

# WRITING MAINTAINABLE CSS

Natalie Downe | Barcamp London 5

28th September 2008

1

Hello,

my name is Natalie Downe and the original title of my talk was ‘writing maintainable CSS’ ..

# ~~WRITING MAINTAINABLE CSS~~

Natalie Downe | Barcamp London 5

28th September 2008

... but perhaps a better title would be ...



# CSS SYSTEMS

Natalie Downe | Barcamp London 5

28th September 2008

3

... CSS systems (pause)

for reasons which with any luck will become apparent by the end of the talk,

there should be time at the end for any questions so if you could cross your metaphorical legs till then that would be great..

so

hello, my name is Natalie downe and ...



# I am an optimist

... I am an optimist ...

so I am going to  
assume that

... so I am going to assume that ...





you already write beautiful and valid markup

6

... your markup is a work of art

you delight in finding the perfect element for the context

you are a perfectionist ...

<http://www.flickr.com/photos/21984953@N00/38992864/>



you use semantic markup & class / id names

7

... I don't need to persuade you to use semantic markup and class names,

You follow the POSH principles of **Plain old semantic HTML** and ...

<http://www.flickr.com/photos/24839392@N00/2396027766/>



you separate content, presentation & behavior

8

... you embrace the trifle of the web,  
so you separate your content from presentation and behavior,  
you are not using inline styles **or inline** javascript  
and you do all this ...  
<http://www.flickr.com/photos/83096974@N00/466881574/>





# because you care

... because you care ...

# Maintainability in CSS?

10

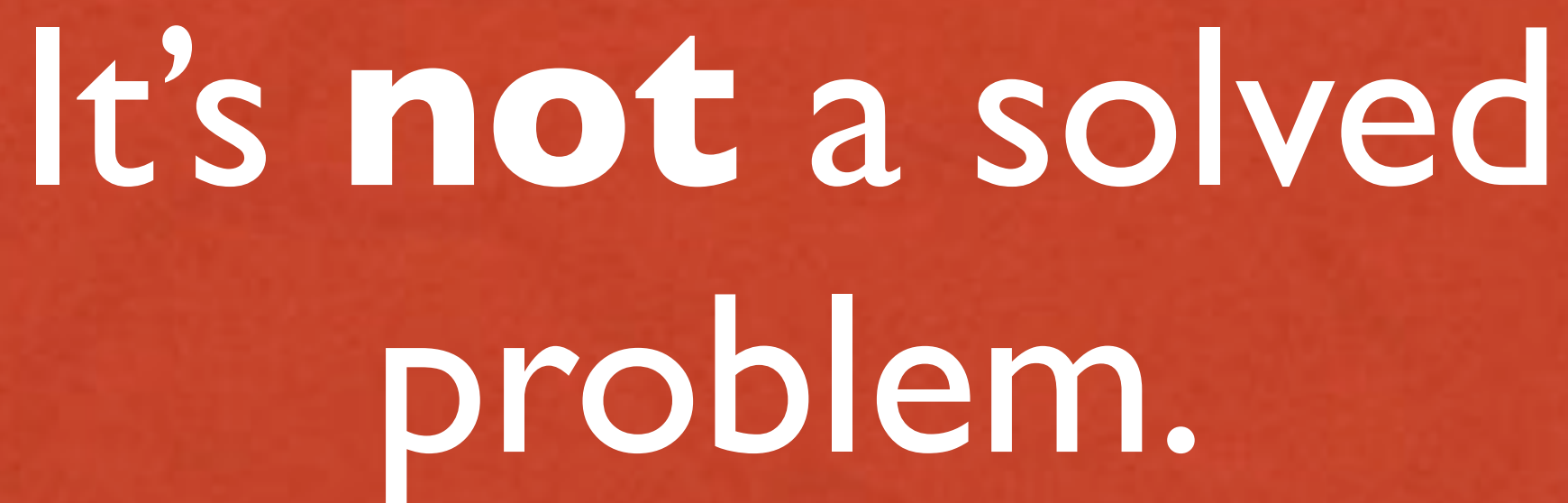
... ok, so put up your hand if you feel you have already solved the problem of maintainability in CSS? ...

yes? so why are you here?

so you write maintainable code all the time?

how well would it **actually** shape up if the person maintaining it only **learned** CSS last week?

...



# It's **not** a solved problem.

11

... no

its not a solved problem

Sorry, currently there is no silver bullet, no miracle cure or revolutionary method

yet –

the thing to bear in mind with maintainability though ....



# It's not just about the future

12

... its not just about the future

its a big part of it of course,

but designing for maintainability isnt just about 'what happens if I look at this in **2 years time**' ...

# It's about now

13

...its about writing good **quality** code **now**

It helps when you come to

- work with other people **or**
- deal with changing requirements throughout the development process

this happens an awful lot in **agile**

maintainability is a **characteristic** of good quality code

...



# CLEARLEFT

- High **quality** code
- To tight **deadlines**
- For **handover** to external developers

14

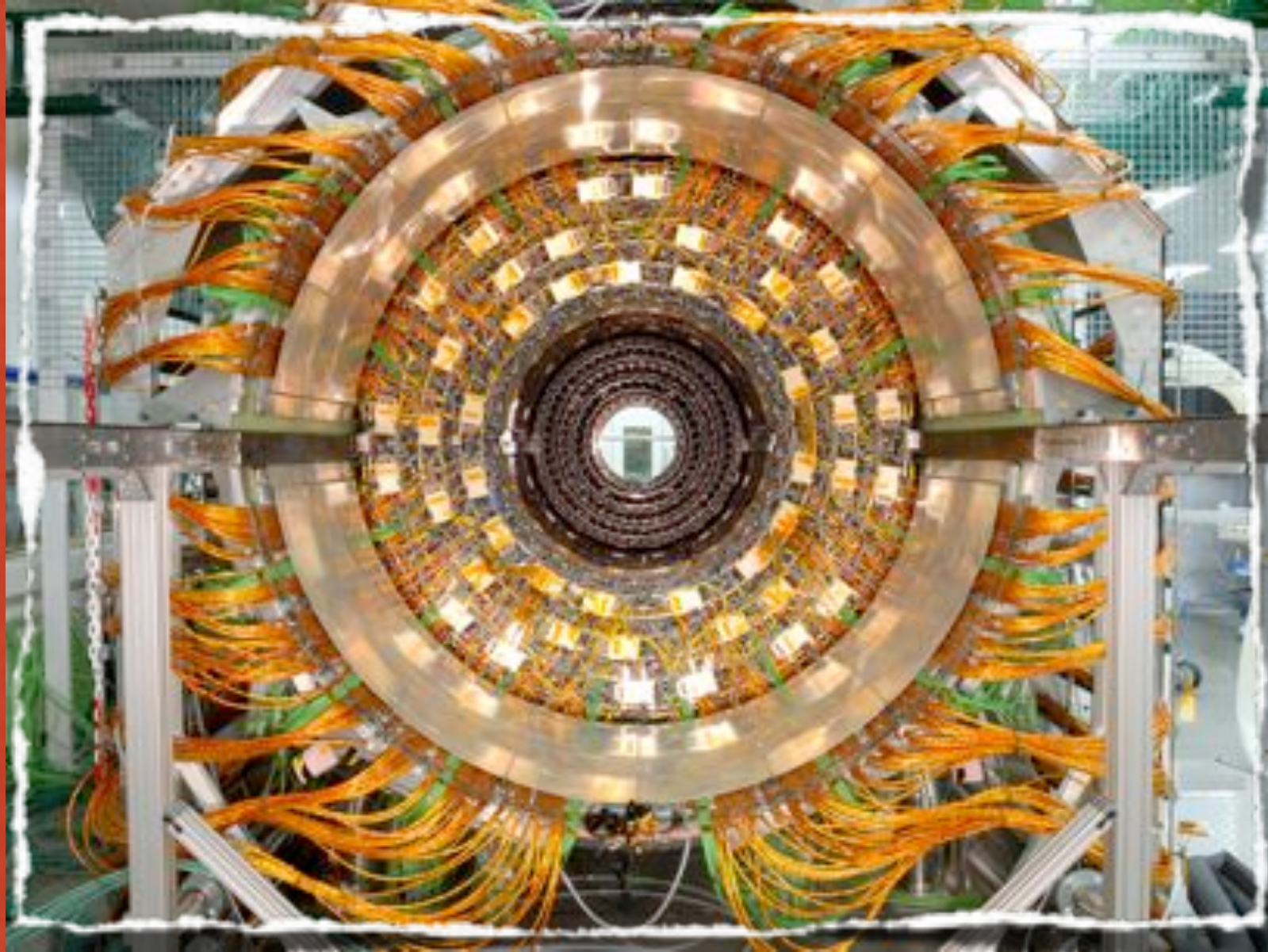
... for those of you who dont know, I work for clearleft,  
a user experience agency in Brighton.

We strive to produce

- high **quality** code,
- to **agency** deadlines, this normally works out as about 2 or 3 weeks,
- this is then handed over for implementation by **external developers** or delivery to the client

to meet these challenges we need a standard method of writing markup and css,  
enter stage left ...





# CSS SYSTEMS

15

... CSS systems

This is a new term that I am introducing today to describe a process that is working very well for us at Clearleft

now don't have a snappy one line description so for now I am going to inflict yet more bullet points on you, sorry ...

