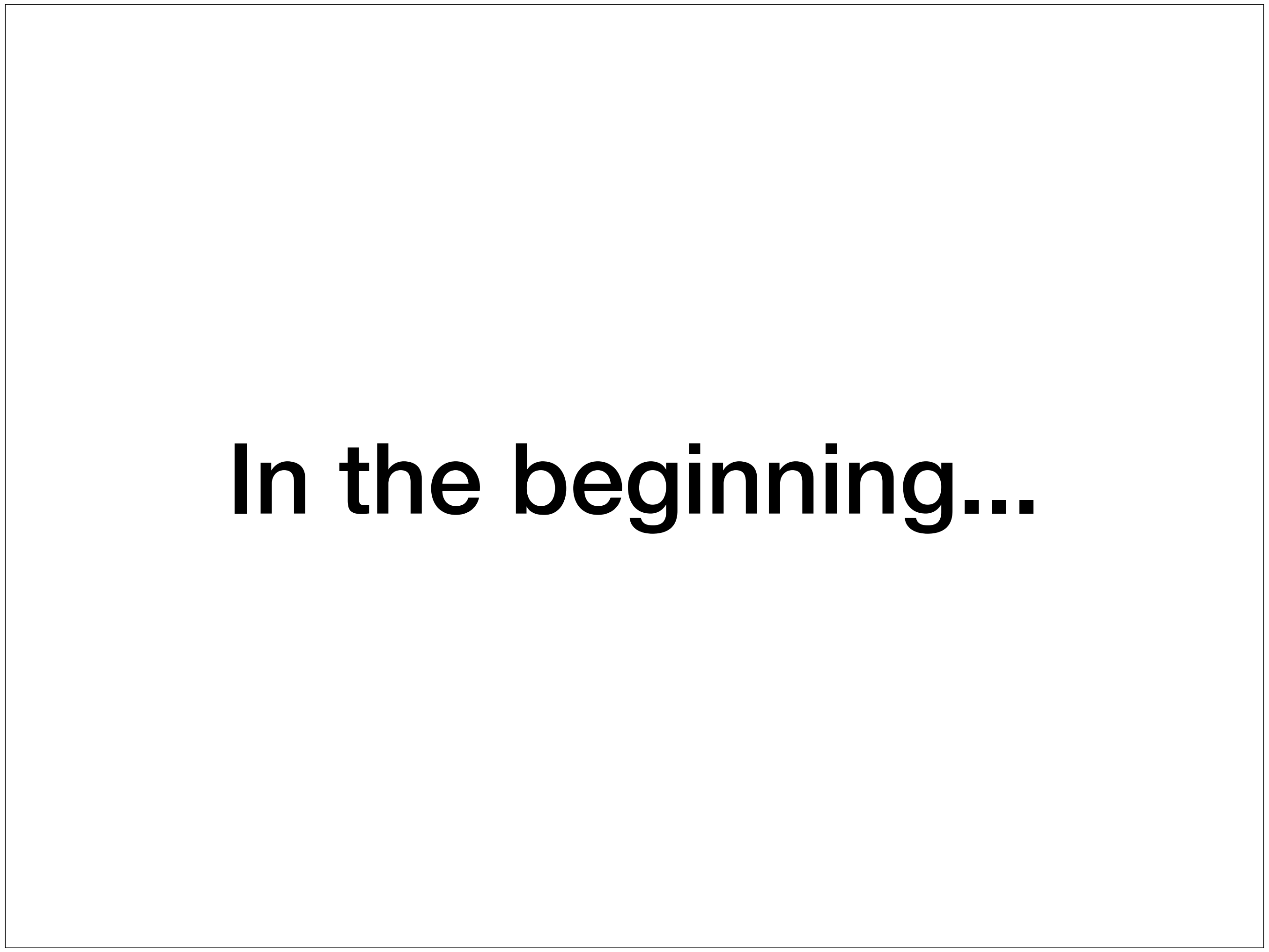
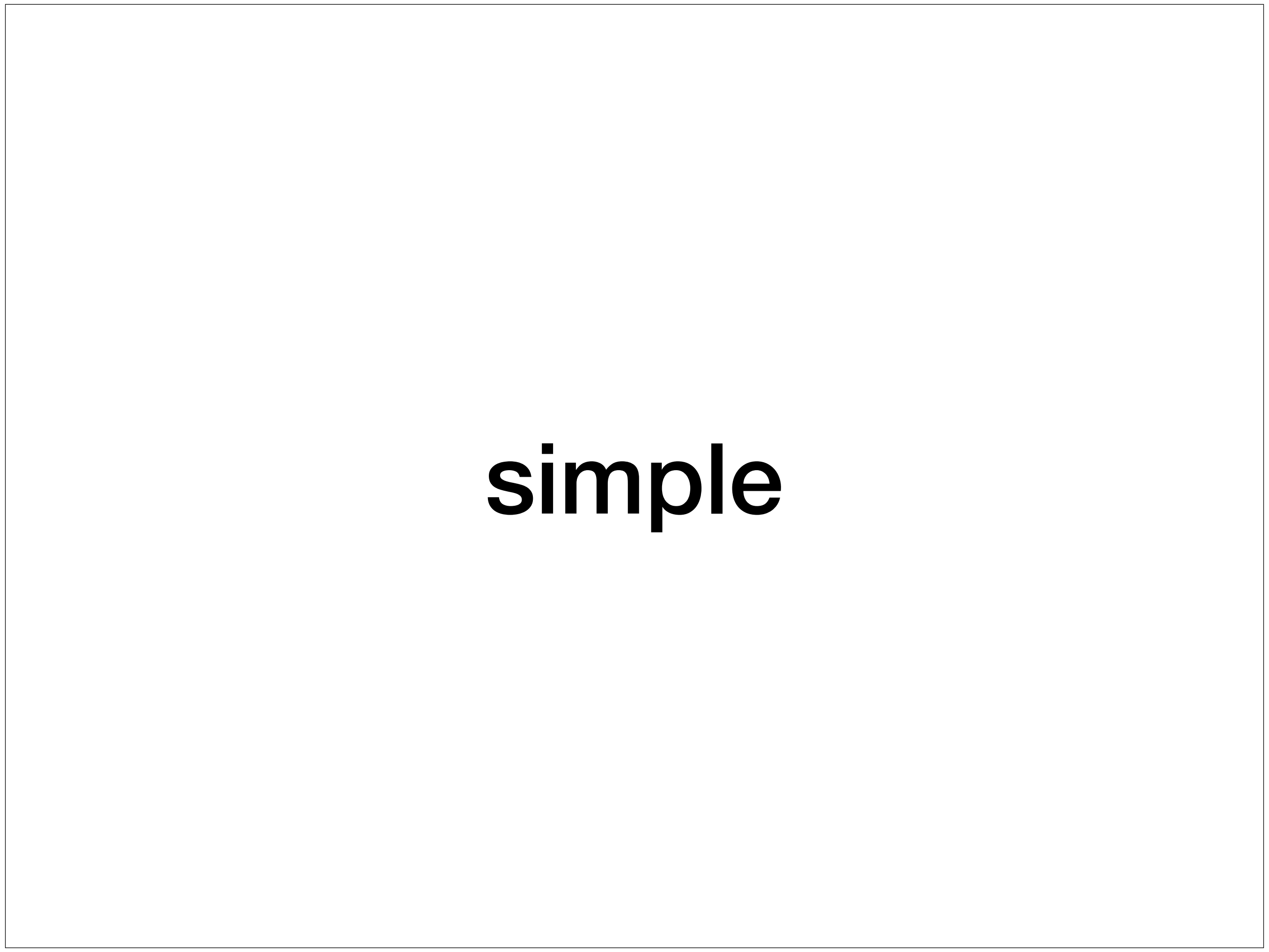
The Web is an Elegant Series of Hacks

Rufo Sanchez
@rufo
me@ru.fo



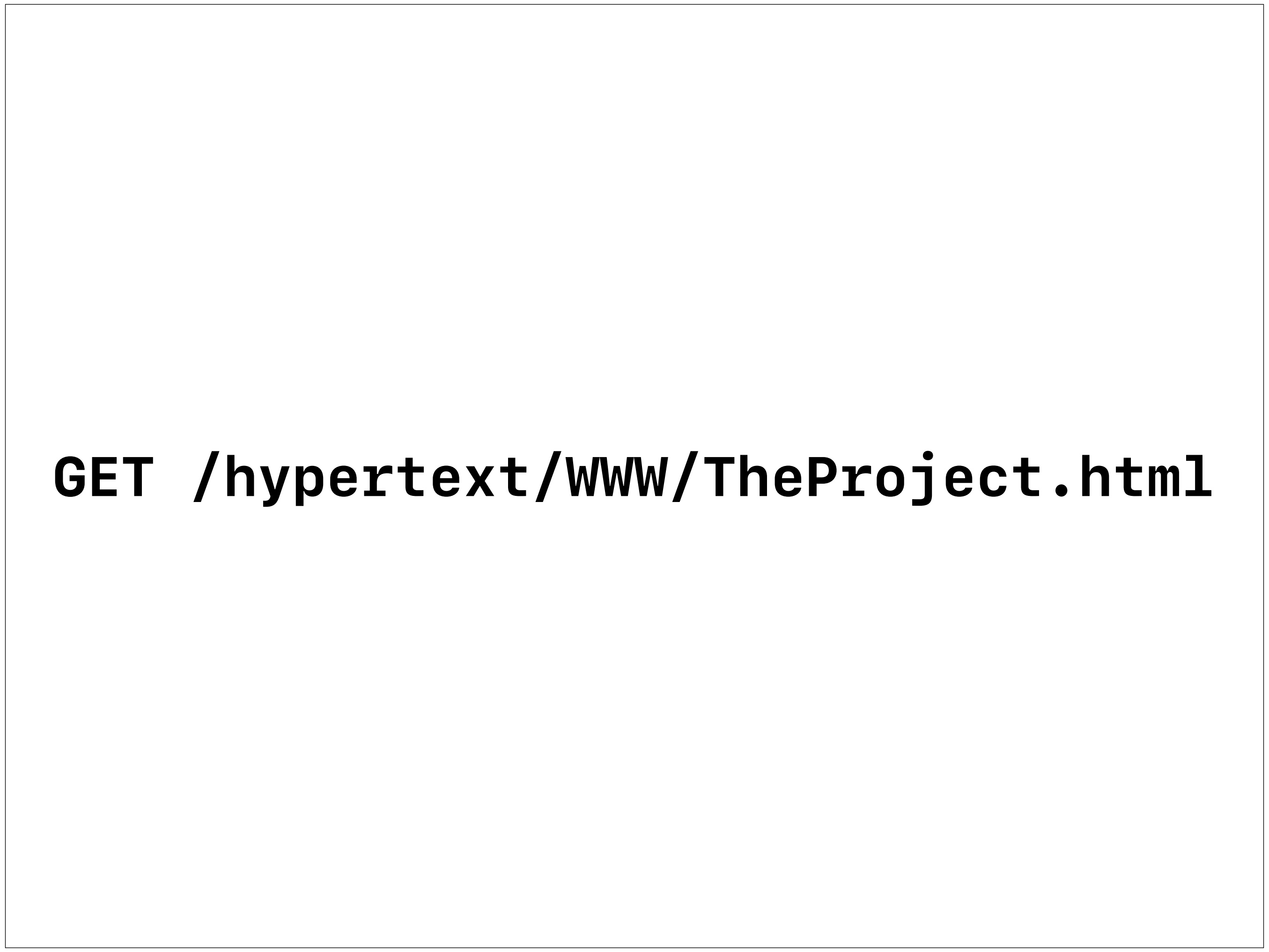
In the beginning...

