The art and pain of teaching JavaScript

Christian Heilmann | http://wait-till-i.com | http://twitter.com/codepo8

<head> 2008

Both teaching and learning JavaScript is tough.

Probably the main issue with it is how to get people interested in learning it.

JavaScript information has become a commodity and there are thousands of ready-made solutions available.

The range of people that want to work with JavaScript is staggering.

Designer

Java/Ruby/Python/C++
coder

... which is fun cause 5 years ago neither of these groups wanted to even touch JavaScript.

Still each of these groups want JavaScript to do different things...

...and work the same way they are used to working.

And that makes it very hard to teach it.

As someone who wants to get to grips with learning JavaScript you are in the same dilemma.

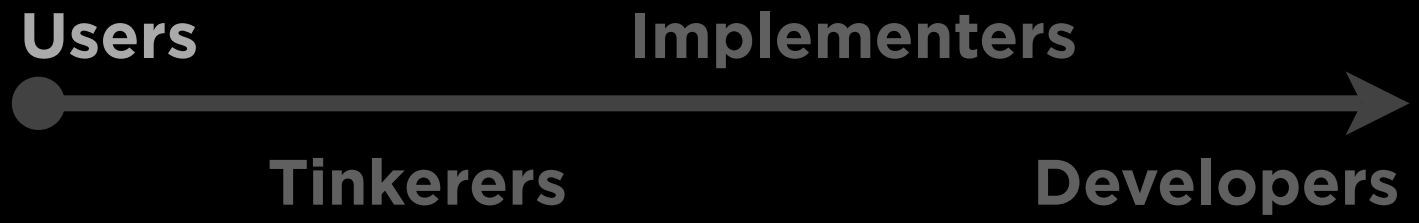
There are hundreds of tutorials on “Learning JavaScript” online.

How could you define which one is good and which isn’t?

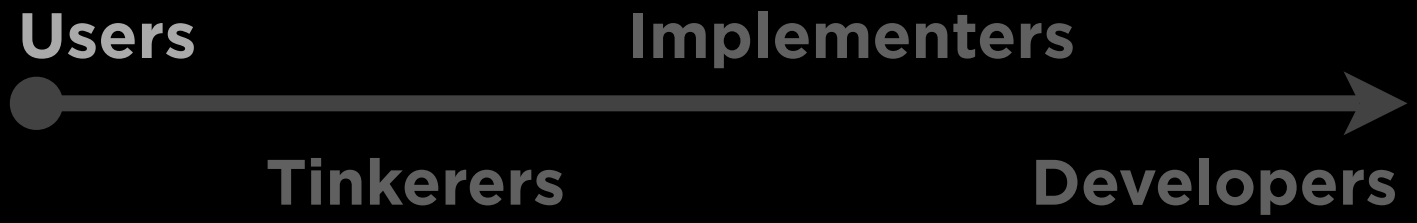
We have to consider several types of people who want to use JavaScript.


Users
Implementers
Tinkerers
Developers

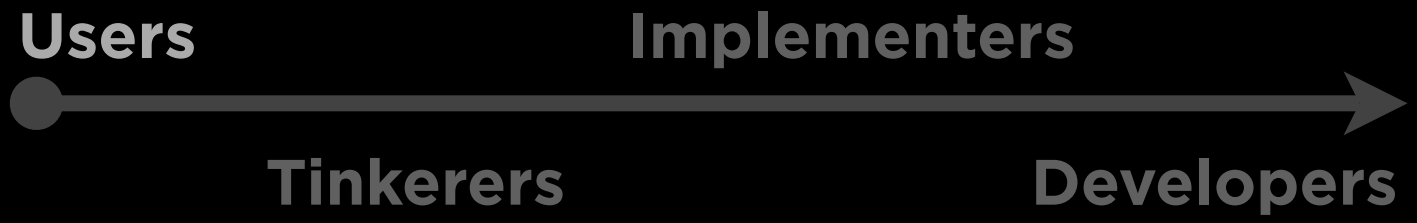
Users simply need a JavaScript solution for a certain task.