[http://www.boston.com/bigpicture/2008/08/the\\_large\\_hadron Collider.html](http://www.boston.com/bigpicture/2008/08/the_large_hadron Collider.html)

Maximilien Brice, © CERN (used for promotion)



# A CSS SYSTEM IS...

- a **top down** approach
- **personalised** for an individual site
- a **reusable set** of markup patterns and CSS
- looking at the **overall structure and individual components**
- glossary of **shared vocabulary** for developers

16

... A CSS system is top down approach for thinking about your css.

it is personalized for an individual site

a unique snowflake

and by that I mean it is a **reusable collection** of markup patterns and accompanying that exists for one site,

they are **unique** and built against that particular site's **design goals**

you stop thinking about individual rules or even about your selectors

instead, think about the **overall structure and components** of the pages within the site

a system also **incorporates a shared vocabulary** for developers to talk about the code,

so for example they know exactly what I mean if I am talking about a teaser or a content block

...

# ORDERING

- Rule blocks are loosely ordered by specificity,
  - **Elements**, grouped by type of tag
  - **Classes**, grouped by the effect they create
  - **IDs**, grouped by the component they affect

17

... whether you are using one CSS file split into several main groups or several files –  
– ordering is very important, order rule blocks loosely by specificity.

(as a house style we use one file, to reduce the number of HTTP hits and because CSSEdit makes it so easy to do)

**elements** are grouped by the type of tag  
for example Lists, links, blockquotes

**classes** are grouped by the affect  
for example Notifications, pagination, content modules

**ID's** are grouped by the individual page component they affect  
for example the Header, toolbar, nav

...



# GROUPING

- **general styles**
  - body styles
  - reset
  - links
  - headings
  - *other elements, tags*
- **helper styles**
  - forms
  - notifications and errors
  - *consistant items with normally just one class*
- **page structure**
  - *skeleton including page furniture,*
- **page components**
  - *most of your styles will be in here*
- **overrides**

18

... your system is made infinitely easier by arranging your rule blocks into a **hierarchy of groups** that don't affect the code but do make it easier to understand

so this is how we order our CSS into **primary** groups at clearleft.

the whole file is roughly ordered by **specificity**,

**general styles** contains the body styles, our reset and default element styles.

for those of you who dont know, a reset is a set of styles that blanket the way browsers display elements by default. My advice however is to **only** use a reset you have written yourself or that you understand

**helper styles** are generally a class or a few classes with elements, eg notification styles, pagination and forms. ---

# GROUPING

- **general styles**
  - body styles
  - reset
  - links
  - headings
  - *other elements, tags*
- **helper styles**
  - forms
  - notifications and errors
  - *consistant items with normally just one class*
- **page structure**
  - *skeleton including page furniture,*
- **page components**
  - *most of your styles will be in here*
- **overrides**

19

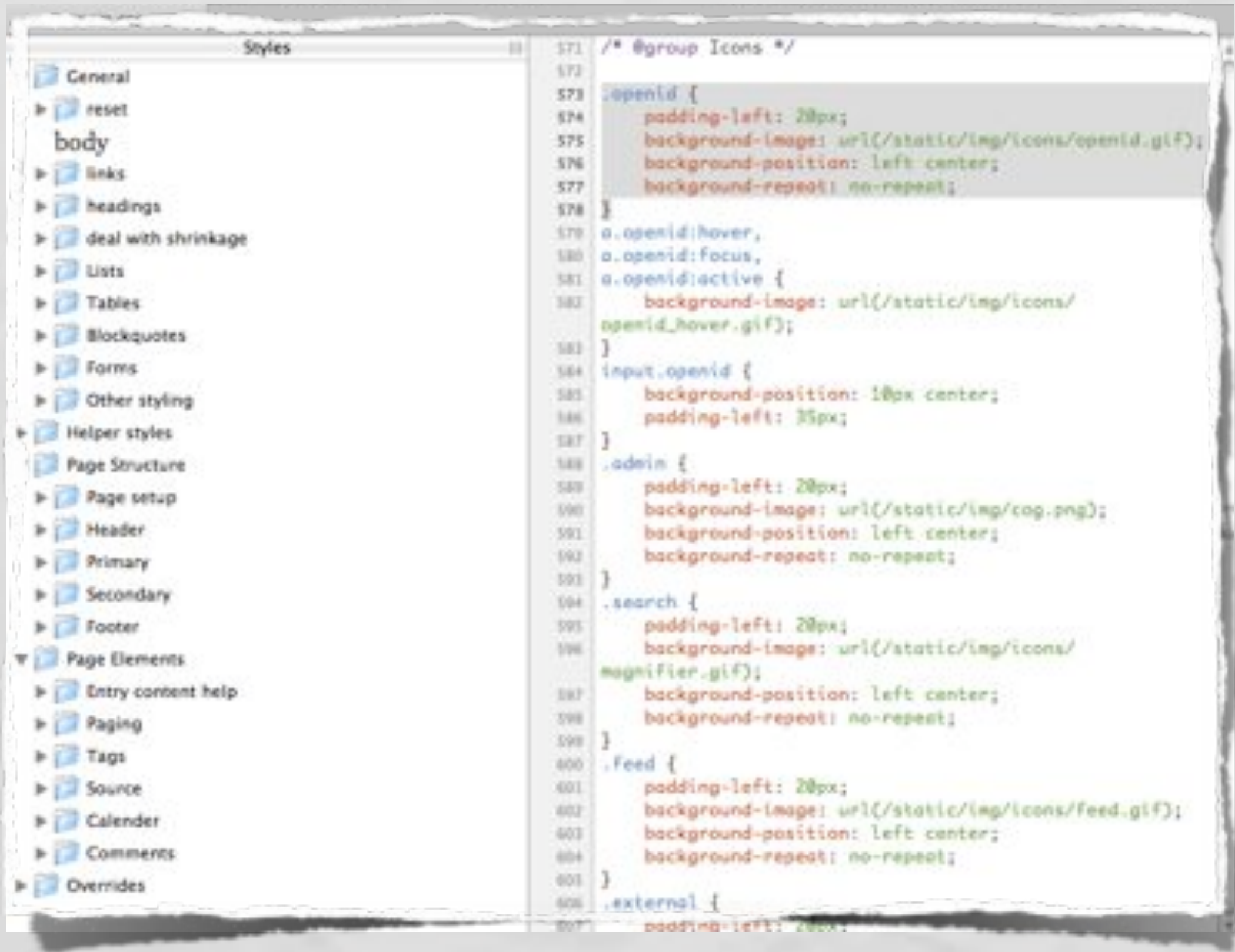
... the **page structure** refers to the header, footer, and the generic layout blocks that make up a basis for your components to fit snugly into.

**page components** is where most of the magic will happen.

any over-rides for page specific styles go at the **very bottom** of the page, though with some luck and good planning you shouldn't need to use this very much

I avoid putting things in here as much as possible, concentrating more on the helper styles and page components, **creating our system** ...





## CSS Edit

[macrabb.it.com/cssedit](http://macrabb.it.com/cssedit)

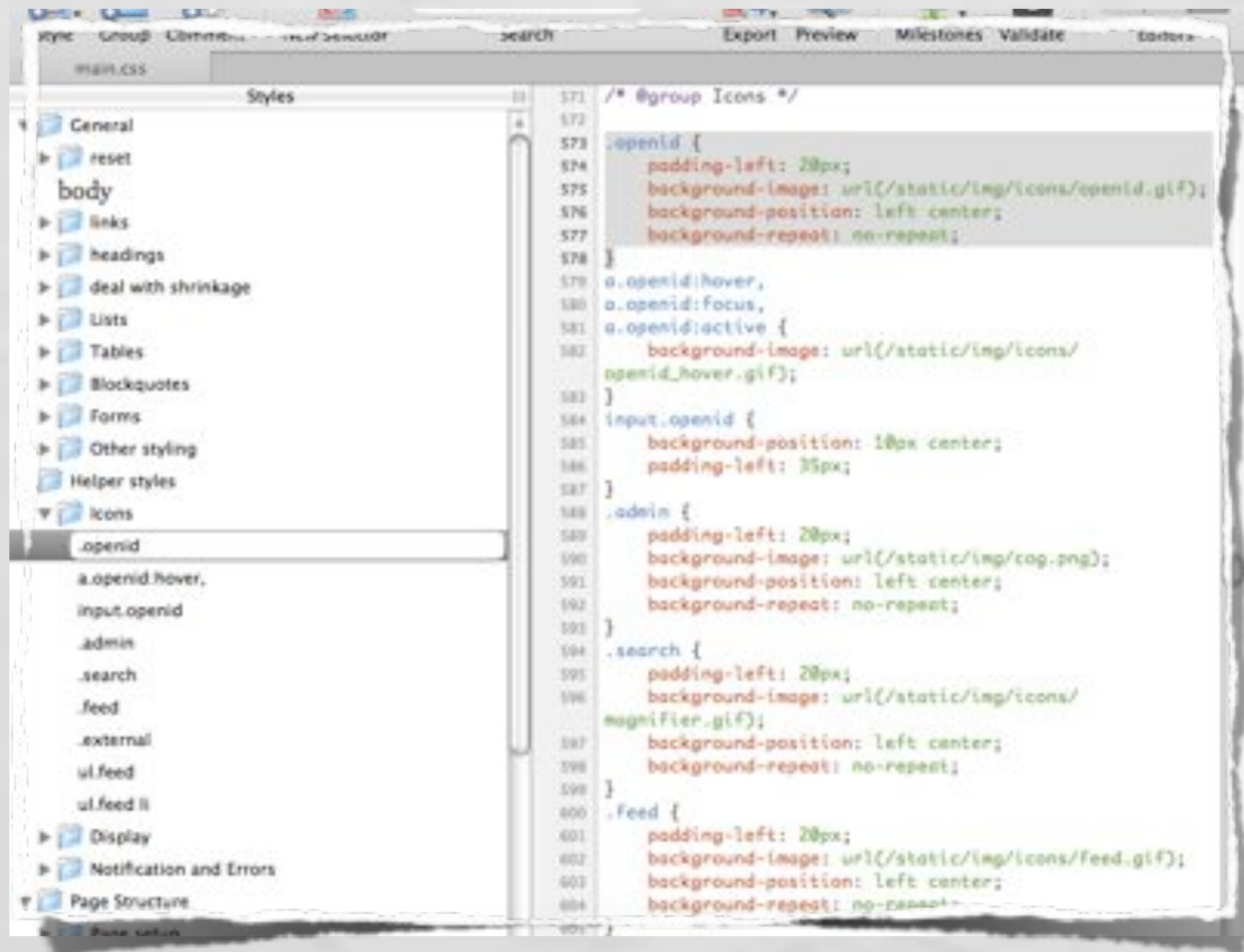
20

... CSS Edit is a development tool for the mac that makes seeing your groups a lot easier

you can see here on the left hand side it has displayed my nested groups which you use as a navigation aid to traverse the CSS on the right...