In the beginning, the web was pretty



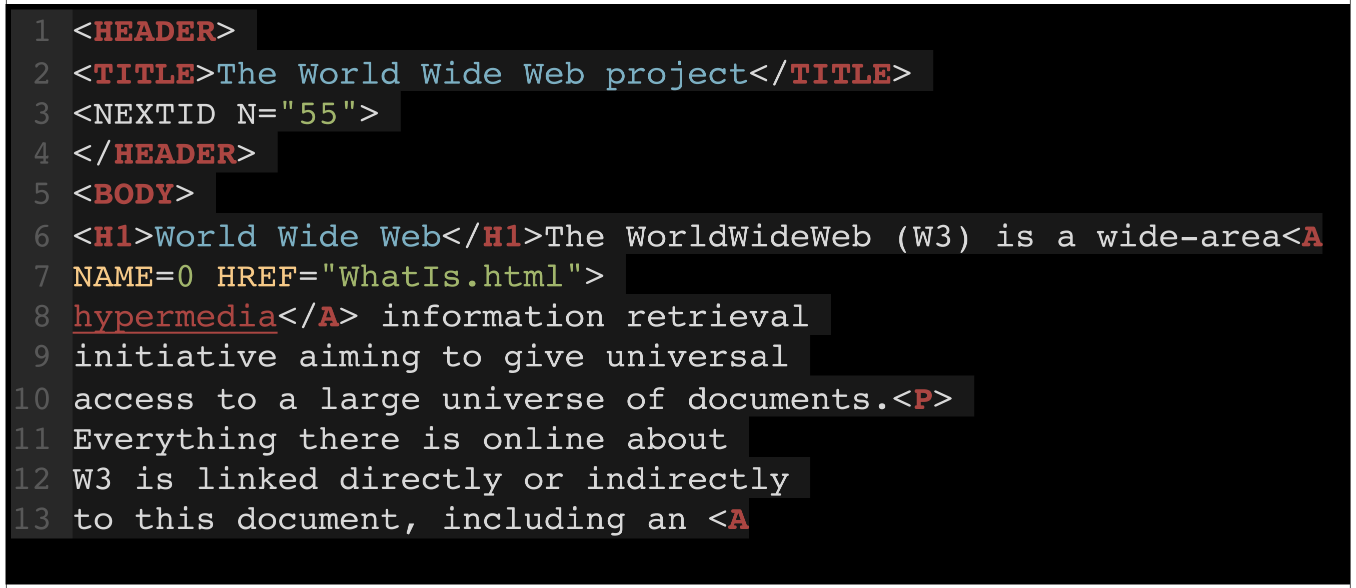
simple

simple. It had a simple server, that accepted a simple



```
GET /hypertext/WWW/TheProject.html
```

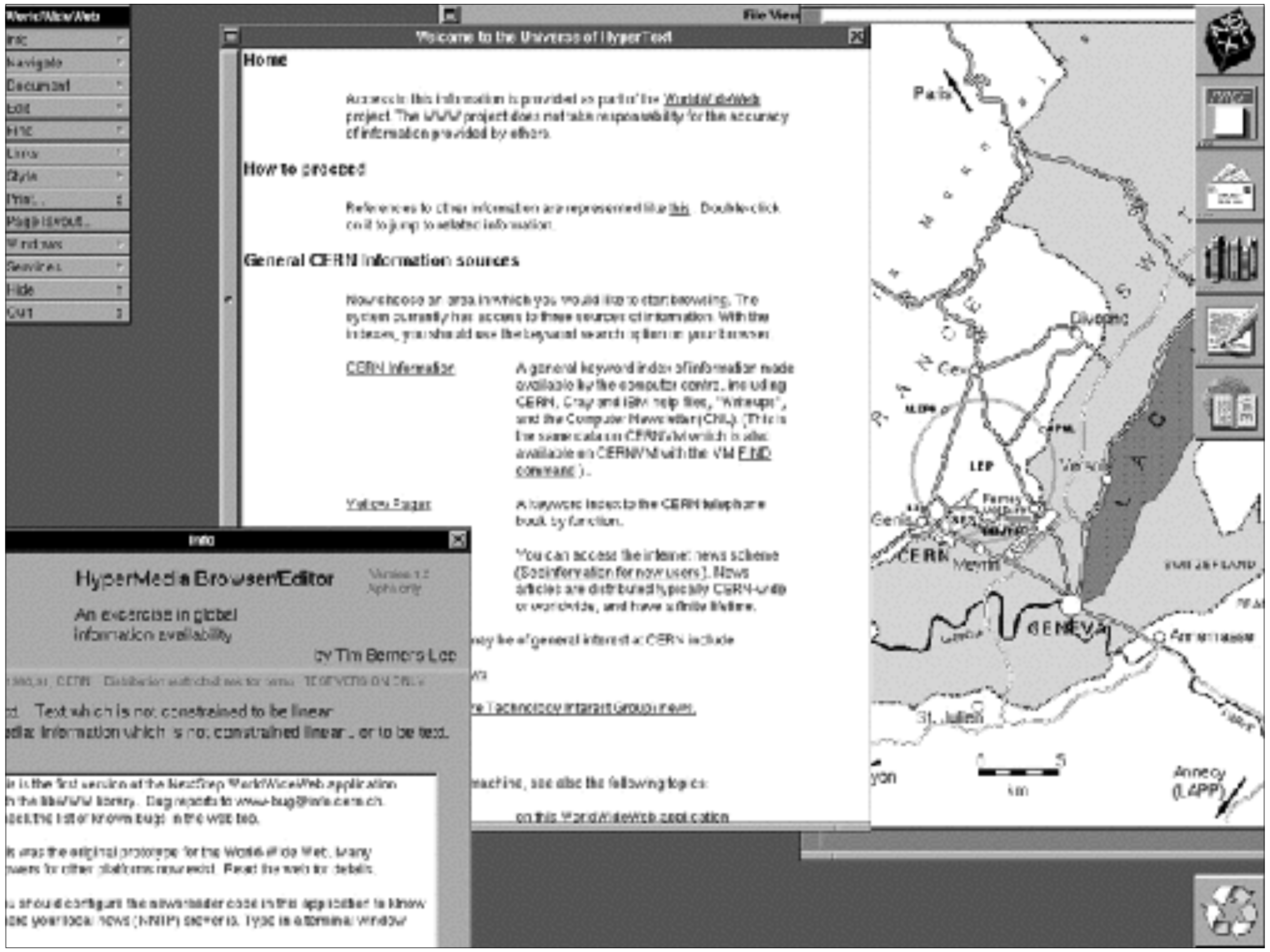
request. A request fit entirely on one line, no additional information, no headers, no anything else. Simple. And it returned



```
<HEADER>
<TITLE>The World Wide Web project</TITLE>
<NEXTID N="55">
</HEADER>
<BODY>
<H1>World Wide Web</H1>The WorldWideWeb (W3) is a wide-area<A
NAME=0 HREF="WhatIs.html">
hypermedia</A> information retrieval
initiative aiming to give universal
access to a large universe of documents.<P>
Everything there is online about
W3 is linked directly or indirectly
to this document, including an <A
```

simple HTML. this is the beginning of the response that the server would send out - notice, there’s no surrounding HTML tag, no protocol headers or metadata… just a simple response with a simple markup language.

And it looked, something like this



Home

Access to this information is provided as part of the WorldWideWeb project. The WWW project does not take responsibility for the accuracy of information provided by others.

How to proceed

References to other information are represented like this. Double-click on it to jump to related information.

General CERN information sources

Now choose an area in which you would like to start browsing. The system currently has access to three sources of information. With the indexes, you should use the keyword search option on your browser.

CERN Information — A general keyword index of information made available by the computer centre, including CERN, Cray and IBM help files, "Writeups", and the Computer Newsletter (CNL). (This is the same data as CERNVM which is also available on CERNVM with the VM FIND command).

Yellow Pages — A keyword index to the CERN telephone book by function.

You can access the internet news scheme (See information for new users). News articles are distributed typically CERN-wide or worldwide, and have a finite lifetime.

...may be of general interest at CERN include

...ws

...e Technology Interest Group news.

...machine, see also the following topics:

on this WorldWideWeb application

Info

HyperMedia Browser/Editor

Version 1.0 Alpha only

An exercise in global information availability

by Tim Berners-Lee

...1990, 91 CERN Distribution authorised for terms TESEVERS ON CRL...

...d: Text which is not constrained to be linear.
...edia: Information which is not constrained linear... or to be text.

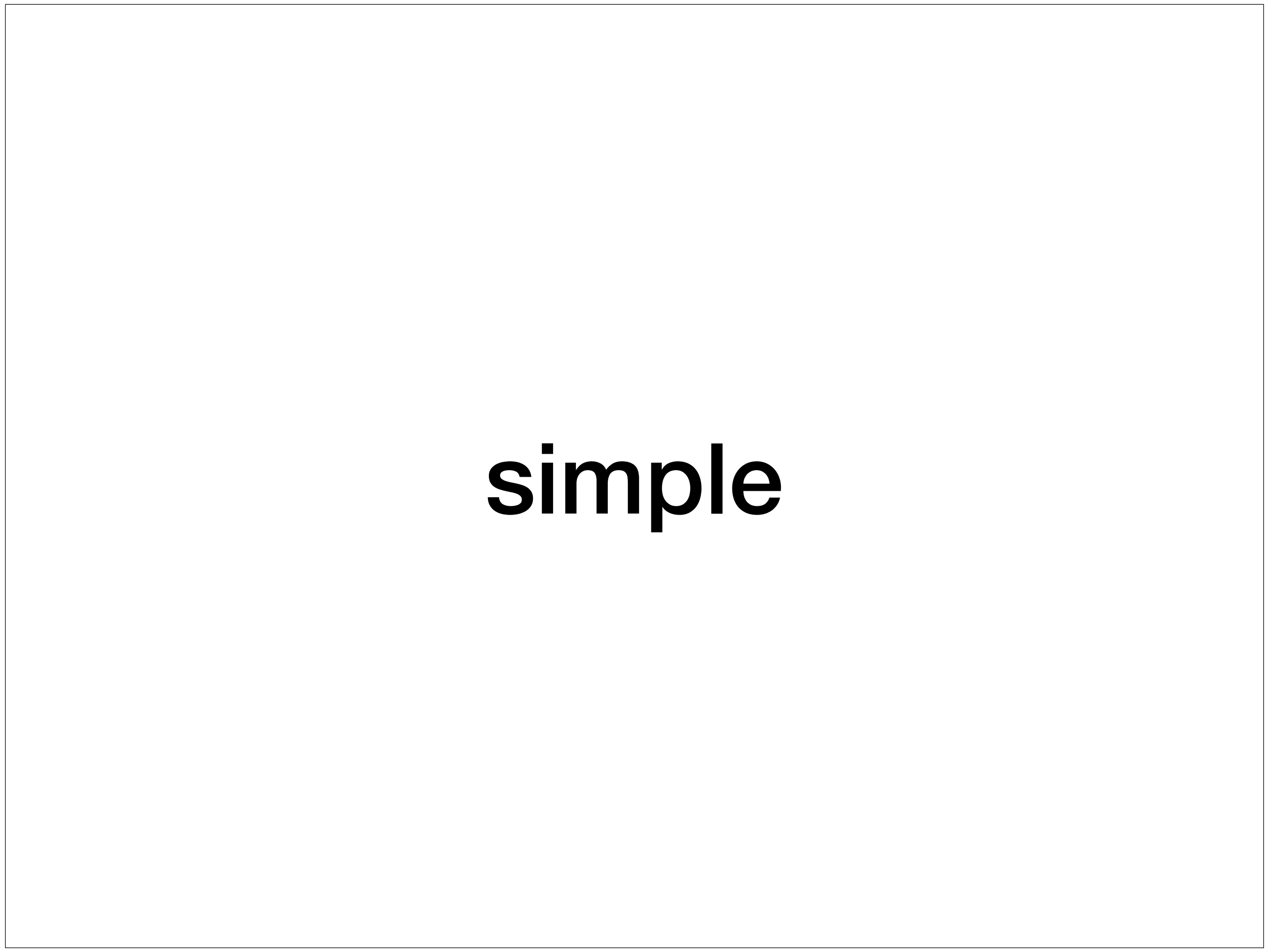
...is the first version of the NextStep WorldWideWeb application
...h the libWWW library. Bug reports to www-bug@info.cern.ch.
...See the list of known bugs in the web too.