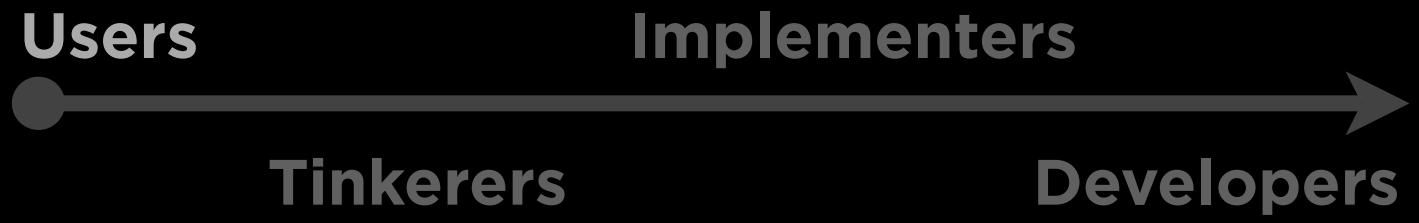
All they are looking for is a copy and paste script that does something.



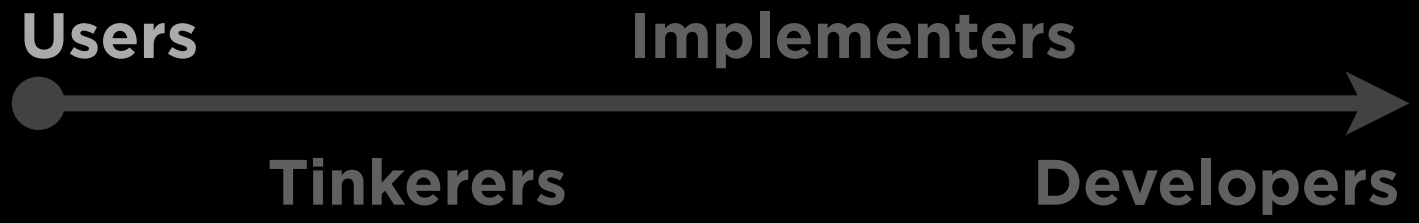
You'll be most successful if you manage to get this implementation in a format they are used to.



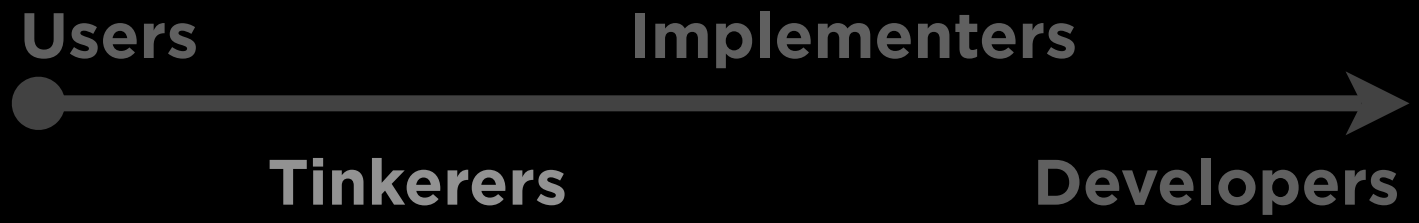
“add this script into a document that also has an element with the ID ‘menu’ to get a sliding menu”



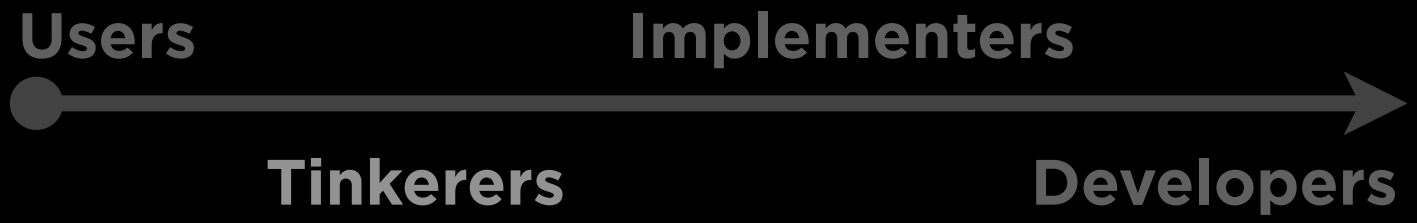
“to add client-side validation of your form simply add the following script and mark mandatory fields with a class called ‘mandatory’.”