<http://macrabb.it.com/cssedit>





## CSS Edit

[macrabb.it.com/cssedit](http://macrabb.it.com/cssedit)

... and if you expand a bit further you can even see in the index on the left hand side the first line of your selector which is incredibly helpful. ...

# CONDITIONAL COMMENTS

- **Never** use browser hacks
- Browser specific stylesheets for all versions of IE and another for IE6 and under
- If you have to entirely re-engineer for IE, overriding your standards ready CSS in a conditional comment, you're *Doing It Wrong*

22

... how you cater for internet explorer depends very much on your site's statistics, I am sure we all dream of a day where one day, we wake up and none of our users are on IE 6 any more

until then though we have conditional comments, I am sure none of you use old Skool browser hacks that exploit vulnerabilities in the rendering engine, but just in case I'm going to say

Never use browser hacks and certainly not browser sniffing.

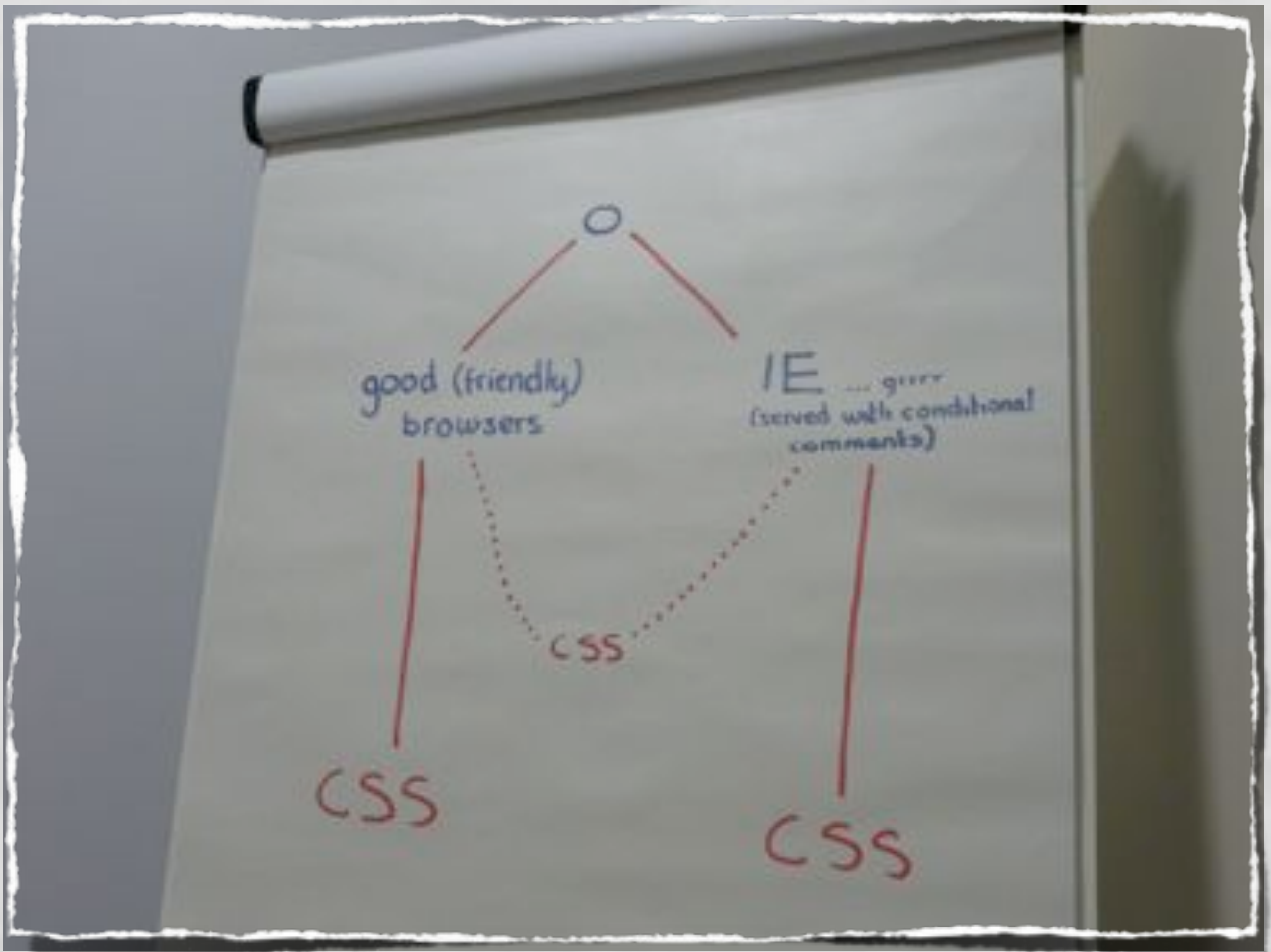
again, depending on browser support this might vary, but what we do is include one file for all IE's and another for IE6

one last point here. if for the page structure or even a page component, you feel the temptation to completely re-engineer how you are **skining that particular cat**. then you are **Doing It Wrong**. ...



please look  
to my right  
for a moment





24

... overriding a css solution with a specific implementation for IE in conditional comments is lazy. you will be hard pushed to juggle your inheritance in the future and will be forced to maintain both branches (or forks) of your code. twice the work

the solution, well a way of avoiding it is to test constantly in several main browsers, **dont** get to the end and then open up IE. **really**.

If you have to scrap how you are doing something and sacrifice a little markup thats better than having to branch your code.

But remember, things do not have to look the same in every browser, if you can get away with progressive enrichment and not have rounded corners for IE for example then thats ideal.

also, issues with maintaining branches may be trixy though they are more preferable to outright **CSS 'hacks'** ...





# BEFORE YOU START

So I have a design ... what now?

... you have a design, a lovely comp from fireworks or photoshop, what now? ...

# look at the basics



# will it be liquid? elastic? or fixed?

27

... a decision you need to make as a team is whether the site will be

**liquid**: scales with the viewport, so when the browser is resized

**elastic** scales with the font size or

**fixed** layout remains the same.

Im not going to cover the advantages and disadvantages of each, thats another talk all in itself

I am going to say that it certainly isnt true that one is traditionally ‘more maintainable’ than others if done right and you follow a few simple rules ...

# MAINTAINABLE LAYOUT

- if you are using an em or fixed width layout, try to only set a **width on the container**, using percentages inside
- a **max-width** of 100% will stop your layout creeping out the side of the viewport
- up and down **font size** all the time
- be **afraid of heights**, vertigo is healthy on the web. NEVER use height in px on anything with text inside

28

...

- if using an elastic M based layout or even fixed width then try to only use widths on the overall container and percentages inside,

this will save time if needs change later down the line (watch your padding here though)

- set a max width of 100% on elastic designs to make sure your layout doesn't scale outside your viewport and cause a crawl bar

- you should be upping and downing your font size all the time in every browser to see if behaves as expected

even if you are not doing an elastic design.

