

Seven Reasons for Code Bloat

WSG London, May 2007

Christian Heilmann

- All of the following is licensed under a Creative Commons Attribution-Share Alike 3.0 License

<http://creativecommons.org/licenses/by-sa/3.0/> which means you can re-use it by attributing anything you use to me.

Go Nuts!



- **First of all: as there has been talk that there is not enough diversity in speakers on web events:**

- I am a foreigner ✓
- I am a pescetarian ✓
- I have flat feet ✓
- I have a sight impairment ✓
- I am ginger ✓

- **There is a new buzzword in town: POSH**
- **Plain Old Semantic HTML**
 - **It takes a bit of effort to make pretty**
 - **It is very good in communicating your content in a clear manner**
 - **It is glamorous to talk about it right now.**



- **But today we are going to talk about BECS instead:**
- **Bloated Embarrassing Code Solutions**
 - **Fast on their feet**
 - **Expensive to make work somewhere else**
 - **Obvious communication problems**



- **One of the biggest problems of web development right now is bloat.**
- **Bloat manifests itself in several forms:**
 - **Slow web sites**
 - **Overloaded servers**
 - **Stretched or missed deadlines**
 - **Maintenance nightmares**



- **I won't talk about server or server side code optimization**
- **I will not talk much about client side code optimization either.**
- **Instead I will talk about the reasons for web site bloat.**

- **We all know how to avoid bloated code.**
- **It has been best practice for years and every Java developer has a lot to say how to optimize your CSS and JavaScript**
- **How come we still have to deal with it?**



- **Reason#1 for bloated code:**
- **Wrong perception of time needed to accustom ourselves to a project**

- **Web developers are gods**
- **We can be allocated to any product, lay our hands on it in a Vulcan mind-meld fashion and understand immediately what is going on.**
- **No need to plan any time for reading up or handover from the previous developer.**



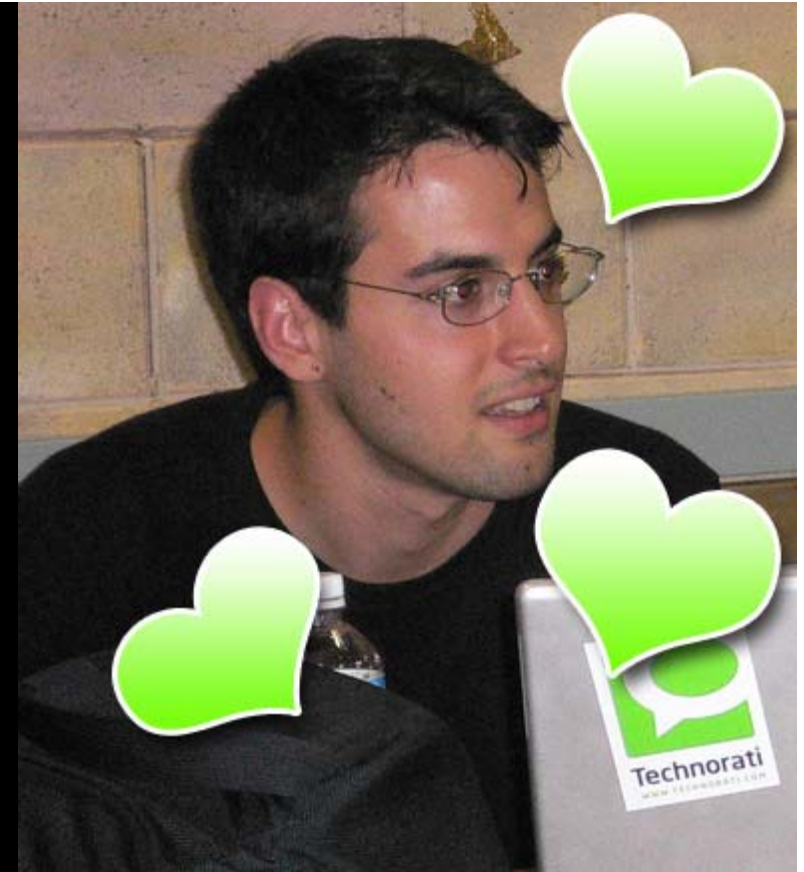
- **Reason#2 for bloated code:**
- **Maintenance without using the right tools**

- **Maintained CSS:**

```
html body #content #mainsection li a:link{
    color:#333;
}
html body #content #mainsection li a:visited{
    color:#000;
}
html body #content #mainsection li a:hover{
    color:#999;
}
html body #content #mainsection li a:active{
    color:#999;
}
```