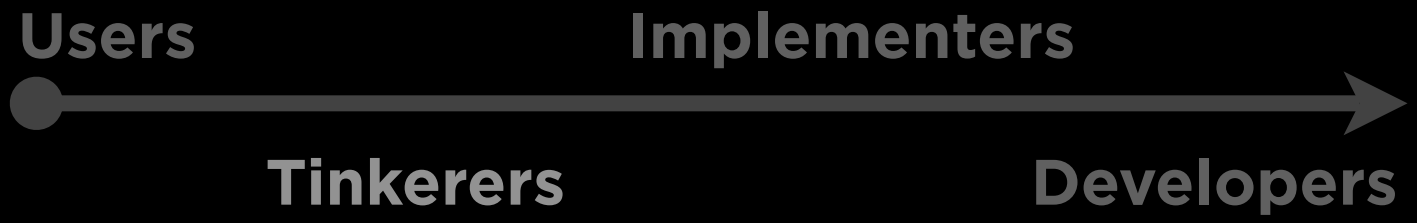
Tinkerers want a JavaScript solution that can be slightly modified.



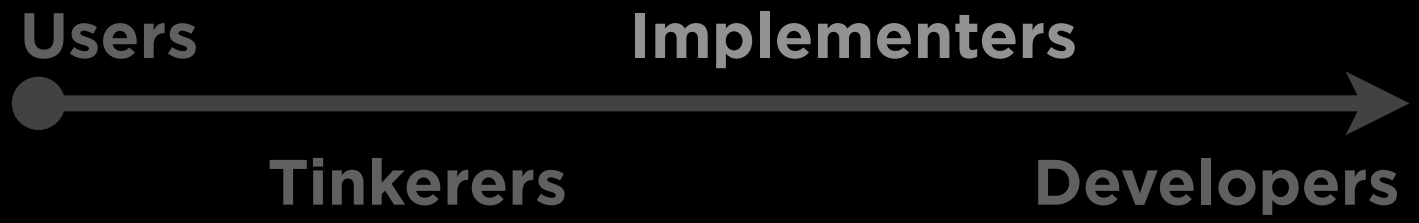
This starts with styling the solution differently and ends at heavier customisation (f.e. label translation)



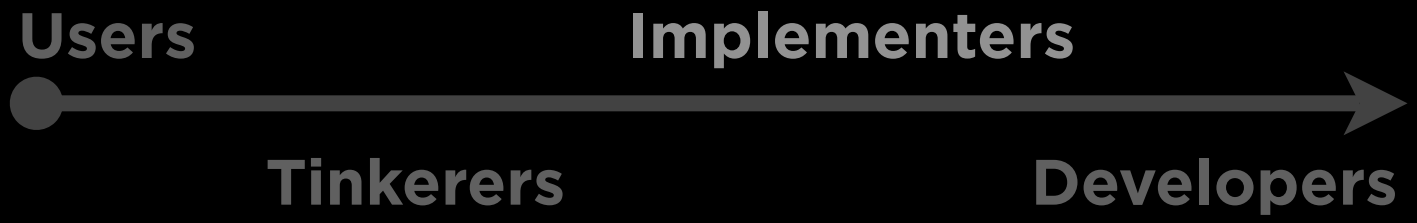
The easier your solution is to customise, the more tinkerers will be happy with it.



Implementers will use your solution and will need much heavier customisation.

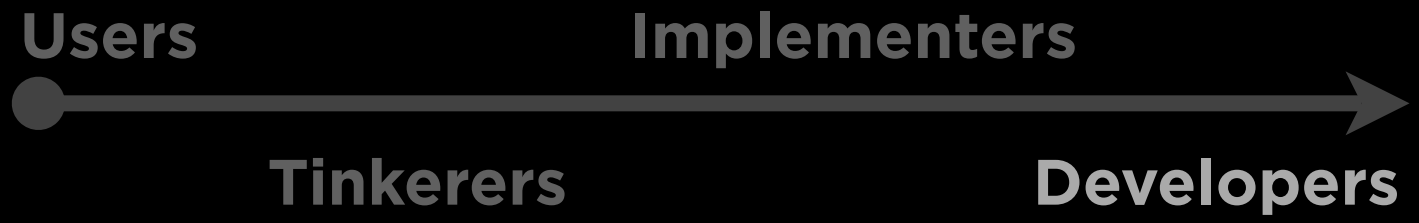


Customisation that will be far beyond what you thought your solution was meant to deliver.