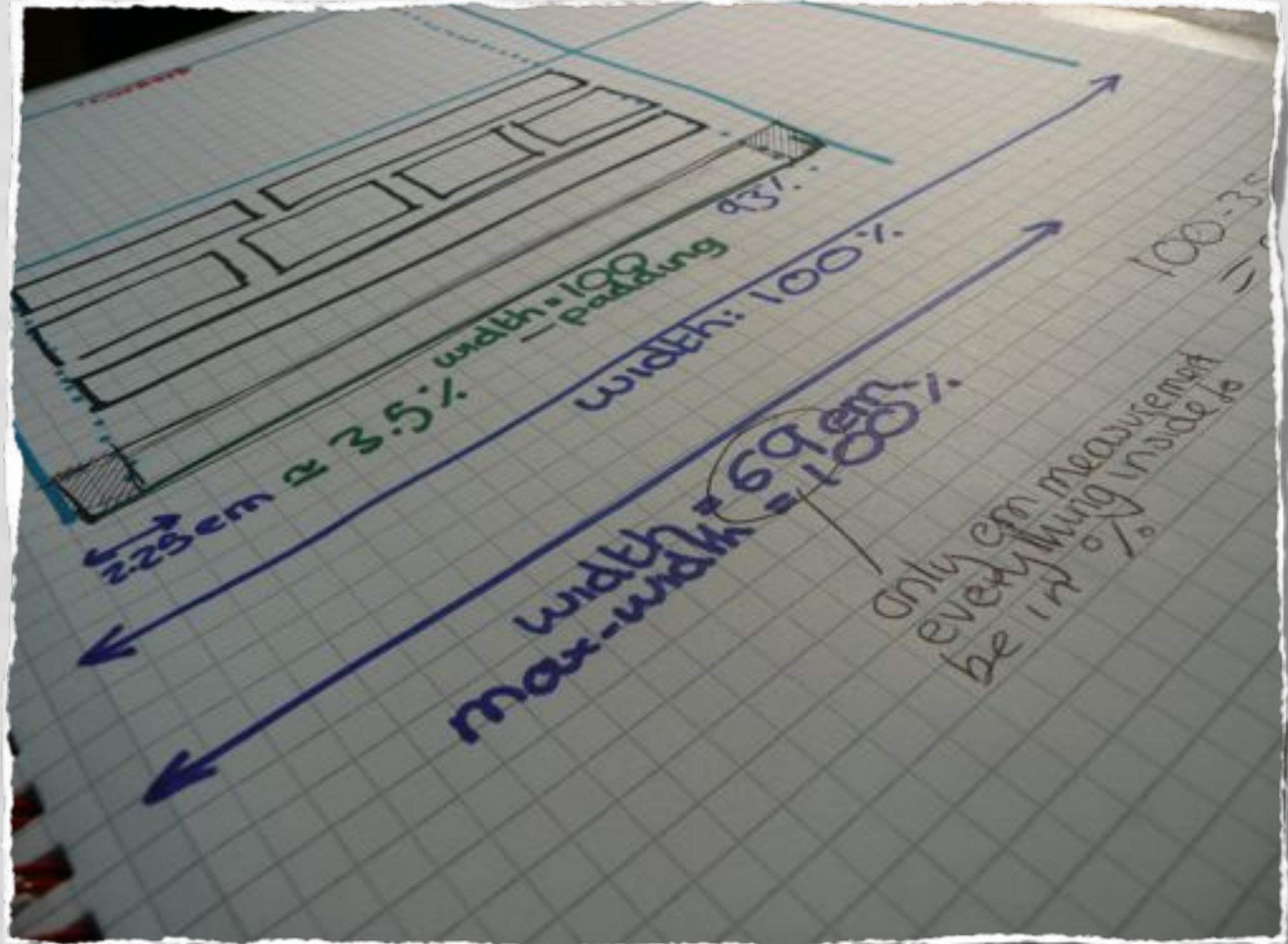
- vertigo is healthy, height on the web is difficult to get right, one surprisingly common thing people do wrong is specify a height in pixels on an element that contains text,

this is either going to bleed outside in the case of overflow visible, in which case why are you setting the height

or if you are using overflow hidden it vanishes which is worse.

...





... here is a planning artifact I made recently for an elastic site

I set the container to have a width of 69 M's and inside the measurements are in percentages

this does require a little more calcuation in the outset but overall the benefit outweighs the negative.

...  
...

# the grid

30

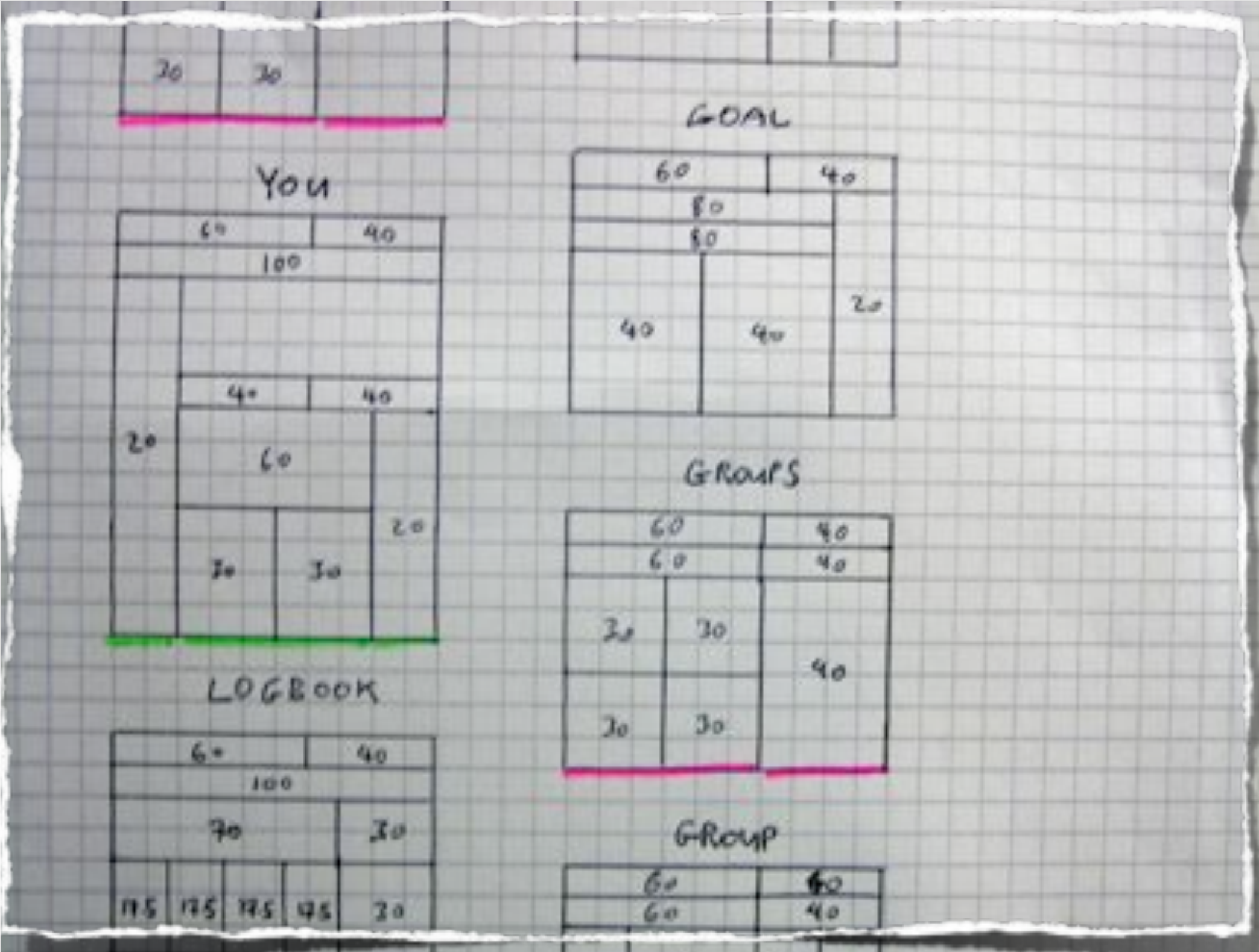
...

next, the Grid

– this is a design term, you can read about it on all the trendy design blogs

...

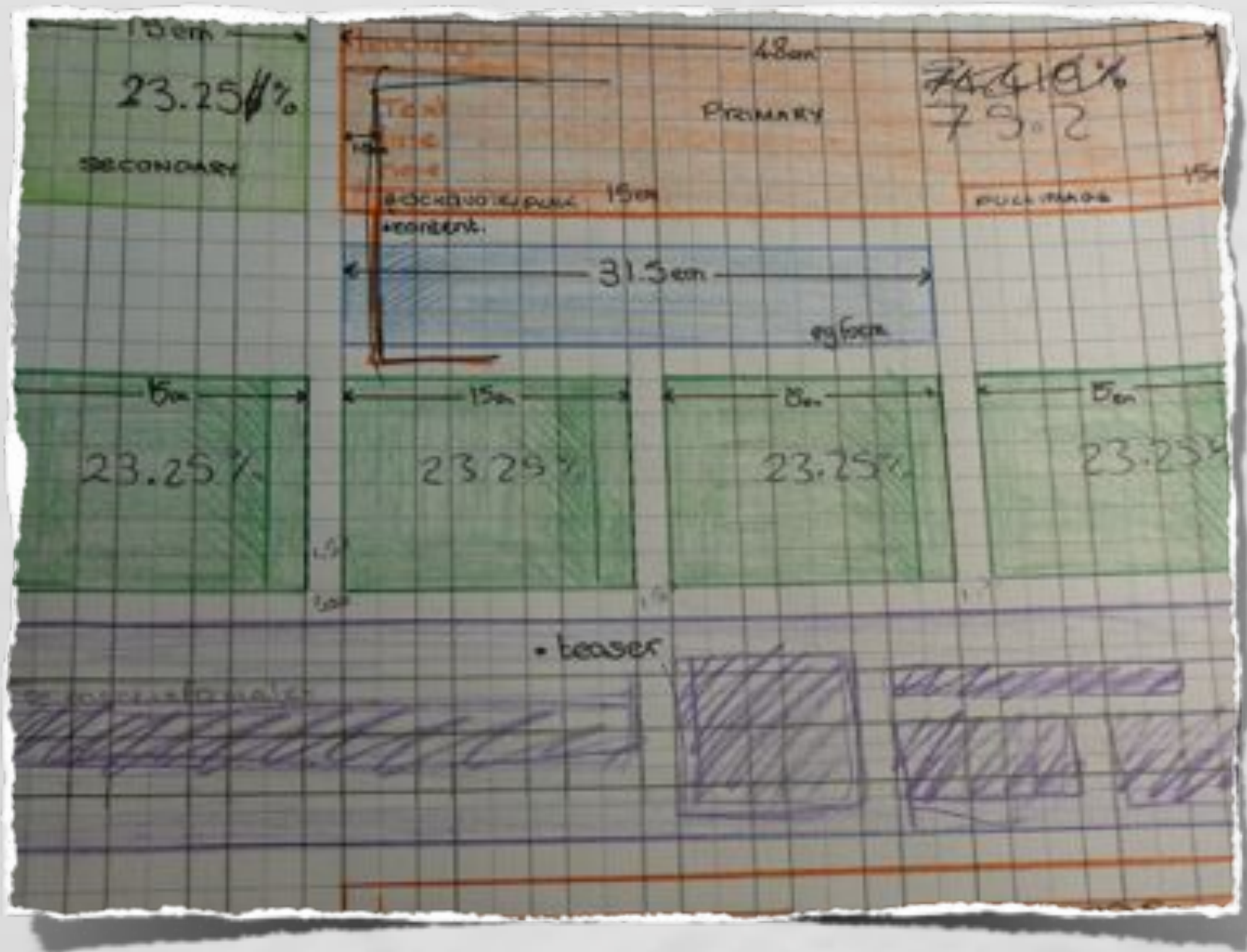




draw diagrams

... draw diagrams ...

