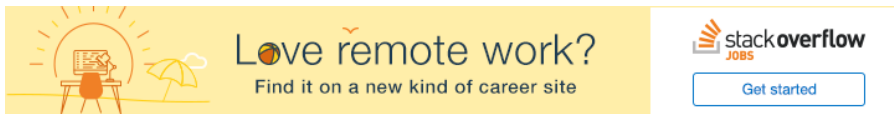
Announcing Stack Overflow Documentation

We started with Q&A. Technical documentation is next, and we need your help.

Whether you're a beginner or an experienced developer, you *can* contribute.

[Sign up and start helping →](#)[Learn more about Documentation →](#)

How to remove an HTML element using Javascript?



I am a total newbie. Can somebody tell me how to remove an HTML element using the original Javascript not jQuery.

index.html

```
<html>
<head>
<script type="text/javascript" src="myscripts.js" > </script>
<style>
#dummy {
  min-width: 200px;
  min-height: 200px;
  max-width: 200px;
  max-height: 200px;
  background-color: #fff000;
}
</style>
</head>
<body>
<div id="dummy"></div>

<form>
<input type="submit" value="Remove DUMMY" onclick="removeDummy();" />
</form>
</body>
```

myscripts.js

```
function removeDummy() {
  var elem = document.getElementById('dummy');
  elem.parentNode.removeChild(elem);
}
```


What happens when I click the submit button, is that it will disappear for a very very short time and then appear back immediately. I want to completely remove the element when I click the button.

[javascript](#) [html](#)

edited Nov 16 '15 at 14:24

 [Flip](#)
822 6 23

asked May 9 '11 at 6:07

 [Newbie Coder](#)
2,300 14 29 46

What if I click it, spot an error and press *stop* ? – [alex](#) May 9 '11 at 6:09

9 Answers

What's happening is that the form is getting submitted, and so the page is being refreshed (with its original content). You're handling the `click` event on a submit button.

If you want to remove the element and *not* submit the form, handle the `submit` event on the form instead, and return `false` from your handler:

HTML:

```
<form onsubmit="return removeDummy();" >
  <input type="submit" value="Remove DUMMY"/>
</form>
```

JavaScript:

```
function removeDummy() {
  var elem = document.getElementById('dummy');
  elem.parentNode.removeChild(elem);
  return false;
}
```

But you don't need (or want) a form for that at all, not if its sole purpose is to remove the dummy div. Instead:

HTML:

```
<input type="button" value="Remove DUMMY" onclick="removeDummy()" />
```

JavaScript:

```
function removeDummy() {
  var elem = document.getElementById('dummy');
  elem.parentNode.removeChild(elem);
  return false;
}
```

However, that style of setting up event handlers is old-fashioned. You seem to have good instincts in that your JavaScript code is in its own file and such. The next step is to take it further and avoid using `onXYZ` attributes for hooking up event handlers. Instead, in your JavaScript, you can hook them up with the newer (circa year 2000) way instead:

HTML:

```
<input id='btnRemoveDummy' type="button" value="Remove DUMMY"/>
```

JavaScript:

```
function removeDummy() {
  var elem = document.getElementById('dummy');
  elem.parentNode.removeChild(elem);
  return false;
}
function pageInit() {
  // Hook up the "remove dummy" button
  var btn = document.getElementById('btnRemoveDummy');
  if (btn.addEventListener) {
    // DOM2 standard
    btn.addEventListener('click', removeDummy, false);
  }
  else if (btn.attachEvent) {
    // IE (IE9 finally supports the above, though)
    btn.attachEvent('onclick', removeDummy);
  }
  else {
    // Really old or non-standard browser, try DOM0
    btn.onclick = removeDummy;
  }
}
```

...then call `pageInit()`; from a `script` tag at the very end of your page `body` (just before the closing `</body>` tag), or from within the `window load` event, though that happens *very late* in the page load cycle and so usually isn't good for hooking up event handlers (it happens *after* all images have finally loaded, for instance).

Note that I've had to put in some handling to deal with browser differences. You'll probably want a function for hooking up events so you don't have to repeat that logic every time. Or consider using a library like [jQuery](#), [Prototype](#), [YUI](#), [Closure](#), or [any of several others](#) to smooth over those browser differences for you. It's **very important** to understand the underlying stuff going on, both in terms of JavaScript fundamentals and DOM fundamentals, but libraries deal with a lot of inconsistencies, and also provide a lot of handy utilities — like a means of hooking up event handlers that deals with browser differences. Most of them also provide a way to set up a function (like `pageInit`) to run as soon as the DOM is ready to be manipulated, long before `window load` fires.

edited May 9 '11 at 6:21

answered May 9 '11 at 6:09



[T.J. Crowder](#)

476k 78 753 889

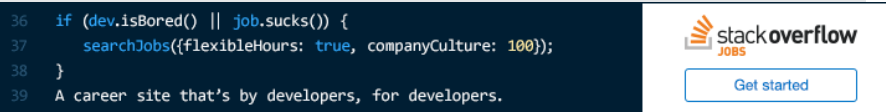
Change it to `type="button"` or put `return false;` in your JS function – Pav May 9 '11 at 6:13