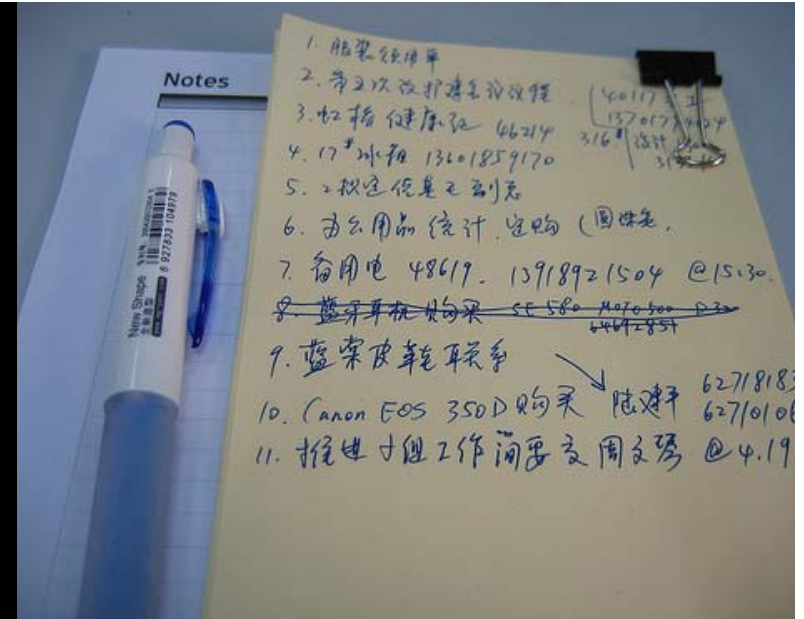
- **Other solutions:**
 - **!important for all**
 - **Adding extra DIVs with IDs.**
 - **Inline styles or style attributes**
- **For JavaScript:**
 - **Add another addEvent**
 - **Add an inline script**
 - **Add some inline event handlers or javascript: links**

- A hero to the rescue



- **Reason#2 for bloated code:**
- **Bad or non-existent documentation**

- **Documentation is a PITA:**
 - You need to know the system to write it
 - You need to be able to communicate the system to others in an easy to understand manner.
 - This means a lot of time and effort.



- **The cheap solution**
 - **Create the documentation from source code comments.**
 - **That way you don't need to spend extra time on it.**
 - **You also ensure that only the people who originally developed the system will be able to understand the documentation.**



- **A real solution**
 - **Let your development team explain the system to someone who can write in a fashion comprehensible to humans.**
 - **By all means have a good JavaDoc style API documentation.**
 - **Don't forget the “cookbook” approach though.**
 - **Don't bother with “hello world” examples, use real implementation scenarios instead.**



- **Reason#3 for bloated code:**
- **People do not read or look before they start**

- **Sounds harsh?**
- **First example: a very easy binary system in daily use.**



- **Or take this example from Amazon.**
- **It is a comment on my book “Beginning JavaScript with DOM Scripting and Ajax”**

I keep running into a custom object in the code examples of the book called "DOMhelp". (...) For example, instead of using the actual DOM methods to get all the links on the page and loop through them, he shows you a line of code that just says "DOMhelp.getlinks". Yes, that line does the same thing by accessing his object and running the regular DOM functions, but what does it teach me? Nothing. That alone is a big enough annoyance to regret buying this book.



- **There is no function with that name anywhere in the book.**
- **Chapter 4 introduces the idea and rationale of JavaScript libraries and takes tool methods explained in Chapter 1 to 3 to assemble DOMhelp.js.**



- **Another example:**
 - **A supplier of one of my clients was to build a small web site explaining functionality with slide shows.**
 - **The slide shows were JS dependent and my client asked me to send them an unobtrusive script.**

- **This is what I did for them:**

```
popups = {  
  // id of the section with the popup links  
  popupsContainerId:'main',  
  // id of the login show link  
  loginLinkId:'loginshowtrigger',  
  // id of the subscribe show link  
  subscribeLinkId:'subscribeshowtrigger',  
  // text of the login slideshow link  
  loginLabel:'how to login',  
  // text of the subscribe slideshow link  
  subscribeLabel:'how to subscribe',  
}
```