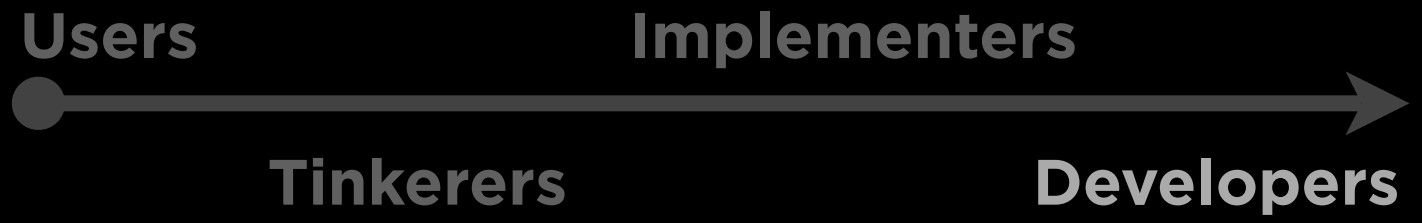


What you'll need to provide there is a way to build upon and extend your solution.

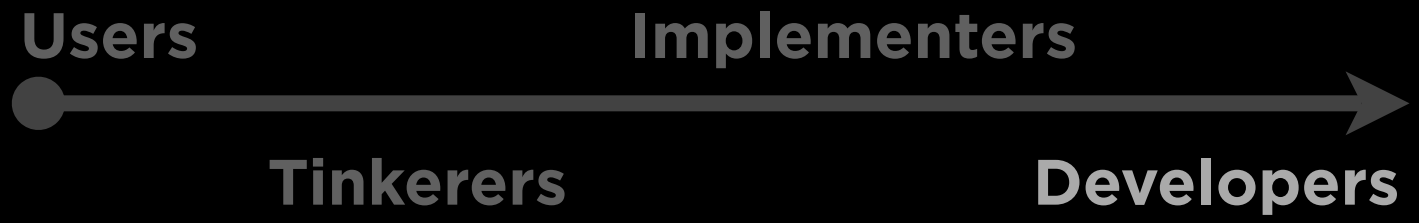
Developers will see your solution as either a base to build upon, inspiration or something to improve.



They are the most vocal, critical and at the same time a very interesting group to reach.



They are people that want to reach under the hood and play with the cables.



Quo Vadis?

http://www.flickr.com/photos/stephmcg/2756340726/

The biggest issue with learning JavaScript is that there are too many old and outdated sources of information on the web.

