THINKFUL PRESENTS

Javascript Best Practices, Part 2

Avoid Heavy Nesting

Code gets unreadable after a certain level of nesting.

A really bad idea is to nest loops inside loops as that also means taking care of several iterator variables (i,j,k,l,m...).

```
function renderProfiles(o){
   var out = document.getElementById('profiles');
   for(var i=0;i<o.members.length;i++){
      var ul = document.createElement('ul');
      var li = document.createElement('li');
      li.appendChild(document.createTextNode(o.members[i].name));
      var nestedul = document.createElement('ul');
      for(var j=0;j<o.members[i].data.length;j++){
         var datali = document.createElement('li');
         datali.appendChild(
            document.createTextNode(
               o.members[i].data[j].label + ' ' +
               o.members[i].data[j].value
            )
         );
         nestedul.appendChild(datali);
      }
      li.appendChild(nestedul);
   }
   out.appendChild(ul);
}
```

You can avoid heavy nesting and loops inside loops with specialized tool methods.

```
function renderProfiles(o){
   var out = document.getElementById('profiles');
   for(var i=0;i<o.members.length;i++){
      var ul = document.createElement('ul');
      var li = document.createElement('li');
      li.appendChild(document.createTextNode(data.members[i].name));
      li.appendChild(addMemberData(o.members[i]));
   }
   out.appendChild(ul);
}
function addMemberData(member){
   var ul = document.createElement('ul');
   for(var i=0;i<member.data.length;i++){
      var li = document.createElement('li');
      li.appendChild(
         document.createTextNode(
            member.data[i].label + ' ' +
            member.data[i].value
         )
      );
   }
   ul.appendChild(li);
   return ul;
}
```

Think of bad editors and small screens.

Optimize Loops

Loops can get terribly slow in JavaScript.

Most of the time it's because you're doing things in them that don't make sense.

Don't make JavaScript read the length of an array at every iteration of a for loop. Store the length value in a different variable.

```
var names = ['George',
'Ringo',
'Paul',
'John'];
for(var i=0;i<names.length;i++){
   doSomethingWith(names[i]);
}
```

```
var names = ['George',
'Ringo',
'Paul',
'John'];
for(var i=0,j=names.length;i<j;i++){
   doSomethingWith(names[i]);
}
```

Keep computation-heavy code outside of loops. This includes regular expressions but first and foremost DOM manipulation.

You can create the DOM nodes in the loop but avoid inserting them to the document.

Keep DOM Access to a Minimum

If you can avoid it, don't access the DOM.

Reason: It's slow and there are all kinds of browser issues with constant access to and changes in the DOM.

Solution: Write or use a helper method that batch-converts a dataset to HTML.

Seed the dataset with as much as you can and then call the method to render all out in one go.

Don't Yield to Browser Whims

Instead of relying on flaky browser behavior and hoping it works across the board...

Avoid hacking around and analyze the problem in detail instead.

Most of the time you'll find the extra functionality you need is because of bad planning of your interface.

Don’t Trust Any Data

Good code does not trust any data that comes in.

- **Don’t believe the HTML document**
  Any user can meddle with it for example in Firebug.
- **Don’t trust that data reaches your function is of the right format.**
  Test with **typeof** and then do something with it.
- **Don’t expect elements in the DOM to be available.**
  Test for them and that they indeed are what you expect them to be before altering them.
- **Never ever use JavaScript to protect something.**
  JavaScript is as easy to crack as it is to code :)

Add Functionality with Javascript Not Content

If you find yourself creating lots and lots of HTML in JavaScript, you might be doing something wrong.

It is not convenient to create using the DOM, it’s flasky to use **innerHTML** (IE’s Operation Aborted error), and it’s hard to keep track of the quality of the HTML you produce.

If you really have a massive interface that should only be available when JavaScript is turned on, load the interface as a static HTML document via Ajax.

That way you keep maintenance in HTML and allow for customization.

Build on the Shoulders of Giants

Javascript is fun, but writing JavaScript for browsers is less so... start with a good library.

JavaScript libraries are specifically built to make browsers behave and your code more predictable by plugging browser holes.

Good libraries help you write code that works without keeping the maintenance overhead of supporting current browsers and those to come.

Development Code is Not Live Code

Live code is written for machines. Development code is written for humans.

- **Collate, minify and optimize your code in a build process.**

- **Don’t optimize prematurely and punish your fellow developers and those who have to take over from them.**
- **If we cut down on the time spent coding we have more time to perfect the conversion to machine code.**

If you're interested in learning more about web development, you should take a look at our Frontend Web Development Course or our AngularJS Course.

Get notified when new guides are released

Enter your email address here

Send Me Your Next Guide »

Created by Thinkful