<http://flickr.com/photos/adactio/1799116343/>



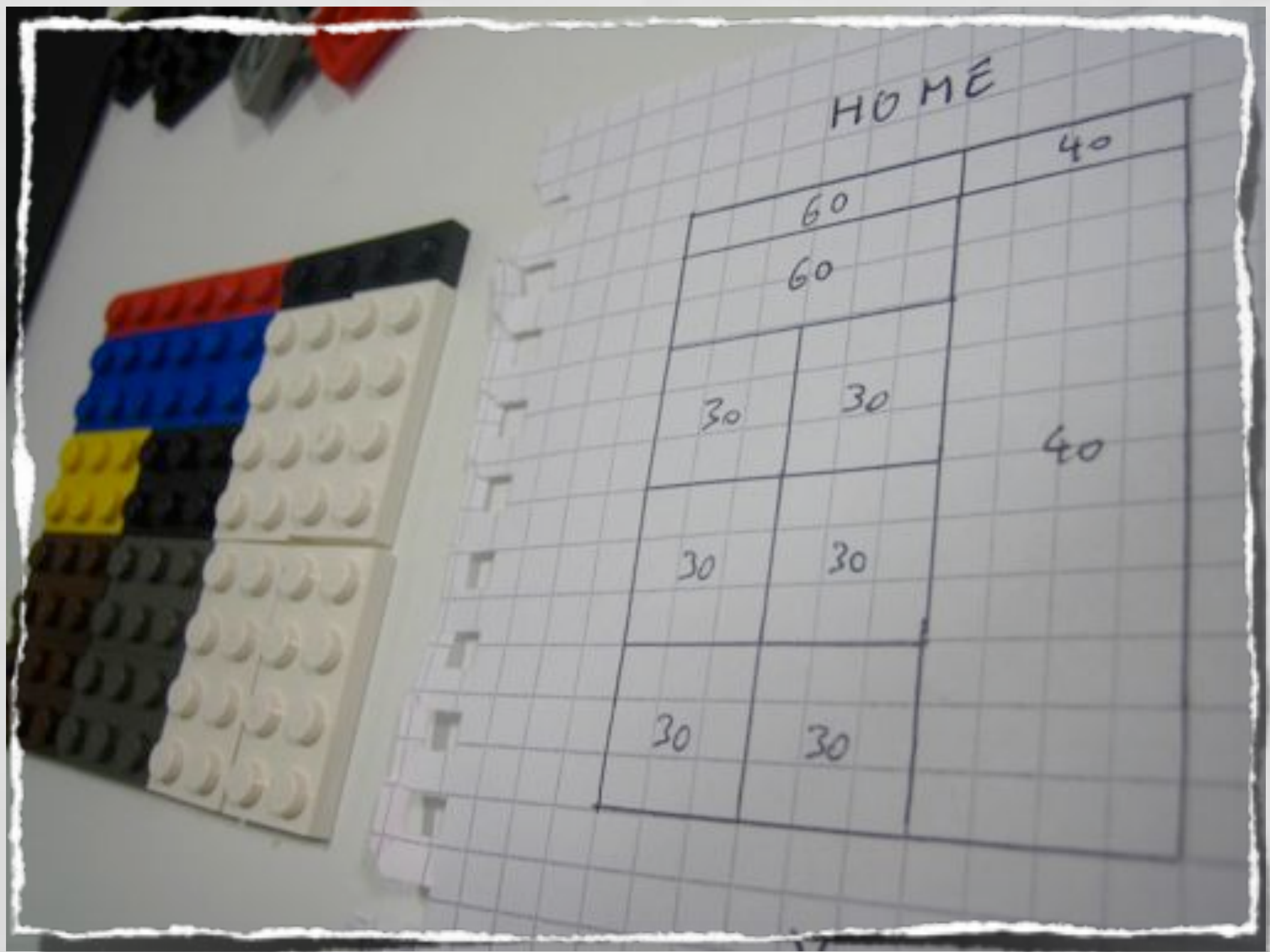
lots of diagrams

... lots of diagrams,

it basically boils down to **proportions**, analyse the site as a whole, look at **content areas** and see how these **interact** ...







get inventive

34

... use lego if it helps, make models if it helps you to understand how the grid fits with different types of content. This is a technique my colleague Jeremy Keith is experimenting with in this photo ...

<http://flickr.com/photos/adactio/1799953270/>



# levels of headings

35

... If you figure out your heading levels from the start this will help decisions later on. ...



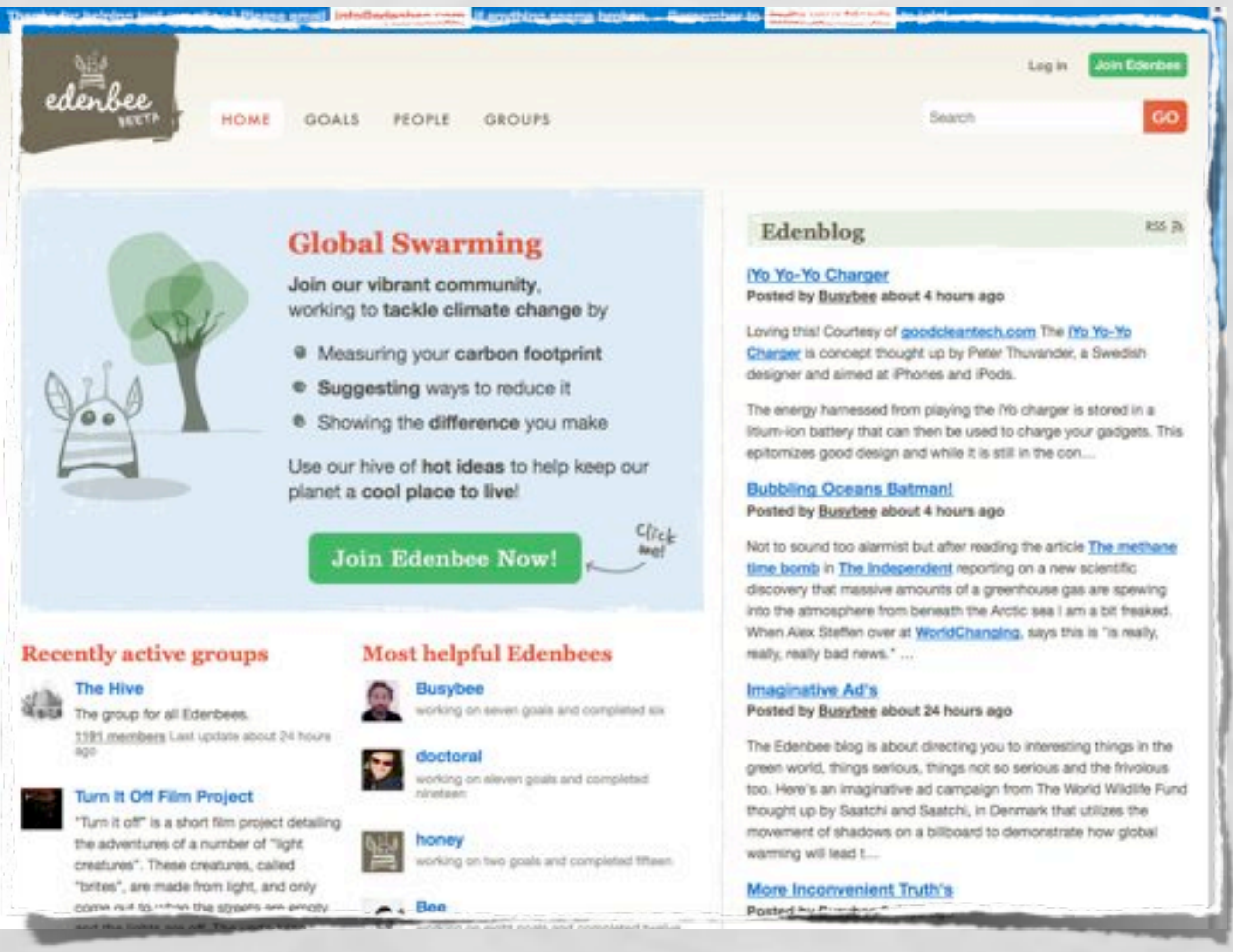




# page structure and components

37

... now, looking at the site as a whole, we need to identify the page structure, and content areas, as well as individual components....



... for example, identify areas, such as the

- header
- footer,
- the page furniture
- main content areas structure.
- and any common patterns such as icons etc ...





