Best Practices for onload Javascript

Asked 15 years, 11 months ago Modified 15 years, 10 months ago

Viewed 3k times

3

What is the best way to handle several different onload scripts spread across many pages?

For example, I have 50 different pages, and on each page I want to set a different button click handler when the dom is ready.

Is it best to set onclicks like this on each individual page,

```
<a id="link1" href="#" onclick="myFunc()" />
```

Or a very long document ready function in an external js file,

```
Element.observe(window, 'load', function() {
  if ($('link1')) {
    // set click handler
  }
  if ($('link2')) {
   // set click hanlder
  }
  ...
}
```

Or split each `if ($('link')) {}` section into script tags and place them on appropriate pages,

Or lastly, split each `if ($('link')) {}` section into its own separate js file and load appropriately per page?

Solution 1 seems like the least elegant and is relatively obtrusive, solution 2 will lead to a very lengthy load function, solution 3 is less obtrusive then 1 but still not great, and solution 4 will load require the user to download a separate js file per page he visits.

Are any of these best (or worst) or is there a solution 5 I'm not thinking of?

Edit: I am asking about the design pattern, not which onload function is the proper one to use.

javascript **onload**

Share

Improve this question

Follow

edited Jan 26, 2009 at 21:40

asked Jan 26, 2009 at 21:17

erik

6,436 ● 3 ● 37 ● 36

7 Answers

Sorted by: Highest score (default)

Have you thought about making a class for each type of behavior you'd like to attach to an element? That way you

3

could reuse functionality between pages, just in case there was overlap.

For example, let's say that on some of the pages you want to a have button that pops up some extra information on the page. Your html could look like this:

```
<a href="#" class="more-info">More info</a>
```

And your JavaScript could look like this:

```
jQuery(".more-info").click(function() { ... });
```

If you stuck to some kind of convention, you could also add multiple classes to a link and have it do a few different things if you needed (since jQuery will let you stack event handlers on an element).

Basically, you're focusing on the behaviors for each type of element you're attaching JavaScript to, rather than picking out specific ids of elements to attach functionality to.

I'd also suggest putting all of the JavaScript into one common file or a limited number of common files. The main reason being that, after the first page load, the JavaScript would be cached and won't need to load on each page. Another reason is that it would encourage you do develop common behaviors for buttons that are available throughout the site.

In any case, I would discourage attaching the onlick directly in the html (option #1). It's obtrusive and limits the flexibility you have with your JavaScript.

Edit: I didn't realize Diodeus had posted a very similar answer (which I agree with).

Share Improve this answer

Follow

answered Jan 26, 2009 at 21:40

Matt Ephraim

1,360 ● 1 ● 11 ● 15

1

First of all I dont understand why you think setting event listeners is obtrusive?

but ...

Solution one is a bad idea

```
<a id="link1" href="#" onclick="myFunc()" />
```

because you should keep your make-up and your scripts seperate.

Solution two is a bad idea

```
Element.observe(window, 'load', function() {
  if ($('link1')) {
    // set click handler
  }
  if ($('link2')) {
   // set click hanlder
  }
```

```
  ...
}
```

because you are using a lot of unneeded javascript for every page.

Solution three is a bad idea for the same reason I said solution one is a bad idea.

Solution 4 is the best idea, yeah its one extra load per page, but if for each page you just split each if ($('link')) {} section, the file size can not be that large? Plus, if you take this code out of the global javascript, then its load time will be reduced.

Share Improve this answer

Follow

answered Jan 26, 2009 at 21:32

ForYourOwnGood

40.3k ● 5 ● 31 ● 27

> Solution one is obtrusive for the reasons you stated. Your mark-up should be separate from your scripts. I didn't (or least didn't intend to) say event handlers are obtrusive. – erik Jan 26, 2009 at 21:38

> O, ok, from my understanding obtrusive javascript is code that gives the user a hard time, like popping up windows, not code that out of place. – ForYourOwnGood Jan 26, 2009 at 21:49

1

You could hack the class name and use it in a creative manner:

```
<a class="loadevent functionA" id="link1" href="#" onc
```

... on another page...

```
<a class="loadevent functionB" id="link1" href="#" onc
```

You could select by class name "**loadevent**" and grab the other class names for that tag, the other class name being the actual function name you want to hook into. This way one handler would be able to do every page and all you have to do is provide the corresponding class names.

Share Improve this answer

Follow

edited Jan 26, 2009 at 21:42

annakata

75.7k ● 18 ● 115 ● 180

answered Jan 26, 2009 at 21:39

Diodeus - James MacFarlane

114k ● 33 ● 163 ● 180

I would use JQuery's *document ready* if possible

1

```
$(document).ready(function() {

    // jQuery goodness here.
});
```

Share Improve this answer

edited Feb 12, 2009 at 9:48

Follow

answered Jan 26, 2009 at 21:21

cgreeno

32.3k ● 7 ● 68 ● 87

0

While Chris is somewhat correct in that you can do this:

```
$(document.ready(function() {
  // A
});
$(document.ready(function() {
  // B
});
$(document.ready(function() {
  // C
});
```

and all functions will be called (in the order they are encountered), it's worth mentioning that the ready() event isn't quite the same to onload(). From the jQuery API docs:

> Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.

You may want the load() event instead:

```
$(document).load(function() {
  // do stuff
});
```

which will wait for images and the like to be loaded.

Share Improve this answer

Follow

answered Jan 26, 2009 at 21:34

cletus

625k ● 169 ● 917 ● 945

0

Without resorting to the kneejerk jQuery, if the page varying JS is relatively light I would include it in an inline header script (binding to the onload event trigger, yes) similar to #4, but I wouldn't do this as a separate JS script and download, I'd be looking to handle this with a server side include - however you want to handle that (me? I'd go with XSLT includes).

That gives you both a high degree of modular separation and keeps the download as light as possible.

Share Improve this answer

Follow

answered Jan 26, 2009 at 21:39

annakata

75.7k ● 18 ● 115 ● 180

> Amen, on the "use jQuery". I'm tired of all the "just use jQuery" answers that fail to explain the process.
> – Diodeus - James MacFarlane Jan 26, 2009 at 21:42

0

Having a lot of different pages, I would allow different handling of events for those pages ...

If, however, differencies were slight, I would try to find a pattern I could hang on to, and probably make a real simple algorithm to tell the pages apart ...

The last thing I would resort to, was to use a (big) library (jquery, mootools or whatever !-) if I wasn't going to use it in any other way ...

Now you're talking of best practices, best practice would always be what your users will experience as the lightest solution, and in that, users should be understood in the widest possible way, including developers and so on who are to maintain that site !o]

Share Improve this answer

Follow

answered Jan 26, 2009 at 22:34

roenving

2,626 ● 15 ● 14