...is was the original prototype for the WorldWide Web. Many
...owsers for other platforms now exist. Read the web for details.

...u should configure the newsreader code in this application to know
...bat your local news (NNTP) server is. Type in a terminal window

This is the original web browser, written at CERN for the NeXT operating system in 1990. Notice that there are no images in the text; at this point, the img tag hasn’t even been invented yet. There’s also not a lot of styling or layout. If you wanted to view an image, you clicked a link. And when you clicked on a link, text or image, it opened in a new window. the concept of a single frame with an ongoing, continuous history had yet to be introduced.

Like I said, pretty



simple

simple.

But the web of today is much more

complex. We no longer have one-line requests that return a few kilobytes of unencrypted HTML; we have HTTP/2 pipelined requests with cookies and user agent headers and SSL security. Those pages are generated by entire frameworks dedicated to running web applications, running behind front end servers that communicate with databases and a complex ecosystem of HTTP-based APIs. And what we return to the browser contains its own complex ecosystem. It's not just HTML on its own; it's a

trio of technologies, each with its own syntax, each with its own ecosystem of libraries, frameworks, tools, and even entire languages that compile into them. Cascading style sheets, with hundreds of rules on the average site, often organized into frameworks themselves, and Javascript, a lingua franca unto its own, with thousands of libraries and frameworks combined in exponentially complicated ways. Indeed, there are so many layers, we toss around the term

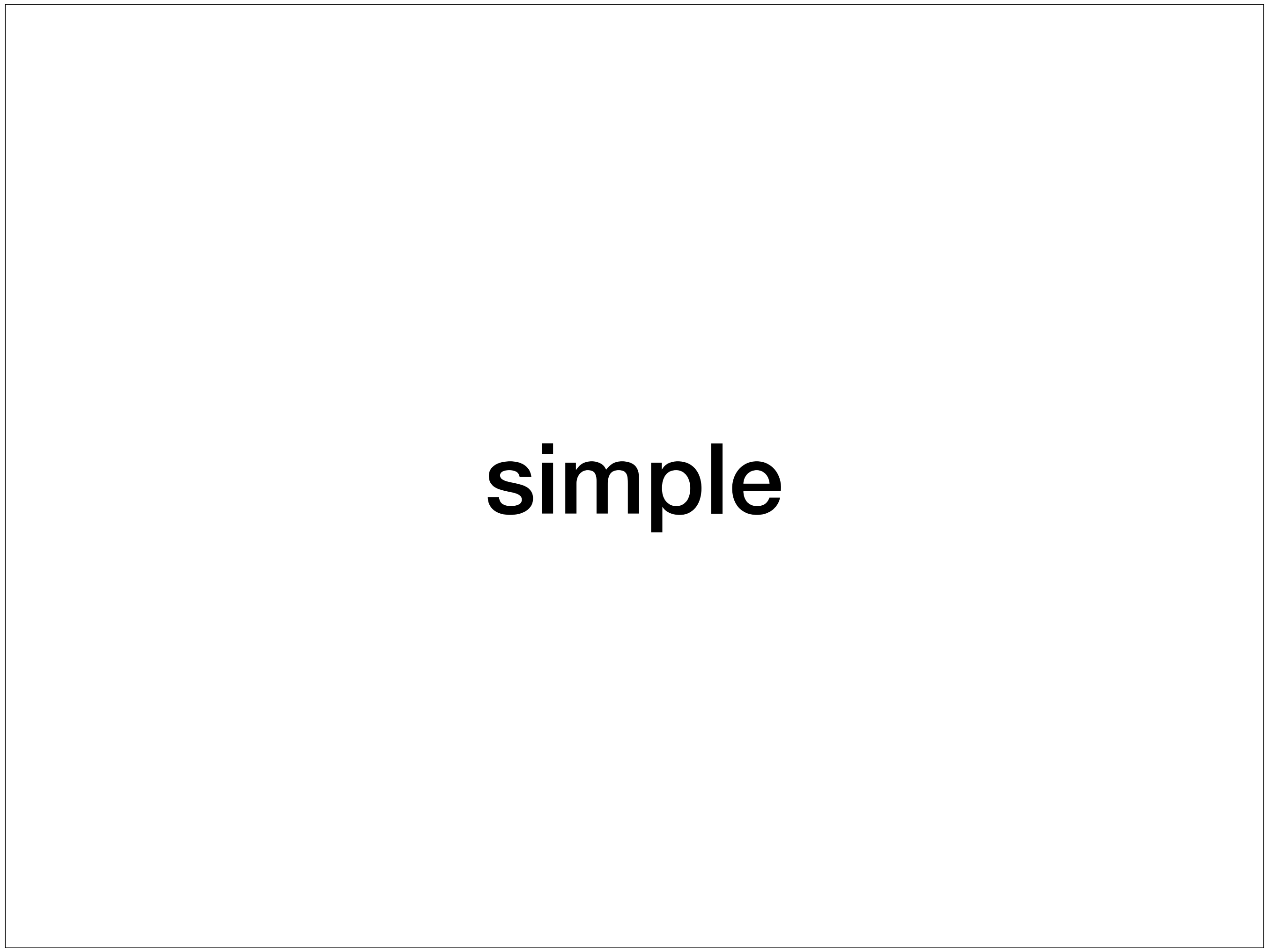
stack to describe the combinations. I prefer Ruby on Rails + Ubuntu + Nginx + MySQL, you might prefer Node.js + Express + Postgres + React. A lot of us probably identify as a full-stack developer, whatever that means, or at least have applied for a job requested one.

(Sidenote: have you ever seen a job posting for a half-stack developer?)

So

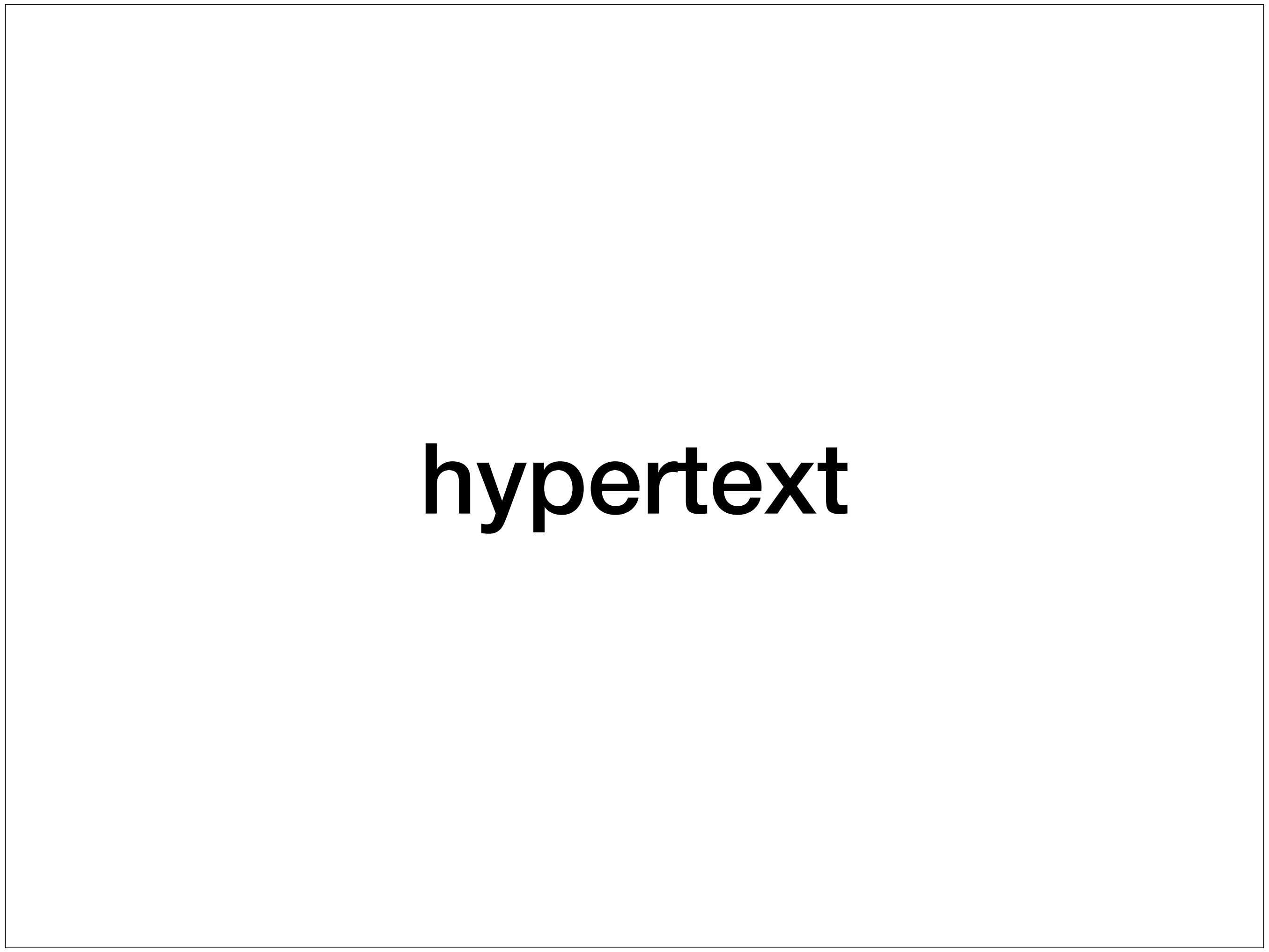
how’d we get here?

how’d we get here? 28 years ago, all of this started simply. What happened in the intervening years that built all of this up? And is there anything we can take away when we look at what’s happened over the close to 30 years of evolution? What I’d like to do with this talk is to look at the interesting ways that one particular tool in this toolset evolved, in ways that touch on everything in our stacks and the web.