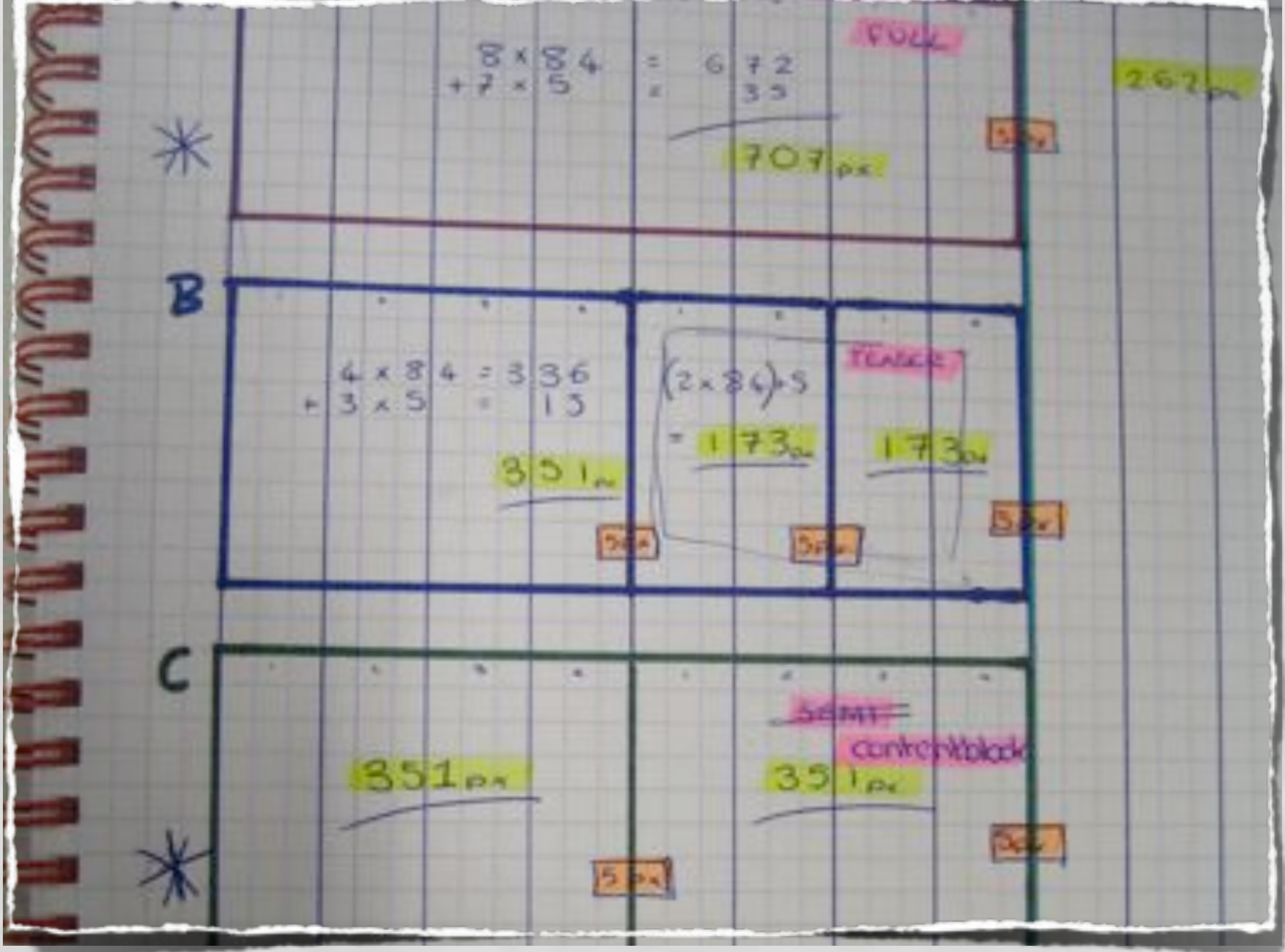
39

... look for **commonalities** in the design. If things have the same structure they should probably consist of the same markup.

you can always have variations on the commonalities depending on context

but always Concentrate on **what** the element is, not **where** it is .....





40

... For example

Here I've identified how chunks of content fit in to different layouts on the page. It's really important to continue to reference the basic grid.

the shared vocabulary for this site design defines a teaser to be two blocks wide, here a teaser is next to a content block.

this design was really interesting because the structure and markup was the same for a content block, a teaser or even a full width component. a class name of the type of content was all it needed to change the layout.

I was also using a class on the block of the section name, the sections all had different colours and this over-rode the default colour depending on which section the teaser was for.

I am quite proud of the resulting CSS system in this site, but unfortunately I am under NDA so I cant show you any more than this.

How your content blocks relate to each other and whether they affect the overall layout will depend very much on the nature of your site. you are producing a unique CSS system

...

<http://flickr.com/photos/adactio/2679956143/>





# MAINTAINABILITY MYTHS

Any existing 'best practices' or approaches?

41

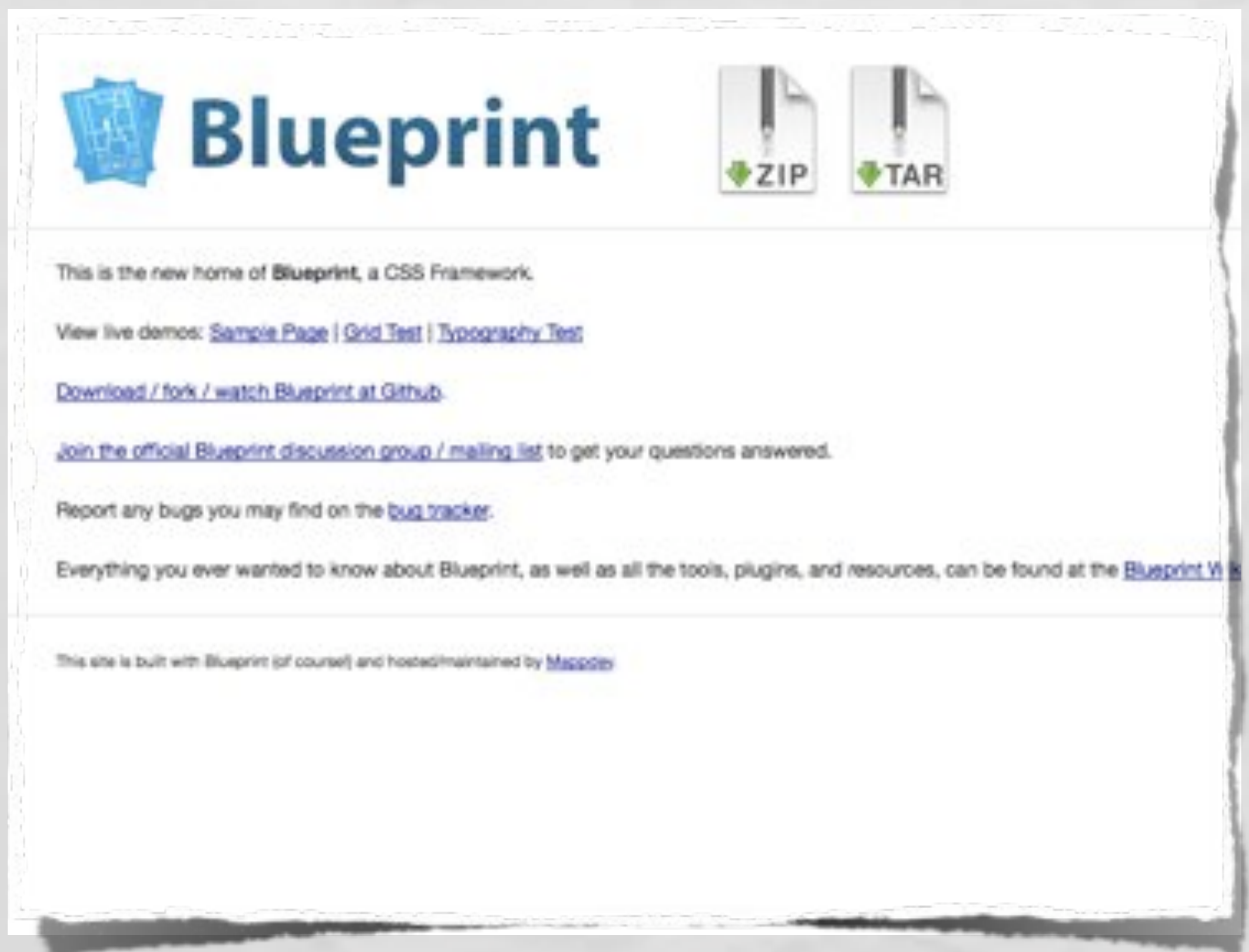
... So let's talk about some maintainability myths.

Lots of people have developed and shared ideas for improving CSS maintainability in the past; unfortunately, many of these techniques fail to integrate with this larger idea of a CSS system for an individual site.

Firstly, I'm going to mention ...

<http://www.flickr.com/photos/46425925@N00/521605639/>





## CSS frameworks

42

... css frameworks

I feel the need to define what I call a framework. For me this is something that alters how you write HTML itself.

This is different from a library, which simply provides individually reusable components.

This is Blueprint. Blueprint is one of the worst offenders (along with YUI grids – sorry guys) when it comes to encouraging your HTML to directly reflect the design of your site.

In my opinion, CSS frameworks **decrease** the maintainability of code.

Simply introducing this extra abstraction now means that to maintain the site you have to know not only CSS but also the framework that was used – **down to the specific version**.

Frameworks are made for ease of development not maintainability. You are introducing presentation back into the markup and the classes often dont make sence out of context, reducing the usability of the code.