- **This is what I did for them:**

```
init:function() {
    var o =
document.getElementById(popups.popupsContainerId);
    if(o) {
        popups.loginLink = document.createElement('a');
        popups.loginLink.setAttribute('href', '#');
        popups.loginLink.appendChild(popups.loginLabel);
        popups.loginLink.id = popups.loginLinkId;
        popups.addEvent(popups.loginLink, 'click',
                        popups.loginShow);
        o.appendChild(popups.loginLink);
    }
}
```

- **This is what I did for them:**

```
popups.subscribeLink = document.createElement('a');
popups.subscribeLink.setAttribute('href', '#');

popups.subscribeLink.appendChild(popups.subscribeLabel);
popups.subscribeLink.id = popups.subscribeLinkId;
popups.addEvent(popups.subscribeLink, 'click',
                popups.subscribeShow);
o.appendChild(popups.subscribeLink);
}
},
loginShow: function() {...},
subscribeShow: function() {...},
addEvent: function() {...}
}
popups.addEvent(window, 'load', popups.init);
```

- **This is part of the HTML that came back:**

```
<a href="javascript:popups.loginShow()">Show login  
demo</a>  
<a href="javascript:popups2.subscribeShow()">Show  
subscribe demo</a>
```

- **This happened to my script:**

```
popups = {
  [...]
  init:function() {
    var o =
document.getElementById(popups.popupsContainerId) ;
    if(o) {
      [...]
      // o.appendChild(popups.loginLink) ;
      [...]
      // o.appendChild(popups.subscribeLink) ;
    }
  },
  loginShow:function() {...},
  subscribeShow:function() {...},
  addEvent:function() {...}
}
```

- **This happened to my script:**

```
popups2 = {
  [...]
  init:function() {
    var o =
document.getElementById(popups.popupsContainerId);
    if(o) {
      [...]
      // o.appendChild(popups.loginLink);
      [...]
      // o.appendChild(popups.subscribeLink);
    }
  },
  loginShow:function(){...},
  subscribeShow:function(){...},
  addEvent:function(){...}
}
popups.addEvent(window,'load',popups.init);
popups2.addEvent(window,'load',popups2.init);
```

- Now, all of this could be explained with a pretty easy equation.

People = Morons

- **However, I think the problem lies deeper.**
- **There is a natural instinct in people to solve issues their way first and then listen to reason or trust in any information.**
- **Take this behaviour example:**

- **Man buys a ridiculously expensive technical gadget that needs assembling**

- **Man looks at wonderfully designed and written manual**

- **Man puts manual aside**

- **Man starts assembling the gadget to the best of his knowledge**

- **If things don't fit – more pressure will do the trick.**

- **Assembled product doesn't work**

- **Man shakes the product or knocks on its side or top**

- **Man goes back to shop or picks up the phone and complain to the manufacturer that their product is too hard to assemble or doesn't work**

- **Clever companies discovered that pattern and style their products accordingly.**
- **Cryptic iconography**
- **Hard to pronounce names**
- **Not enough or too many assembly parts (pot luck)**



Assembly !== Chore



Assembly === Adventure



THE EXPERT'S VOICE® IN WEB DEVELOPMENT



Beginning JavaScript with DOM Scripting and Ajax

From Novice to Professional

The ultimate guide to modern JavaScript development!

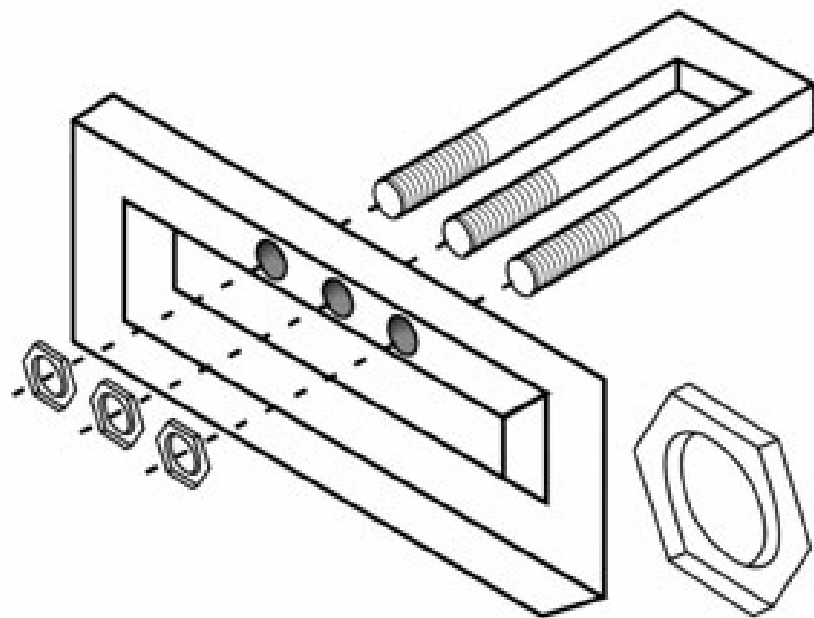
Christian Heilmann

*Foreword by Simon Willison,
Technology Development at Yahoo!*

Apress®

HEDFUK

DØM+Jåvascrip



- **Heilmann's law of documentation:**

- Heilmann's law of documentation:

Your documentation is only as good as the worst recipient.



- **First follow-up fact:**



- **First follow-up fact:**

The worst recipient is the only person that will contact you about your documentation.



- **Second follow-up fact:**



- **Second follow-up fact:**

This is also the only person that will write about your product elsewhere.



- **Reason#5 for bloated code:**
- **Lack of awareness**

- **A lot of times people don't bother understanding the impact of something they use before they use it.**
- **You see web sites that have prototype, scriptaculous (add the dots yourself), jQuery and YUI included as there was a cool demo somewhere they wanted to use.**

- **With YUI you also see the wrong versions in use. Each YUI component comes in three flavours:**
 - **The minified version (no comment, no whitespace)**
 - **The normal version (for easy readability)**
 - **The debug version (with lots of comments being sent to console or logger)**

- **A big problem is that scripting is considered as a given.**
- **It is not. JavaScript can and will be disabled for some users.**
- **This is the test case that should be tried out from time to time.**

- **Make sure that those users don't get overwhelmed with too many options and content on a single page.**
- **If necessary, you can load the extra content with Ajax after the page has loaded.**
- **This will also make the initial loading and rendering a lot faster.**

- **As this easy testing step is not taken we have bloated web sites.**
- **The same phenomenon shows itself when people learnt the syntax of a technology (w3schools.com anyone?) and apply this little knowledge all over the shop.**

- I am sure you have seen constructs like this:

```
<ul>
  <li class="list-item">The Passenger</li>
  <li class="list-item currentlyplaying">Louie Louie</li>
  <li class="list-item">I want to conquer the world</li>
  <li class="list-item">Foxtrott Uniform Charlie Kilo</li>
</ul>
```

```
li.list-item{ padding:.5em; font-
family:courier;color:#000; }
li.currentlyplaying{ color:#c00; }
```

CSS for the logically challenged

- **Start with a global whitespace reset**

```
*{  
  margin:0;  
  padding:0;  
  list-style:none;  
  border:none;  
}
```

- **Now define the most common elements you use and give them a predefined style:**

```
body{
  font-family:helvetica,arial,sans-serif;
  background:#fff;
  color:#333;
  padding:2em;
}
p,li {
  padding-bottom:.5em;
  line-height:1.3em;
}
h1,h2,h3,h4,h5,h6{
  padding-bottom:.5em;
}
```

- Then override or extend these settings with special cases inside elements with IDs:

```
#playlist li{
  padding:1em .5em;
}
```

```
#header p{
  border:1px solid #999;
  background:#ddd;
}
```

```
<ul id="playlist">
  <li>The Passenger</li>
  <li>Louie Louie</li>
  <li>I want to conquer the world</li>
  <li>Foxtrott Uniform Charlie Kilo</li>
</ul>
```

- Then you can detect the exceptions to the rule that need extra formatting. For these use CSS classes.

```
#playlist li{ padding:.5em; font-family:courier;color:#000; }  
#playlist li.currentlyplaying { color:#c00; }
```

```
<ul id="playlist">  
  <li>The Passenger</li>  
  <li class="currentlyplaying">Louie Louie</li>  
  <li>I want to conquer the world</li>  
  <li>Foxtrott Uniform Charlie Kilo</li>  
</ul>
```

- **The best thing about Cascading Style Sheets is that they do cascade.**
- **Instead of re-define, try to re-use and extend.**

- **Reason#6 for bloated code:**
- **Failure to specialize**

- **This is a tricky one.**
- **Code bloat happens when you try to make your code do everything that is even remotely possible instead of allowing it to do one thing right.**
- **Edge cases become more important than covering the basics.**

- **The reason is that you want to meet expectations.**
 - **A script that only does one thing and that thing really well is great to use.**
 - **A script that does one thing, offers fancy options and is as generic as possible to be re-used over and over again is more appreciated.**