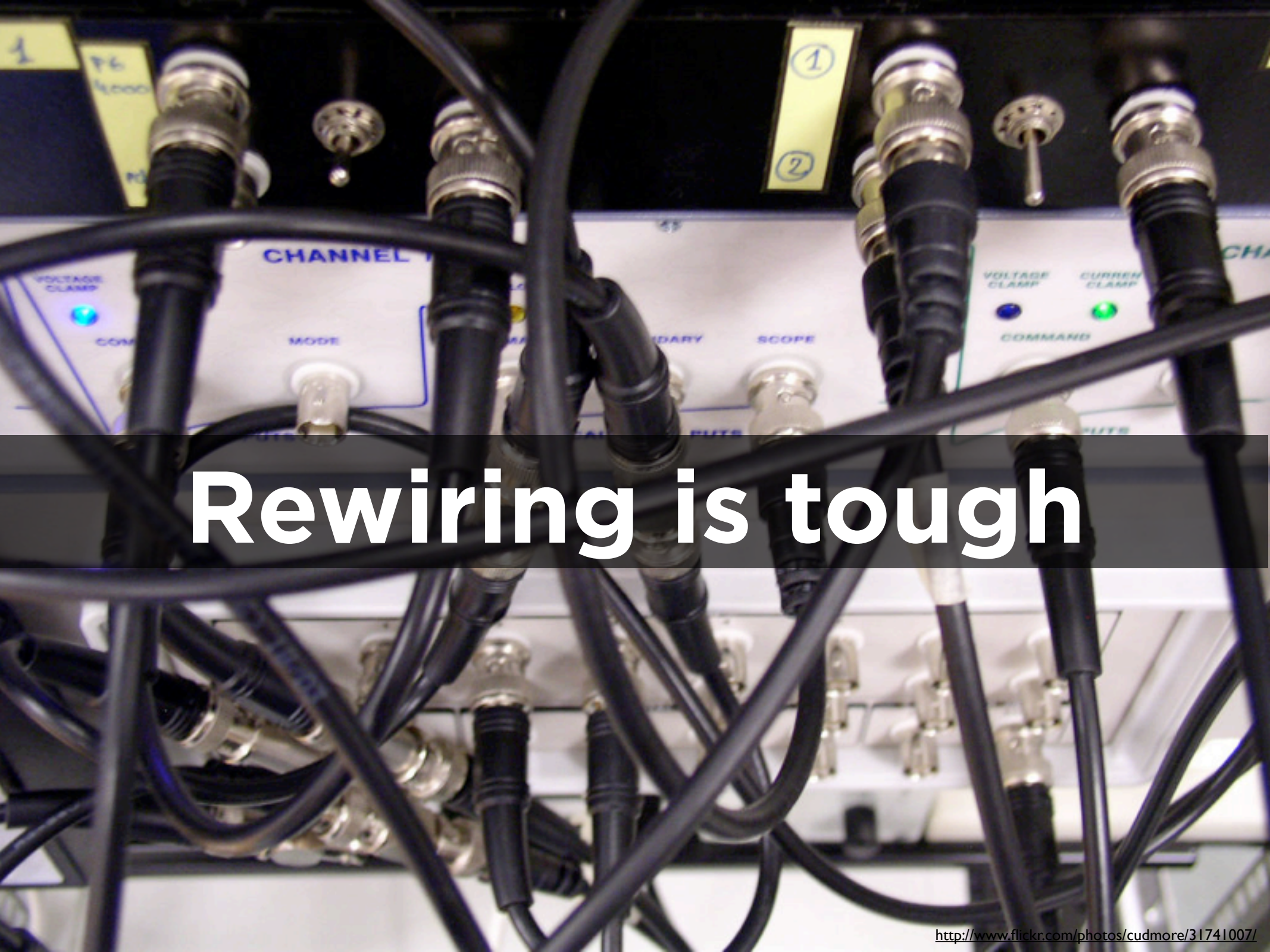
There are a lot of people who provide amazing and up-to-date information.

However, beginners are not likely to ever find this information.

There aren’t any easy ways out of this situation.

Experts are not very inclined to write beginner material.

And the very nature of the web and the workings of search engines favour older material that has been around for a while.



Rewiring is tough

http://www.flickr.com/photos/cudmore/31741007/

Some people* are working on ideas to tackle this problem.

* http://www.thecssdiv.co.uk/2008/09/30/barcamplondon5-slides/

Let’s however now concentrate on building new materials in a way that counteracts attrition.

The first trick is to document what we do more.

“People learn by looking at the source code”

This is how we learnt, yes, but that was because of the lack of good resources.

Looking at source code and copying means we know the how but never the why.

Therefore it might be a good plan to marry code and documentation.

That way code updates mean tutorial updates.