They have their place – they are great for wireframing quick sites where you don't care about the markup. They're basically akin to using Dreamweaver. ...



```
@import url('base.css');
@import url('layout.css');

/* LOGIN FORM */
body.login { background:#eee; }
.login #container { background:white; border:1px solid #ccc;
width:300px; margin-left:auto; margin-right:auto; margin-top:
.login #content-main { width:100%; }
.login form { margin-top:1em; }
.login .form-row { padding:4px 0; float:left; width:100%; }
.login .form-row label { float:left; width:9em; padding-right:
height:2em; text-align:right; font-size:1em; color:#333; }
.login .form-row #id_username, .login .form-row #id_password
.login span.help { font-size:10px; display:block; }
.login .submit-row { clear:both; padding:1em 0 0 9.4em; }
```

## Single line declaration blocks

43

...Some schools of thought think that writing all of your CSS on one line with the selector and rules is better practice because it **focuses the author's concentration on the selectors**.

Whilst I agree that the selectors are indeed more important, I disagree strongly with this method for maintainability. not least because it introduces horizontal crawling.

**Multiple lines also work much better with version control** tools such as Subversion so you can identify the actual rule that has changed.

This also causes issues with **debugging tools** that refer to specific line numbers, such as Firebug.

Also, tools such as CSS Edit **alleviate the need** for this solution by showing all of the selectors down the left hand side. ...







colour.css / layout.css / typography.css

45

... this concept focuses on splitting your CSS in to separate files based on the **nature of the rules**. so to colour, layout and typography (point at ducks)

and surprise ... I don't like this method either.

Splitting colour can make sense for some sites (those that support **themes**)

but I'm sure most pedantic typographers would agree that **typography is very strongly linked to layout**. Changing font size affects dimensions specified in ems, for example.

By splitting a rule block for a component it also adds extra **mental overhead** in deciding which file a rule should go in...

<http://www.flickr.com/photos/iwantamonkey/2335537618/sizes/o/>





# ON YOUR MARKS ...

You have your design and your plan - you're all set!

46

... you have a design, several beautiful planning artifacts, you are ready to start development  
some things to bear in mind ...

<http://www.flickr.com/photos/22839942@N00/2498909856/>





use the cascade

47

... use the cascade (now with added sheeting action)

don't heavily namespace or **sandbox** if you can avoid it, sandboxing is the practice of using very specific paths to identify elements.

for example ...

<http://www.flickr.com/photos/roadsidepictures/497157632/>



```
div#page div.teaser ul.products li p.name
```

48

... not like this.

this strongly couples the structure of your markup with the css, adding an extra level of dependancy that decreaces long term maintainability.

...





ul.products p.name

49

...This is better

think in terms of components,

phrase your selectors based on WHAT it is rather than WHERE it is.

Going too deep in specificity or inheritance too quickly also makes it **harder to override** later on.

incidentally I often keep the tag names intentionally to **avoid style leakage** and

for future reference if you know what the element is, you understand what the **default styles** are, like margin, padding etc.

If you really have to heavily sandbox something, put it in the overrides section. ...



you love to float

50

...For me, the argument about whether to use floats or absolute positioning for layout was over a long time ago.

Absolute positioning takes elements out of the flow of the document. You lose robustness, because your elements no longer take each other's positions in to account.

The combination of resizing your browser, resizing your fonts and variable height content make absolute positioning for layout virtually useless in practice.

Always clear the container of your floated elements of the content within them, if you dont, this can cause problems later on.

Use a method you understand. I dont like using the 'clearfix' technique of inserting content to clear floas because it feels hacky ... it smells bad, my preference is for width and overflow hidden to force clearing....

<http://www.flickr.com/photos/12148520@N00/2630867950/>



## LESSON TWENTY-FIVE

### SIGNS FOR H

Sign	Letter	Name	Examples
	H	Upward Hay	happy  head  heat
	H	Downward Hay	he  high  higher
	H	Dash (or Tick) Hay	home  whole  hear/here

Upward Hay is generally used. Downward Hay is used when H is the only consonant sound in a word, and in words beginning with high. Dash Hay is used before M, L and R.

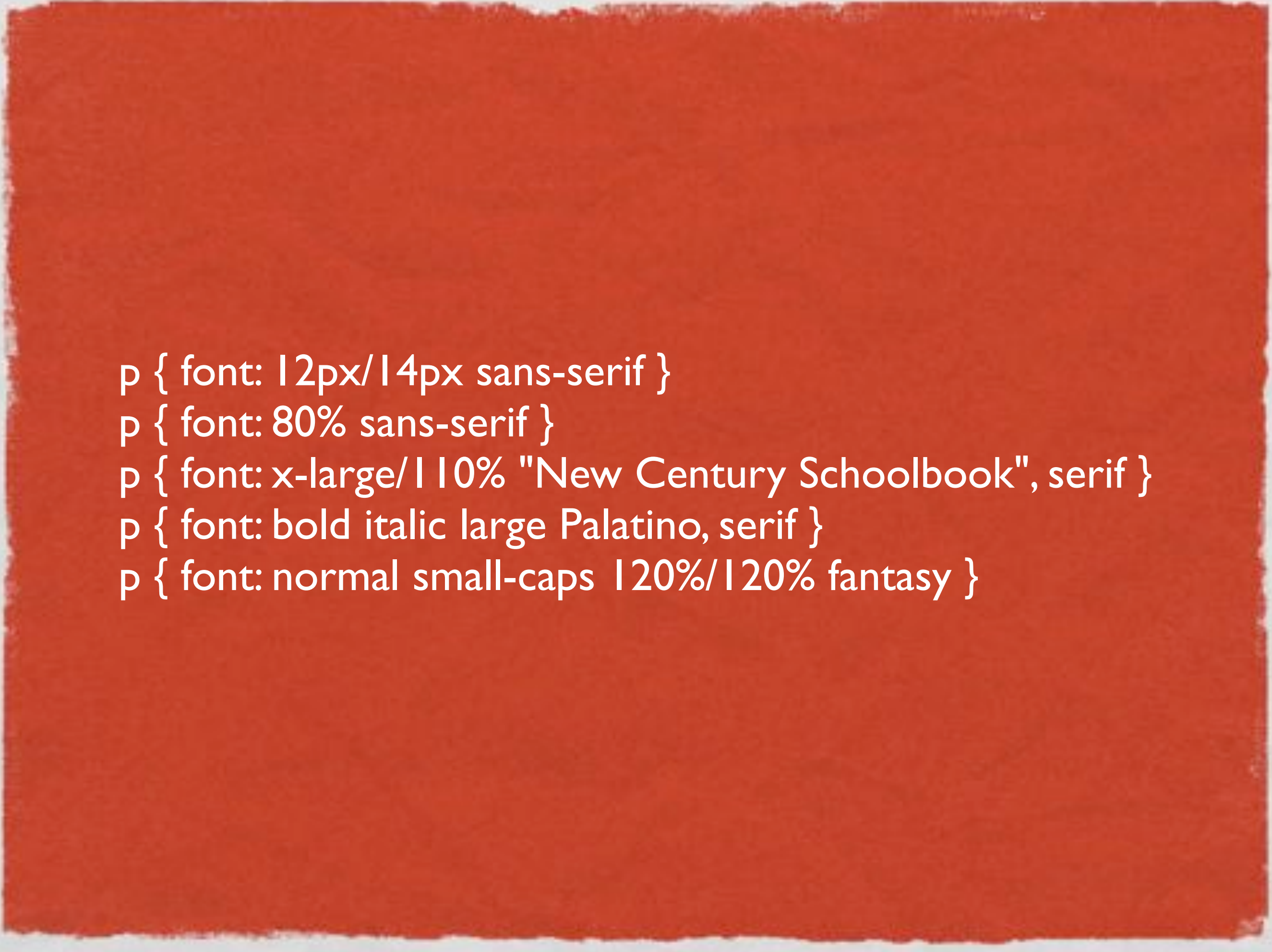
### OUTLINE DRILL

(a) HEAD HOUSE   
 HEAVY HEAT   
 HOPE

shorthand?

... now I know a lot of people believe you should always use shorthand notation ...

<http://www.flickr.com/photos/teresa-stanton/2251386346/>



```
p { font: 12px/14px sans-serif }  
p { font: 80% sans-serif }  
p { font: x-large/110% "New Century Schoolbook", serif }  
p { font: bold italic large Palatino, serif }  
p { font: normal small-caps 120%/120% fantasy }
```



```
body p {  
    margin-top: 4em;  
}  
  
p {  
    margin: 3em;  
}
```

```
body p {  
    margin-top: 4em;  
}  
  
p {  
    margin-top: 3em;  
}
```

... margins or background.

I dont. necercarly.

Longhand can often be preferable, particularly with background.

firstly it makes debugging in firebug easier, you can see where overrides are effective.

you will also have similar issues as the 'one ruleblock per line' when it comes to source control. you can tell that a line has changed but its not imediately obvious which part of that line.

but mainly and this is more noticable with backgrounds ...



```
background: url("w00t.png") orange 0 50% repeat fixed;
```

54

... that by using shorthand its easy to override declarations without meaning to.

For Example if you are using all the sections of the background like we are here, then its easy to see what this is doing,

but if on an element you have already specified it as ...





```
background: lime;
```

55

... having a delightful lime green background, and you then you want to have a variation of that inherit the lime styles but also have a transparent background image ...



```
background: url("rofl.gif");
```

56

... rofl.gif

if you specify it using just the background declaration, then this overwrites the lovely lime green colour

one solution of course is to specify the colour here too, but then ‘lime’ is in two places, you are not taking advantage of **inheritance within declarations**,

and what if you suddenly want to change to pink, now you have to change it in **two places**, this is not optimal

also always bear in mind that people editing your code are likely to be **copy and pasting** from the rest of the file,

so you have to set a good example with this throughout your code.

so a better solution would be ...





```
background-color: lime;
```

```
background-image: url("rofl.png");
```

... use the longhand version, and override with subsequent longhand versions ...