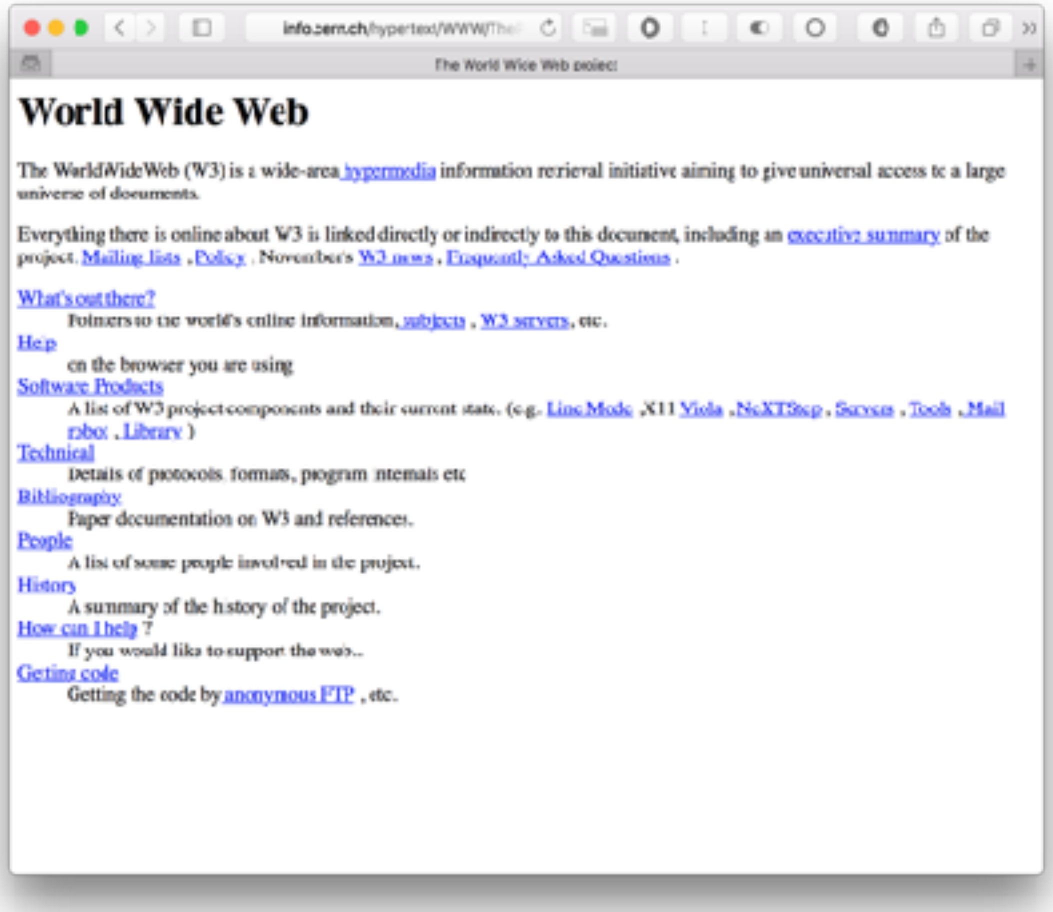
simple

simple. let’s go back to simple. as originally envisioned, the web was mostly about



hypertext

hypertext, the idea that, rather than linear books with at most some footnotes or an index or a table of contents that point to other pages; we could have entire interactive texts, extendable by any, linking between wildly different interrelated texts as easy as the click of a mouse. There were other, earlier implementations of this, with the original concept stretching all the way back to the 40s, and implementations demoed in the 60s.



World Wide Web

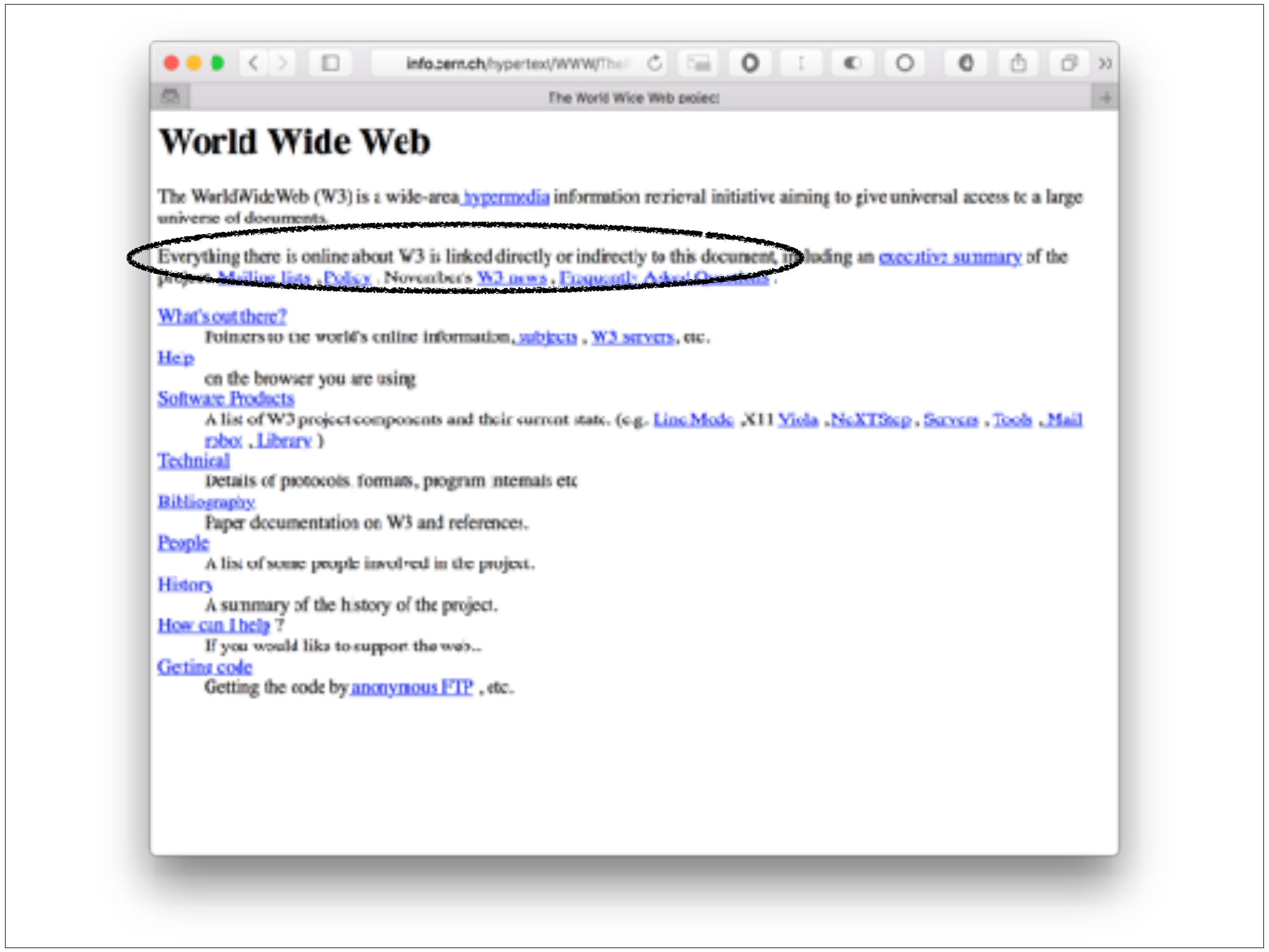
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
    Pointers to the world's online information, subjects , W3 servers, etc.
Help
    on the browser you are using
Software Products
    A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
    Details of protocols, formats, program internals etc
Bibliography
    Paper documentation on W3 and references.
People
    A list of some people involved in the project.
History
    A summary of the history of the project.
How can I help ?
    If you would like to support the web..
Getting code
    Getting the code by anonymous FTP , etc.

The World Wide Web applied the concept of Hypertext to the burgeoning Internet, and since the web started out mostly as a textual editing system, there wasn't an initial need for a lot of what came after.

(sidenote - this is what is most commonly known as the first web page. what I especially like about looking at this in 2018 is this note <build>… everything there is online about the World Wide Web is linked directly or indirectly to this document! still accurate, I suppose…)



World Wide Web

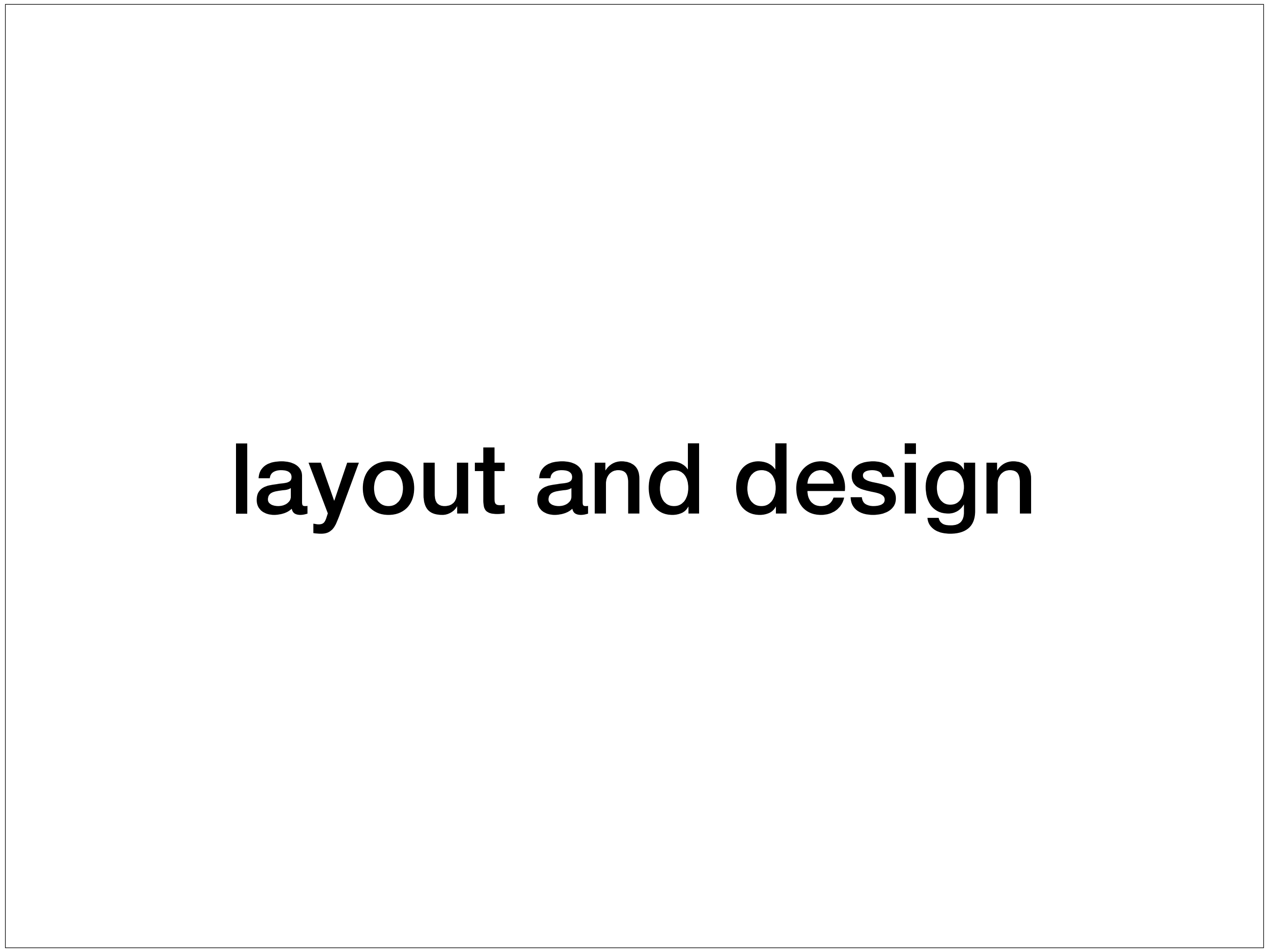
The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