- **This becomes most apparent in JavaScript libraries.**
- **To me, a JS library needs to do one thing:**
- **Make browser and language behaviour less random.**
- **If it does that, I am happy as I do know how to write JavaScript.**

- **The most acclaimed libraries do more though:**
 - **Extend the language with a new syntax – chainable methods for example**
 - **Allow access to the document in various ways: CSS selector, Xpath, background color (not yet, but who knows)**

- **The rationale: easier development by avoiding verbose DOM methods.**
- **However – HTML is the thing that is most likely to change at any time.**
- **With DOM methods I can test every step on the way and know when something fails.**

- **The other snag: compatibility of libraries.**
- **Both method names and parameter order vary.**
- **Not a problem, when you stick to one library.**
- **Agencies however are likely to change technology from project to project.**

- **We already had that:**

- **Intershop**
- **Intershop Infinity**
- **Tridion**
- **Immediacy**
- **Vignette**
- **Documentum**
- **Focus Goal Builder**
- **Mambo / Joomla**
- **Typo3**

- **If I want to hire a good web developer right now, I ask for the following:**
 - **HTML / Semantics**
 - **CSS**
 - **JavaScript / DOM**
 - **PHP**
 - **London based or willing to commute**
 - **Always eager to improve.**



Talk to me later!

- **However, if I were still working in an agency, this might be:**

- **JavaScript**

- **jQuery**

- **Dojo**

- **Apollo**

- **Prototype**

- **Mootools**

- **Atlas**

- **Google Web Kit**

- **HTML, CSS**

- **Another type of bloat is bad implementation of scripts.**
- **Maybe some scripts should be more restrictive.**
- **Does a multi level dropdown script or an image slideshow with transition effects really need to be possible several times in the same page?**

- **JavaScript is there to help aiding the users, not to overwhelm them.**
- **If you allow bad implementers to easily create a bloated interface you don't follow that principle.**
- **This may sound arrogant, but I'd rather have my scripts not used than to have them used to create inaccessible web sites.**

- **Instead of trying to create “one size fits all” solutions we could create the following:**
 - **Do one job right**
 - **Be extendable**
 - **Offer extension modules.**

- **Reason#7 for bloated code:**
- **Lack of a front-end build process**

- **Web Developers don't get any Kudos.**
- **There is a code freeze phase with a kill date on backend code.**
- **There are build scripts that convert maintained code into live code.**
- **The frontend however is for everyone to change.**

- **HTML, CSS and JavaScript are not really programming skills.**
- **Therefore anyone can be parachuted in to use Vi (or Emacs) on the live server to add a FONT here and a CENTER there.**

- **It is tricky to get out of this, as it needs a shift in the perception of the job of a web developer.**
- **The IWA/HWG together with the W3C is working on a web developer certification which can make us a larger blip on the radar of business.**
- **However, right now there is a sneakier way:**

- **Minify the heck out of your code before it goes live.**
- **Concatenate all scripts into one include.**
- **Concatenate all CSS into one include**
- **There is a script for this available on**
<http://www.ejeliot.com>

- **Add a comment above the document stating that this was built and is not code to be edited.**
- **Use CSS sprites to use only a few images on the whole page.**
 - **Your pages will load and render faster (less HTTP requests).**
- **For added fun, add scary comments.**

- **Examples of good comments to throw off non-web developers:**

```
<!-- built on 12.03.07 12.33 GMT Checksum E5322AE - OK,  
build continues -->  
<!-- built on 14.03.07 08.00 Checksum error!- build  
failed, sent notification mail -->  
<!-- built on 14.03.07 08.10 Checksum E5322F3 - fix  
initiated and successful -->  
<!-- verified fix mstephens 14.03.07 09.00 -->
```

(mstephens should be someone in upper management)

- **Examples of good comments to throw off non-web developers:**

```
/* Internet Explorer hack to prevent colour bleed and sync loss*/
```

```
/* do not change order or Firefox will go into hasWrongRender mode */
```

```
<!-- Whitespace HTML rendering mode on, do not change! -->
```

```
/* Last change stopped Google indexing this, please don't change anything! */
```

- Last but not least, see you at **Open Hack Day** on **16th of June** in **Alexandra Palace**.
- Sign up at <http://www.hackday.org>

ALEXANDRA PALACE
HACK DAY
LONDON, JUNE 16/17

THANKS!

rel = "me"



chris.heilmann@gmail.com

<http://wait-till-i.com>

<http://icant.co.uk>