@Pav: I think I'll stick to submit because this is just an experimental exercise. I will be doing some projects that uses submit so I better get used to it... Thanks anyway for the idea. – Newbie Coder May 9 '11 at 6:18

@Newbie: I added a few further notes. – T.J. Crowder May 9 '11 at 6:21

why not just do `elem.remove();` – Muhammad Umer Jul 10 '13 at 0:01

what does `return false` do – user1899563 Sep 13 '13 at 9:29



You should use `input type="button"` instead of `input type="submit"`.

```
<form>
  <input type="button" value="Remove DUMMY" onclick="removeDummy();" />
</form>
```

Checkout [Mozilla Developer Center](#) for basic html and javascript resources

edited May 9 '11 at 6:13



Alastair Pitts

14k 7 52 84

answered May 9 '11 at 6:11



CoderHawk

3,089 1 26 48

- 1 I changed your link from W3Schools to the Mozilla Developer Center. Have a look at [w3fools.com](#) for my reasoning... Apart from that, good answer :) – Alastair Pitts May 9 '11 at 6:14
- 2 @Alastair Pitts - its ok. I just want to point PO to html and js basic tutorials.. – CoderHawk May 9 '11 at 6:17

Your JavaScript is correct. Your button has `type="submit"` which is causing the page to refresh.

edited Sep 17 '13 at 1:53

answered May 9 '11 at 6:10



Pav

1,497 3 12 22

Just do this `element.remove();`

Try it here LOOK

<http://jsfiddle.net/4WGRP/>

answered Jul 10 '13 at 0:11



Muhammad Umer

736 1 7 14

- 3 Careful with this, as this seems to be a fairly new addition, and does not have full browser support namely IE and FF versions 22 and earlier ([red-team-design.com/...](#)). – dule Sep 18 '13 at 18:45
- 1 I'd personally wait another 5 years to use this – slebetman Feb 19 at 2:17

It reappears because your submit button reloads the page. The simplest way to prevent this behavior is to add a `return false` to the `onclick` like so:

```
<input type="submit" value="Remove DUMMY" onclick="removeDummy(); return false;" />
```

answered May 9 '11 at 6:10



mVChr

33k 4 62 80

Change the input type to "button". As T.J. and Pav said, the form is getting submitted. Your Javascript looks correct, and I commend you for trying it out the non-JQuery way :)

answered May 9 '11 at 6:13



Newtang

2,267 4 25 48

I think I'll stick to submit because this is just an experimental exercise for my future projects that uses submit so I better get used to it... – [Newbie Coder](#) May 9 '11 at 6:20

That is the right code. What is probably happening is your form is submitting, and you see the new page (where the element will exist again).

answered May 9 '11 at 6:11



alex

263k 123 645 798

This works. Just remove the button from the "dummy" div if you want to keep the button.

```
function removeDummy() {
    var elem = document.getElementById('dummy');
    elem.parentNode.removeChild(elem);
    return false;
}

#dummy {
    min-width: 200px;
    min-height: 200px;
    max-width: 200px;
    max-height: 200px;
    background-color: #fff000;
}

<div id="dummy">
    <button onclick="removeDummy()">Remove</button>
</div>
```

[Expand snippet](#)

edited Jan 18 at 21:56



Paul Roub

29.1k 8 39 63

answered Jan 18 at 20:26



Durtle02

1

I'm still a newbie too, but here is one simple and easy way: You can use `outerHTML`, which is the whole tag, not just a portion:

EX: `<tag id='me'>blahblahblah</tag>` 's `innerHTML` would be `blahblahblah`, and `outerHTML` would be the whole thing, `<tag id='me'>blahblahblah</tag>`.

So, for the example, if you want to delete the tag, it's basically deleting its data, so if you change the `outerHTML` to an empty string, it's like deleting it.

```
<body>
  <p id="myTag">This is going to get removed...</p>
  <input type="button" onclick="javascript:
    document.getElementById('myTag').outerHTML = ''; //this makes the outerHTML
    (the whole tag, not what is inside it)
    " value="Remove Praragraph">
</body>
```

Instead, if you want to just not display it, you can style it in JS using the `visibility`, `opacity`, and `display` properties.

```
document.getElementById('foo').style.visibility = hidden;
//or
document.getElementById('foo').style.opacity = 0;
//or
document.getElementById('foo').style.display = none;
```

Note that `opacity` makes the element still display, just you can't see it as much. Also, you can select text, copy, paste, and do everything you could normally do, even though it's invisible.

`Visibility` fits your situation more, but it will leave a blank transparent space as big as the element it was applied to.

I would recommend you do `display`, depending on how you make your webpage. `Display` basically deleting the element from your view, but you can still see it in DevTools. Hope this helps!

edited Feb 19 at 1:40

answered Feb 19 at 1:26



[HarryJamesPoter27](#)

36 7