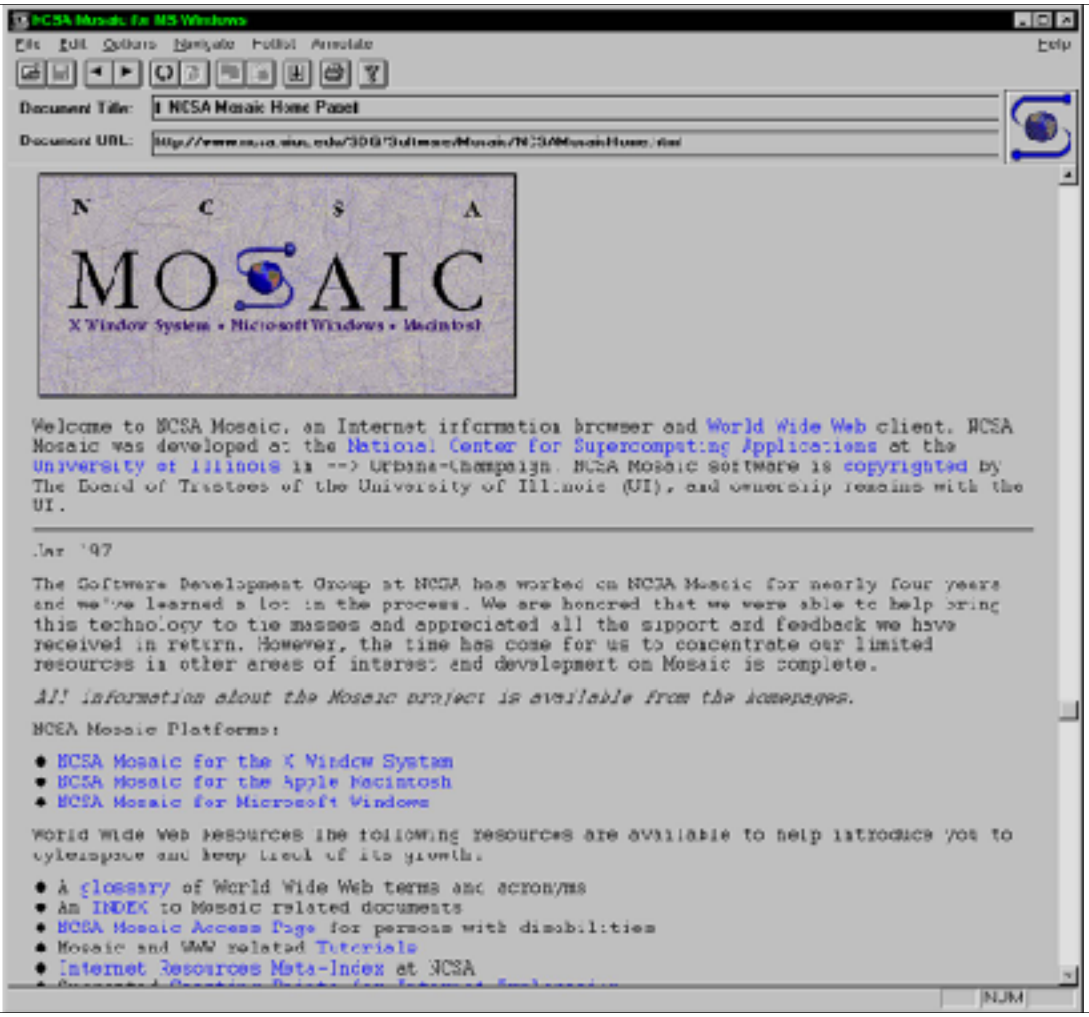
The World Wide Web applied the concept of Hypertext to the burgeoning Internet, and since the web started out mostly as a textual editing system, there wasn't an initial need for a lot of what came after.

(sidenote - this is what is most commonly known as the first web page. what I especially like about looking at this in 2018 is this note <build>… everything there is online about the World Wide Web is linked directly or indirectly to this document! still accurate, I suppose…)



layout and design

Proposals to control design were, at first, ignored or even laughed at, and most tags were specifically geared towards semantic meaning - headers, lists, paragraphs, images, and so on. That all started to change with the introduction of


NCSA Mosaic for MS Windows
File Edit Options Navigate Hotlist Annotate Help
Document Title: NCSA Mosaic Home Page!
Document URL:
N C S A
MOSAIC
X Window System • Microsoft Windows • Macintosh
NUM

Welcome to NCSA Mosaic, an Internet information browser and World Wide Web client. NCSA Mosaic was developed at the National Center for Supercomputing Applications at the University of Illinois in --> Urbana-Champaign. NCSA Mosaic software is copyrighted by The Board of Trustees of the University of Illinois (UI), and ownership remains with the UI.

Jan '97

The Software Development Group at NCSA has worked on NCSA Mosaic for nearly four years and we've learned a lot in the process. We are honored that we were able to help bring this technology to the masses and appreciated all the support and feedback we have received in return. However, the time has come for us to concentrate our limited resources in other areas of interest and development on Mosaic is complete.

All information about the Mosaic project is available from the homepages.

NCSA Mosaic Platforms:

- NCSA Mosaic for the X Window System
- NCSA Mosaic for the Apple Macintosh
- NCSA Mosaic for Microsoft Windows

World Wide Web Resources The following resources are available to help introduce you to cyberspace and keep track of its growth.

- A glossary of World Wide Web terms and acronyms
- An INDEX to Mosaic related documents
- NCSA Mosaic Access Page for persons with disabilities
- Mosaic and WWW related Tutorials
- Internet Resources Meta-Index at NCSA

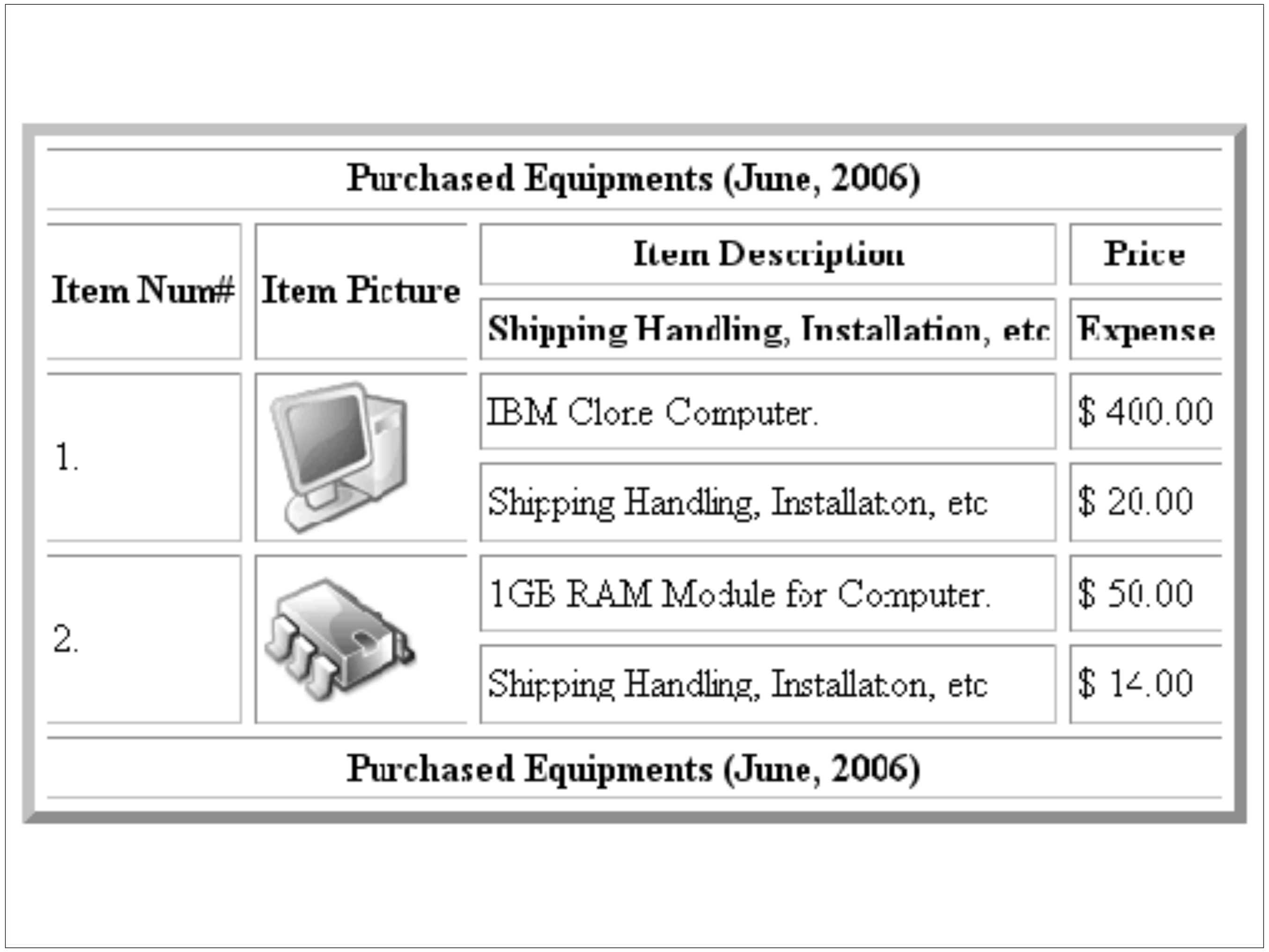
NCSA Mosaic, released in 1993. Its successor (Netscape Navigator) was what really set the world on fire, but Mosaic is widely considered to be the browser that paved the way for the Web to be synonymous with the Internet, and is what started the web on its path away from just another experimental hypertext medium and into what we know it as today. Unlike the original CERN browser, Mosaic didn't allow editing, and it also contained the first implementation of the IMG tag, as you can see with that gorgeous inline image at the top. and it started to let people view the web as more of a visual medium.

(Fun sidenote - at its peak, Mosaic was downloaded more than 50,000 times a month. In 1994.)