# PREPARE FOR THE WORST

n00bs will one day be let loose on your code  
(accept it)

... n00bs will one day be let loose on your code,  
you cant stop the inevitable, you can only prepare for it.

<http://www.flickr.com/photos/85088843@N00/519904699/>



# clear your footer

59

... simple tip, always put **clear:both** on footers

even if you are clearing your containers ok now, that is **no guarentee** someone taking your code and altering it in the future will be.

if you are clearing your page footer, then its **one less thing that can go wrong** ...

# careful with dimensions

60

... **avoid using width padding together** especially for your main page container where you might want to specify the width in M's and a maxwidth of 100% ...





# watch your height

61

... again. height on the web never ends well, just be careful and think about the implications if you use height at all.

also, please please **please** resize your text constantly while developing and make sure **everything** expands nicely ...

# hooks

62

... its ok to add hooks,  
extra divs for **future proofing**. so long as they make sence,  
just don't get carried away, otherwise you can end up with a bad case of **div-it-us** ...



# editable content

63

... editable content, for example product images,  
anything which might need to be changed in the future,  
should be in the source and not the CSS file. always ...

# sanitise text

64

... if there are headings that need to be in uppercase, **dont rely on the CMS or author** to do this, even if they are, put a text-transform declaration on the element.

just in case ...



# abstract your icons

65

... for icons, use a class and specify as a **background image** not as an image element in the source.

if the image path changes, you now only have one place to change it

the one exception for this is **accessability**, anything that adds **important context** or information should be actual image elements with proper alt text.

For example the **star used on labels** to indicate a field is **required**, NEEDS to be an image with the alt text 'required' ...





# DESIGN TO AVOID DEBUGGING

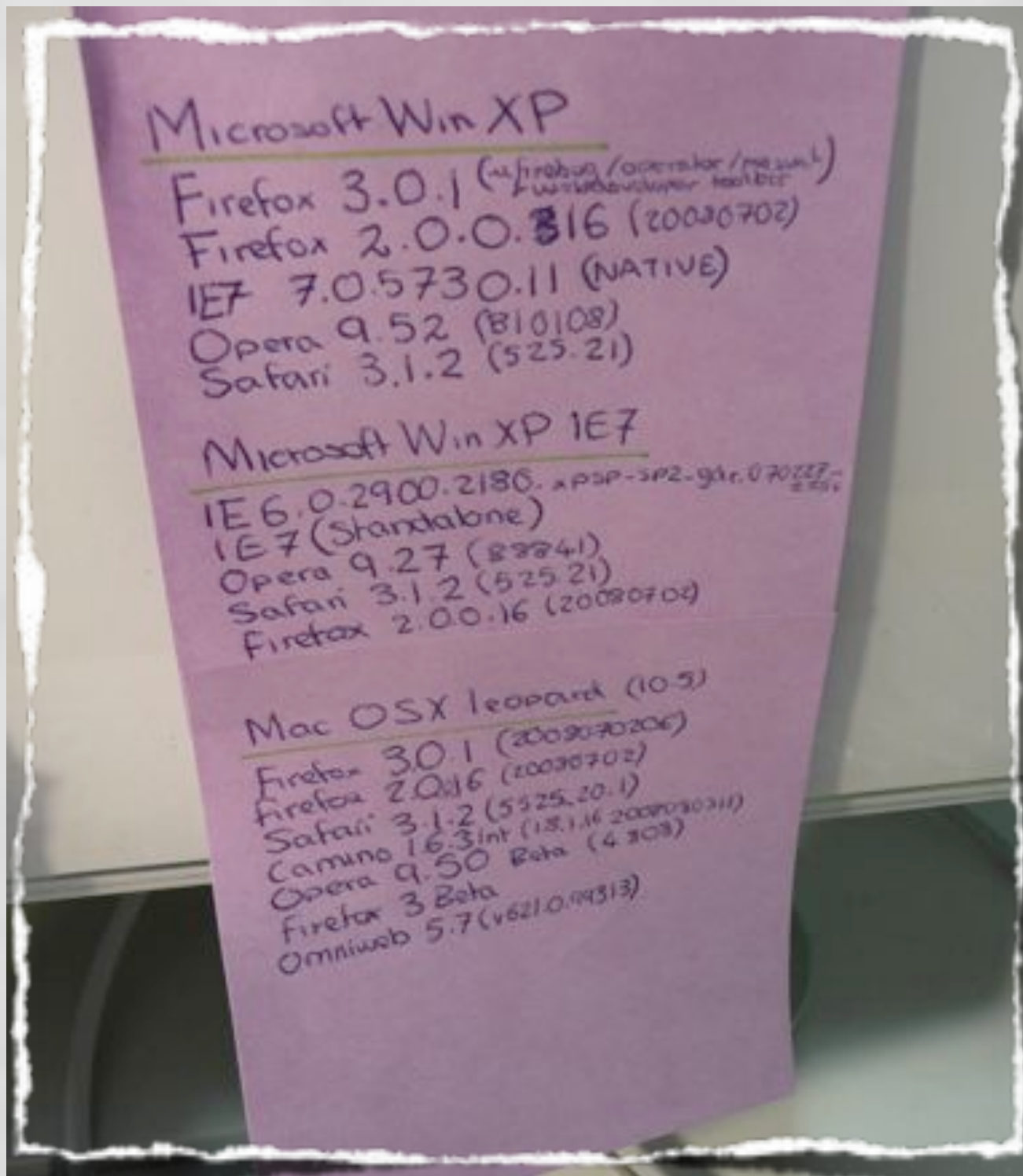
engineer your way around the browser

66

... there are some things you can do to avoid common pitfalls, think about the solution to browser bugs and pre-empt what you can ...

<http://www.flickr.com/photos/60359963@N00/475920725/>





## Simultaneously develop

67

... so the ideal solution to avoiding debugging is to **simultaneously develop** in the main browsers, so IE 6 and 7, firefox, Opera and Safari.

Perhaps have one of the standards based ones as your main development platform (mine is firefox) but dont leave it too long to test in IE or the pain is greater.

quick point to note, if you are using firefox 3 as your main development browser **turn off full page zoom**, you NEED to be able to see how upping and downing the font size effects your page.

toward the end of the project I turn back on full full page zoom and test in my hareem of browsers to double check things are all hunky dory. ...

# don't re-engineer separate solutions

68

... I've mentioned a bit about IE stylesheets, and conditional comments

one point I really really want to hammer home though is that if you have to re-engineer a separate solution for Internet Explorer.

STOP,

I know it's tempting but don't, you now have doubled the workload of anyone working with your code in the future, that's two solutions to debug ...



# floating

69

... when you float something, specify it explicitly as display inline.

sounds odd but this will fix the double margin float bug, where if you float something and have margins, IE will render twice the intended margins.

The fix wont affect any other browsers at all.

Even if you are not specifying margins on your floated element, someone might want to in the future, that someone might not even be testing in Internet Explorer (god forbid) or hopefully more likely they might not know about the fix ...

# you **need** to set psuedo selectors on links

70

... yes yes its **ugly** but you still need to be specifying all the psuedo selectors on links, so :link :visited :hover **:focus** and :active

this is because of **inheritance issues in internet explorer**, so if you start by specifying hover styles you will often get issues later on if you then tie styles just to the a tag

bugs with link inhereitance often come up, a **common solution** is to then use a high level of **specifity** to get over this and that can **worse** issues later on.

So by making sure you always specify the **psuedo selectors** when you style links this will help **alieviate** issues that might arrise ...



# buttons

71

... i could go on for days just on tips to avoid having to debug the shit out of IE, but I wont, really.

just one to say you will need to specify **overflow: visible** on buttons, so just **do** it, otherwise you will spend an inordernate amount of time hacking padding on button elements in a certain browser,

ok Im nearly done now, honest ...

# review

72

... alright, just one more last little point,

code reviews are invaluable, if you can **explain** your CSS system to someone else and have them **understand** it,

theres a pretty **good chance** what you have done is **robust** and **maintainable**. ...





# HANDOVER

what to give the client

... so now you have your CSS system, you have to pass across the baton ...

<http://www.flickr.com/photos/22839942@N00/2521967364/>



# THE IDEAL HANDOVER INCLUDES

- The **markup scheme**, a set of files demonstrating the different markup components
- The **CSS** itself
- Supporting **documents** that explain the system as clearly as possible
- A face to face **meeting** if possible

74

... The **markup scheme**, a set of files demonstrating the different markup components (or patterns)

I used to call these templates until I realised this term was polluted and completely befuddled the poor backend developers.

Think of it as a set of 'unit tests' for your CSS System, if you ever need to change the styles you have a small selection of pages that between them should contain all the components.

The CSS **itself**, if you got it right, the client will never have to modify this unless you change the design.

so you are giving them the code you wrote, but more importantly you are trying to **explain the system to them**.

the supporting documents can be seen more as a **covering letter**, a brief explanation of the main content areas and components.

If you can have a **meeting**, ideally face to face so you can show using **firebug** the markup patterns.

I am still trying to figure out the ideal handover, and if you have any ideas I would love to **talk to you later ...**



A large, textured red rectangle occupies the upper half of the page. The word "FINALLY" is written in white, bold, capital letters in the center of this red area.

# FINALLY

... and finally ...

ITS ALL ABOUT THE SYSTEM

**define it**

**develop it**

**maintain it**

**communicate it**

...

its all about the system

**define** the system  
**develop** the system  
**maintain** the system and  
**communicate** the system

...





# THANK YOU

... thank you ...



... does anyone have any questions ...