http://icant.co.uk/sandbox/tutorialbuilder/

http://jsdoc.sourceforge.net/

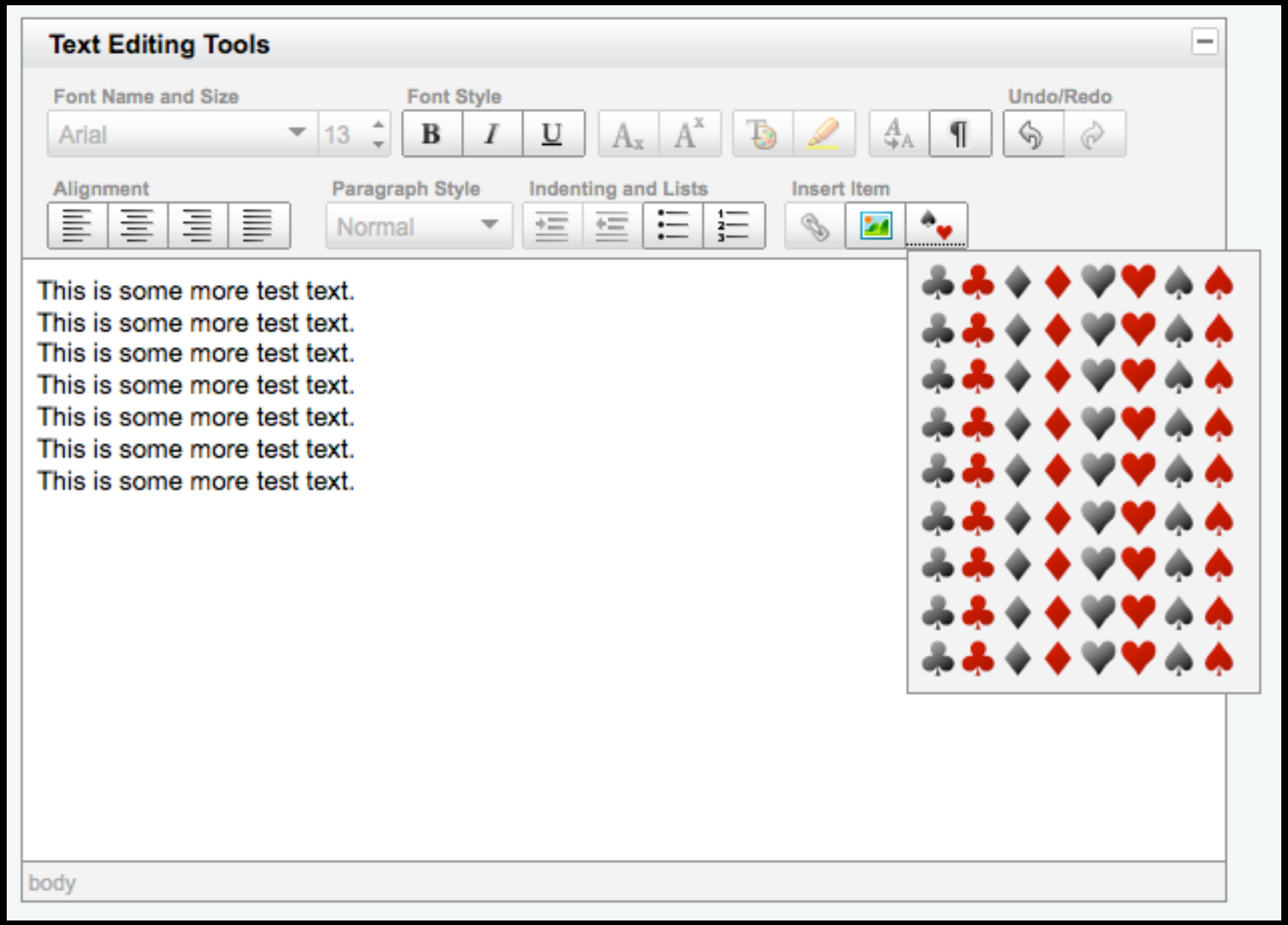
http://www.naturaldocs.org/

In terms of code solutions users of our code should never be forced to go in the source and change it to their needs.

Widgets should be easily skinable.

There should be configuration objects for everything that is likely to change.

In order to support implementers we should consider APIs and plugin systems for our solutions.



http://developer.yahoo.com/yui/editor/

Yes, this will make our solutions to be more work and probably make them a bit more bulky.

But it will ensure that we can control the quality even in the future.

Another idea is to consider an update script for widgets and tutorials that flags outdated local installs.

First and foremost all of this needs a change in how we approach delivering JS information.

It shouldn’t be about the quick win and the applause for being the first.

It should be about releasing products that people can use without needing to know what we know.

It should be about releasing without expecting or forcing the users to alter what they don’t understand.

Basically, it should be about not being elitist...

...and realising that what we had to go through to get where we are now shouldn't be needed any longer.

Instead we should try to liberate people to be creative in new ways that we consider impossible because of our experience.

Or in other words...

OMG! Technology!

Should turn into...

Sharing the joy!

http://www.flickr.com/photos/ptg/2954729291/

Thanks!