Still, pages still tended to be flat, textual.

tables

Things changed when tables made it into HTML, introduced a year or two earlier but shipping in Mosaic and other browsers in 1995. They were, of course, originally designed for tabular data

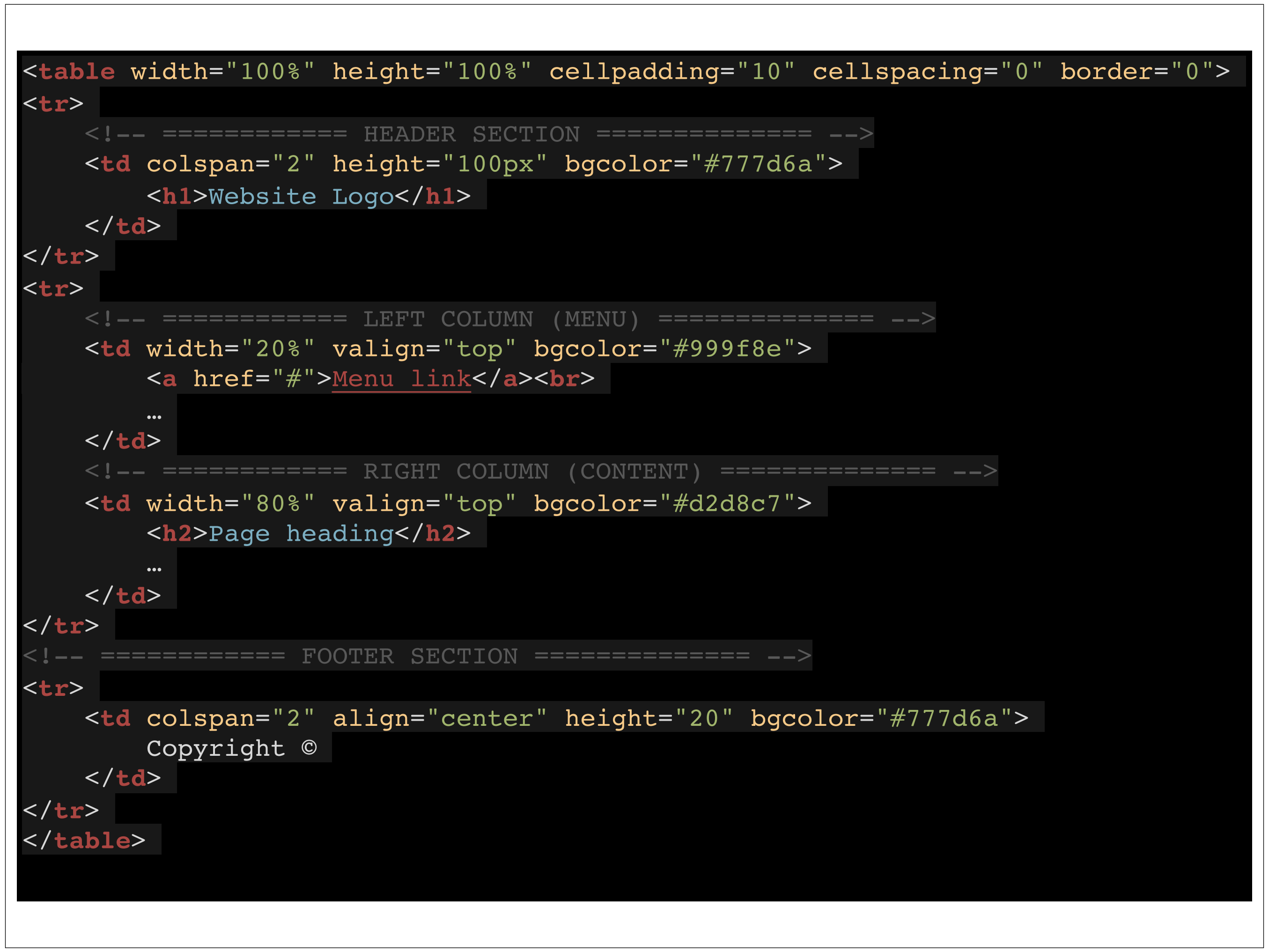


Purchased Equipments (June, 2006)

Item Num#	Item Picture	Item Description	Price
		Shipping Handling, Installation, etc	Expense
1.		IBM Clone Computer.	$ 400.00
		Shipping Handling, Installation, etc	$ 20.00
2.		1GB RAM Module for Computer.	$ 50.00
		Shipping Handling, Installation, etc	$ 14.00

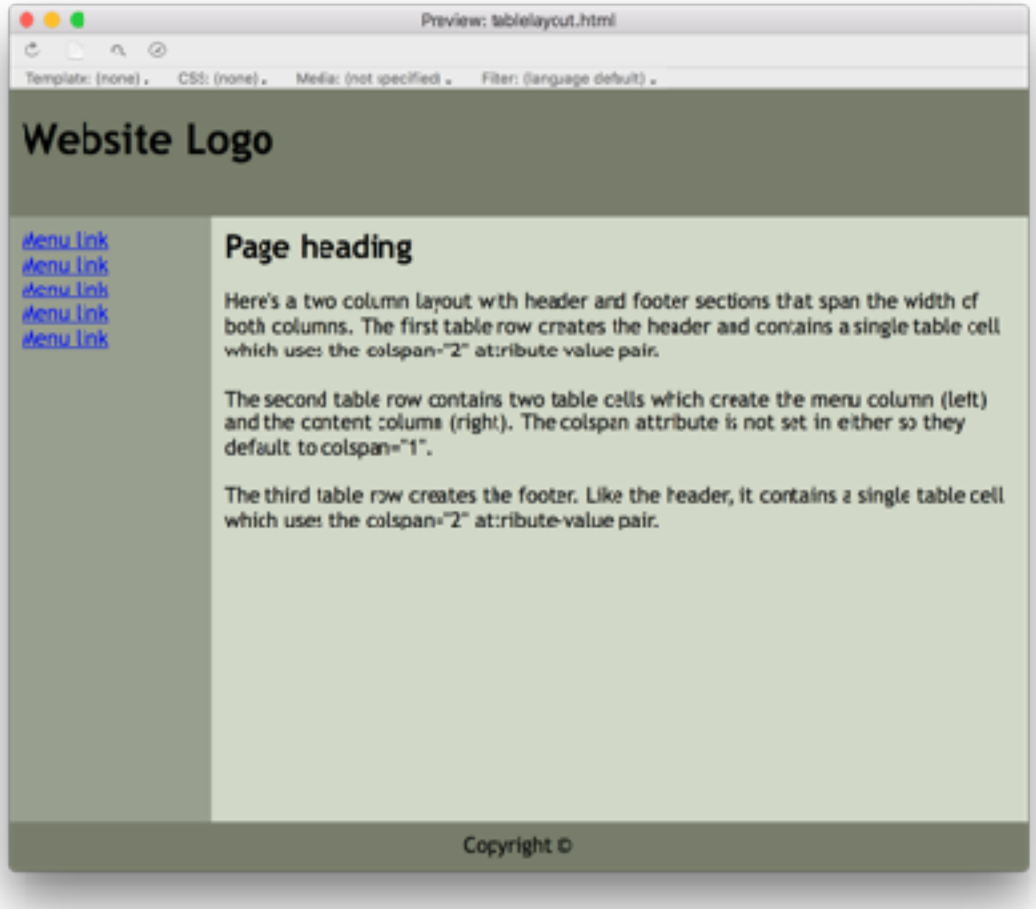
Purchased Equipments (June, 2006)

and (in most browsers) bringing along a charmingly old-school 3D look. It wasn't long, though, before people started to realize that maybe they could be turned to another purpose, and this led to the rise of



```
<table width="100%" height="100%" cellpadding="10" cellspacing="0" border="0">
<tr>
    <!-- ============ HEADER SECTION ============== -->
    <td colspan="2" height="100px" bgcolor="#777d6a">
        <h1>Website Logo</h1>
    </td>
</tr>
<tr>
    <!-- ============ LEFT COLUMN (MENU) ============== -->
    <td width="20%" valign="top" bgcolor="#999f8e">
        <a href="#">Menu link</a><br>
        …
    </td>
    <!-- ============ RIGHT COLUMN (CONTENT) ============== -->
    <td width="80%" valign="top" bgcolor="#d2d8c7">
        <h2>Page heading</h2>
        …
    </td>
</tr>
<!-- ============ FOOTER SECTION ============== -->
<tr>
    <td colspan="2" align="center" height="20" bgcolor="#777d6a">
        Copyright ©
    </td>
</tr>
</table>
```

the table based layout. By laying out a series of multi-cell rows and columns, it was fairly easy to manipulate a browser into showing things that looked like headers, footers and sidebars. Here we have a standard-for-the-time two column layout, along with a header and footer. When rendered, it looks



Website Logo

Page heading

Here's a two column layout with header and footer sections that span the width of both columns. The first table row creates the header and contains a single table cell which uses the colspan="2" attribute value pair.

The second table row contains two table cells which create the menu column (left) and the content column (right). The colspan attribute is not set in either so they default to colspan="1".

The third table row creates the footer. Like the header, it contains a single table cell which uses the colspan="2" attribute-value pair.

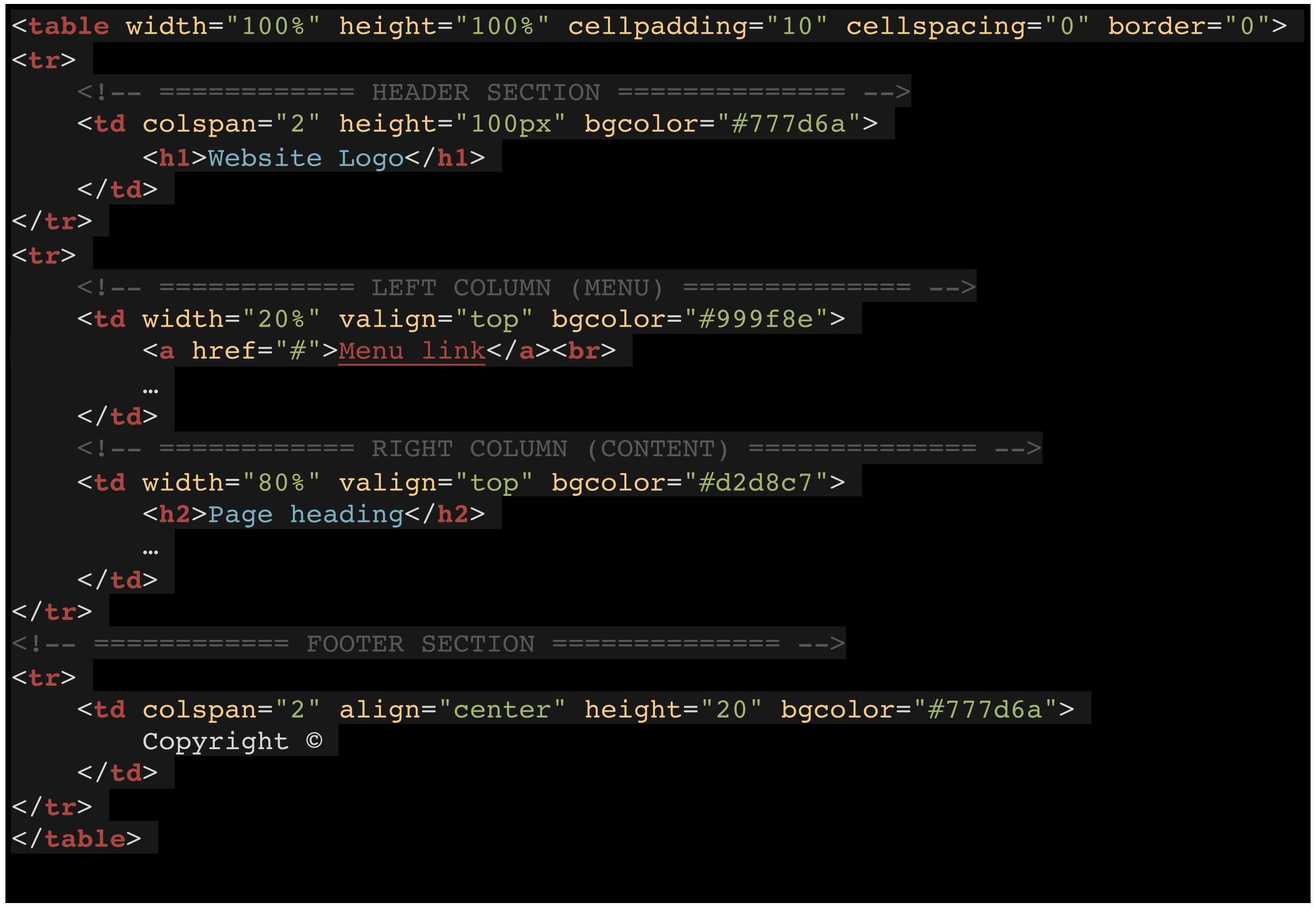
Copyright ©

something like this.

And thus was born layout on the Web. There were other attempts at this of course - frames being one of the more notable, but table layouts were the norm for a shockingly long period of time.

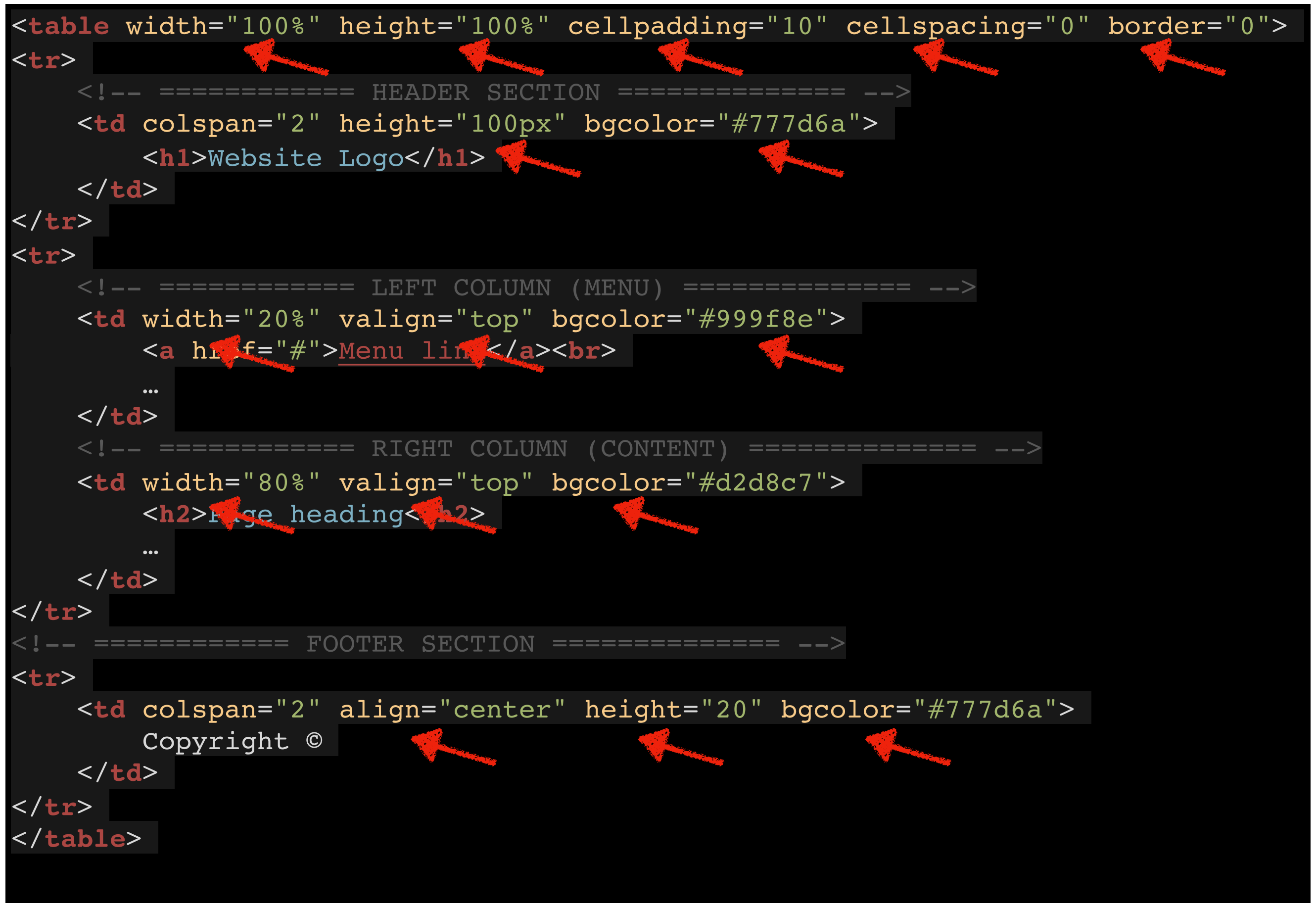
Fun fact - how many folks here have done email templates? if you’ve ever had to do email templates, this persisted waaaay into the nows.

Let’s go back to the source code for a minute.



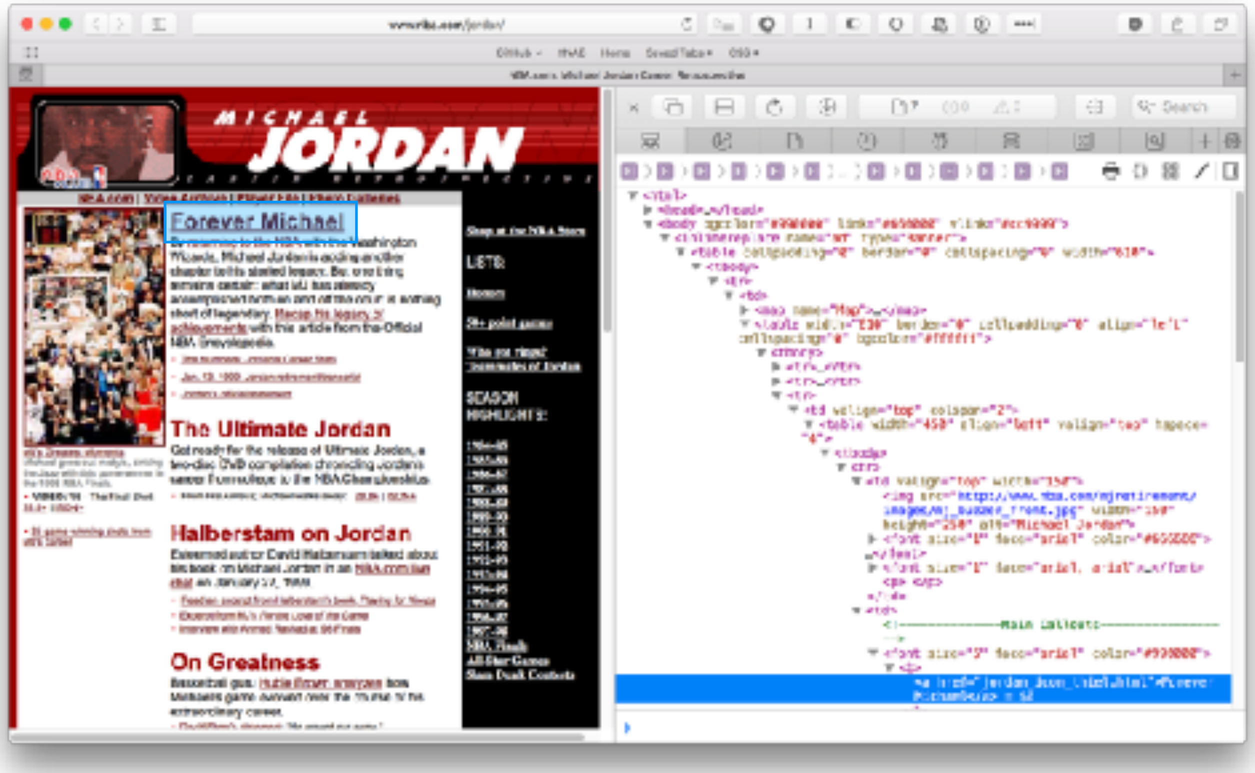
```html
<table width="100%" height="100%" cellpadding="10" cellspacing="0" border="0">
<tr>
    <!-- ============ HEADER SECTION ============== -->
    <td colspan="2" height="100px" bgcolor="#777d6a">
        <h1>Website Logo</h1>
    </td>
</tr>
<tr>
    <!-- ============ LEFT COLUMN (MENU) ============== -->
    <td width="20%" valign="top" bgcolor="#999f8e">
        <a href="#">Menu link</a><br>
        …
    </td>
    <!-- ============ RIGHT COLUMN (CONTENT) ============== -->
    <td width="80%" valign="top" bgcolor="#d2d8c7">
        <h2>Page heading</h2>
        …
    </td>
</tr>
<!-- ============ FOOTER SECTION ============== -->
<tr>
    <td colspan="2" align="center" height="20" bgcolor="#777d6a">
        Copyright ©
    </td>
</tr>
</table>
```

What do you notice about the markup laid out here? Besides the creative use of table tags that we just talked about, what else is odd compared to how you'd do things today?



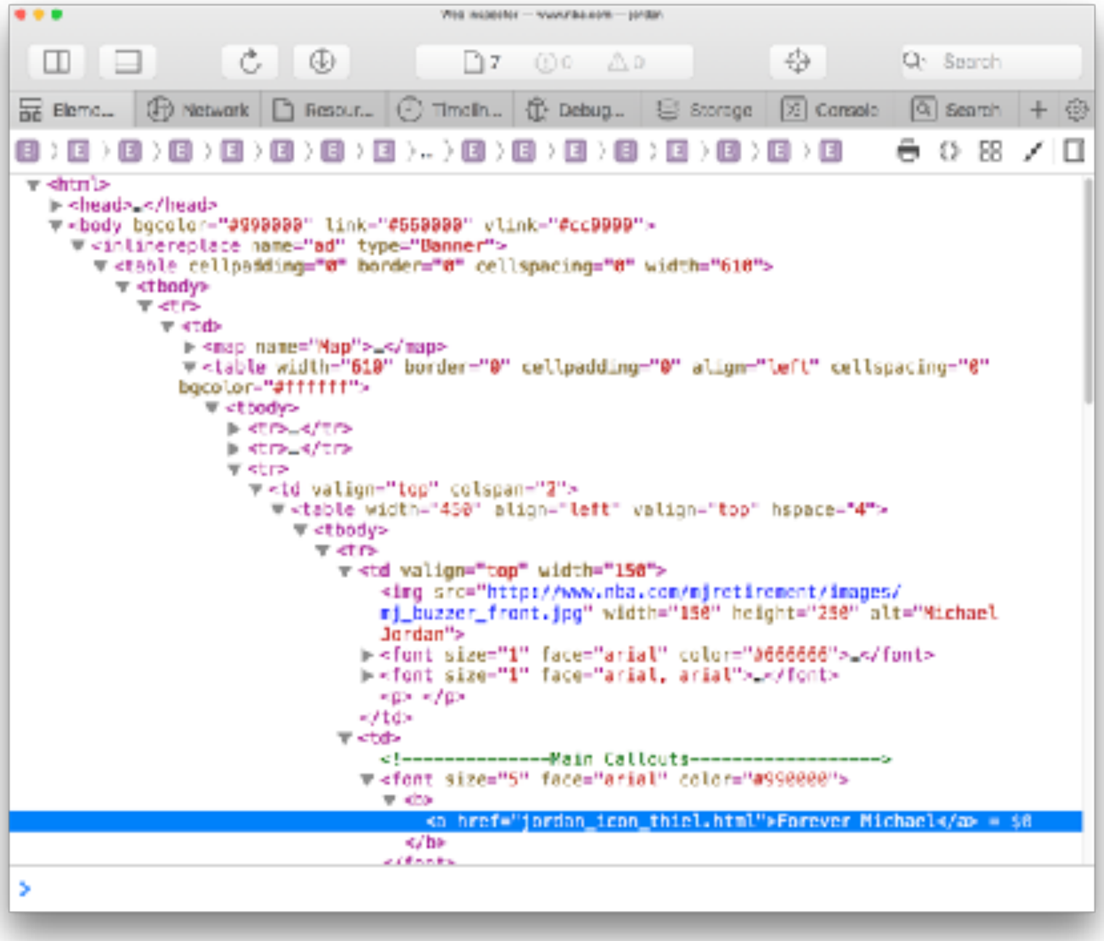
```
<table width="100%" height="100%" cellpadding="10" cellspacing="0" border="0">
<tr>
    <!-- ============ HEADER SECTION ============== -->
    <td colspan="2" height="100px" bgcolor="#777d6a">
        <h1>Website Logo</h1>
    </td>
</tr>
<tr>
    <!-- ============ LEFT COLUMN (MENU) ============== -->
    <td width="20%" valign="top" bgcolor="#999f8e">
        <a href="#">Menu link</a><br>
        …
    </td>
    <!-- ============ RIGHT COLUMN (CONTENT) ============== -->
    <td width="80%" valign="top" bgcolor="#d2d8c7">
        <h2>Page heading</h2>
        …
    </td>
</tr>
<!-- ============ FOOTER SECTION ============== -->
<tr>
    <td colspan="2" align="center" height="20" bgcolor="#777d6a">
        Copyright ©
    </td>
</tr>
</table>
```

It’s all the styling attributes in the markup. The only way to control things like centering, height, background colors was in the HTML itself. This wasn’t just attributes like alignment, heights, paddings or colors, but fonts, centering, and other styling choices were expressed through tags and attributes as well.



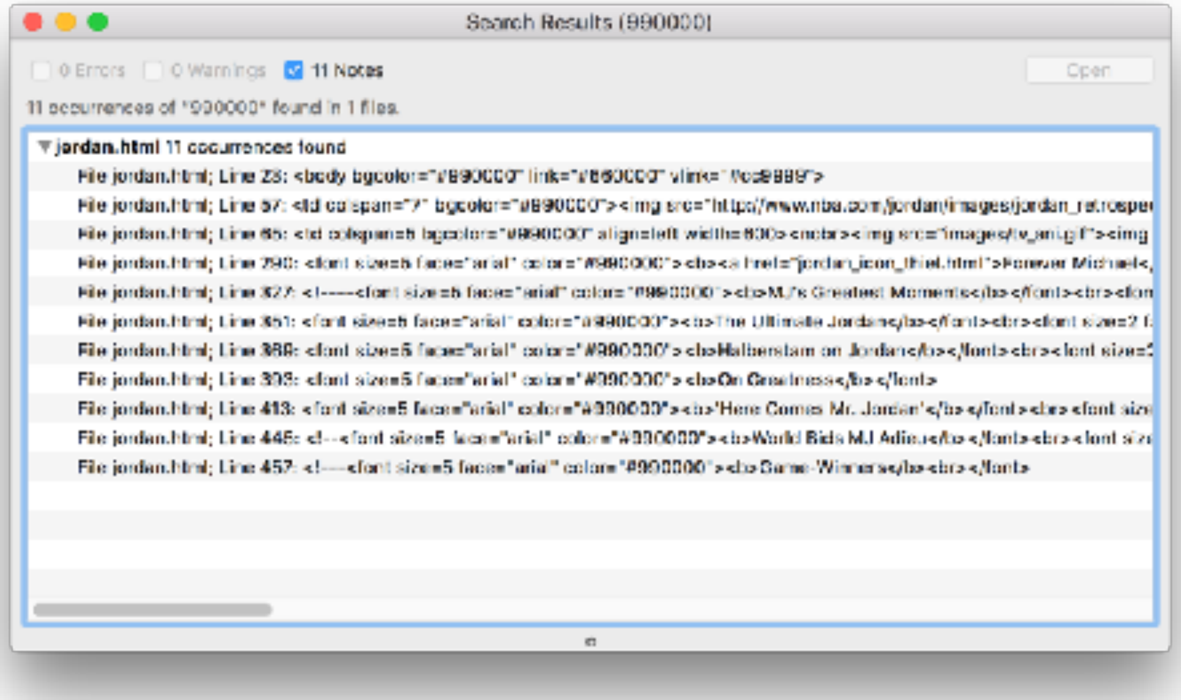
```
▼ <html>
  ▶ <head>…</head>
  ▼ <body bgcolor="#990000" link="#650000" vlink="#cc9999">
    ▼ <[illegible] name="MT" type="banner">
      ▼ <table cellpadding="0" border="0" cellspacing="0" width="610">
        ▼ <tbody>
          ▼ <tr>
            ▼ <td>
              ▶ <map name="Map">…</map>
              ▼ <table width="630" border="0" cellpadding="0" align="left"
                cellspacing="0" bgcolor="#ffffff">
                ▼ <tbody>
                  ▶ <tr>…</tr>
                  ▶ <tr>…</tr>
                  ▼ <tr>
                    ▼ <td valign="top" colspan="2">
                      ▼ <table width="450" align="left" valign="top" hspace=
                        "4">
                        ▼ <tbody>
                          ▼ <tr>
                            ▼ <td valign="top" width="150">
                                <img src="http://www.nba.com/mjretirement/
                                images/mj_banner_front.jpg" width="150"
                                height="250" alt="Michael Jordan">
                              ▶ <font size="1" face="arial" color="#650000">
                              …</font>
                              ▶ <font size="1" face="arial, arial">…</font>
                                <p> </p>
                              </td>
                            ▼ <td>
                                <!--------------Main Content----------------
                                -->
                              ▼ <font size="3" face="arial" color="#990000">
                                ▼ <b>
                                    <a href="jordan_[illegible].html">Forever
                                    Michael</a> = $0
```

While this let people display layouts they couldn't otherwise have dreamed of building in flat HTML, this had a number of caveats. For instance, take a look at this markup from a still-on-the-internet NBA retrospective of Michael Jordan. Notice how deeply nested the the primary article link on this page is - it's buried within three separate tables to even get it to that point. One of them sets up the header and main areas, one of them the right sidebar vs main content area, and the final one the right sidebar and the main main content area. Granted, it gets the job done, but it's not tabular data, and it certainly stretches the definition of what a table should be far past the breaking point.



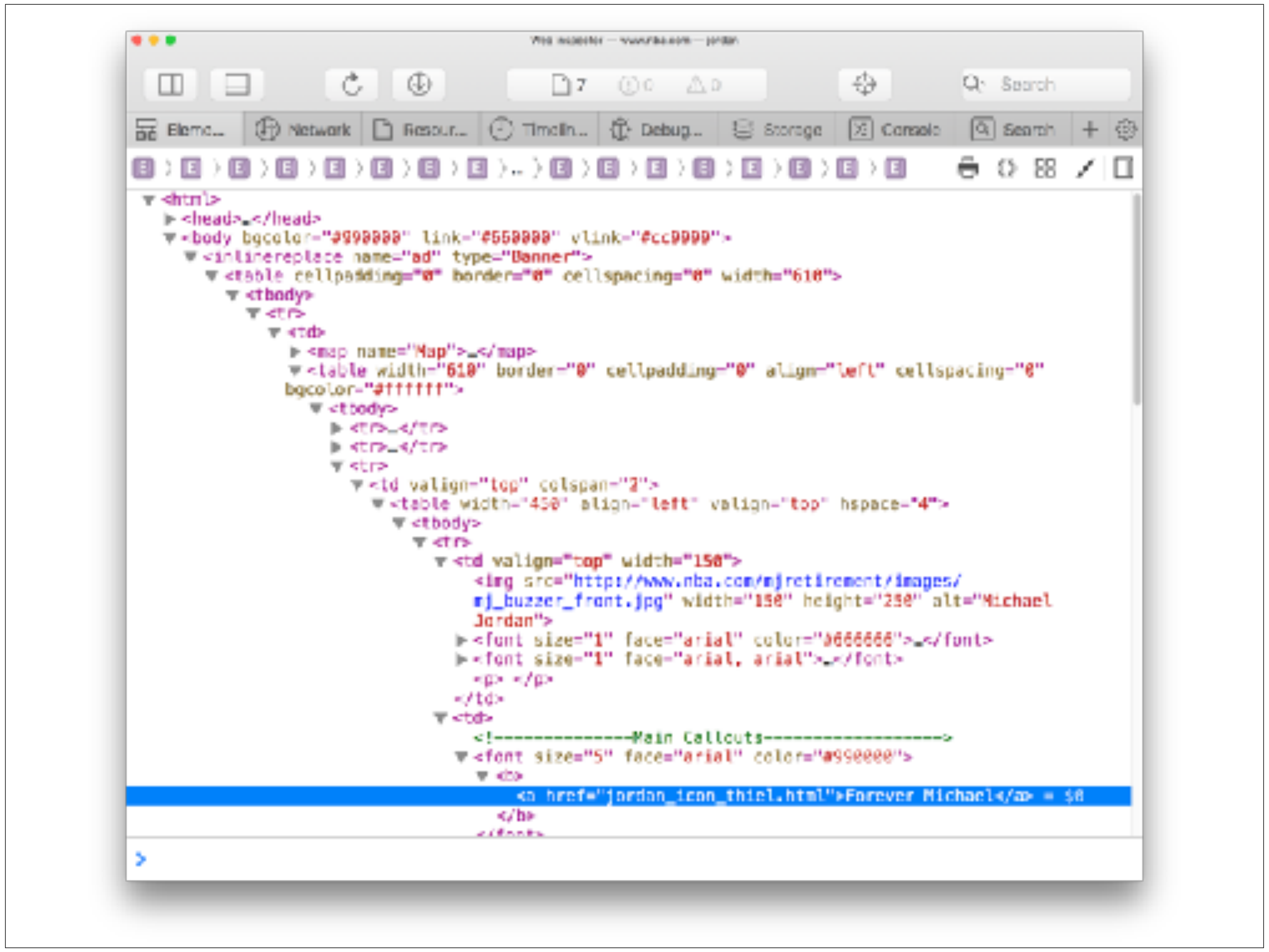
```
▼ <html>
  ▶ <head>…</head>
  ▼ <body bgcolor="#990000" link="#550000" vlink="#cc9999">
    ▼ <inlinereplace name="ad" type="Banner">
      ▼ <table cellpadding="0" border="0" cellspacing="0" width="610">
        ▼ <tbody>
          ▼ <tr>
            ▼ <td>
              ▶ <map name="Map">…</map>
              ▼ <table width="610" border="0" cellpadding="0" align="left" cellspacing="0"
              bgcolor="#ffffff">
                ▼ <tbody>
                  ▶ <tr>…</tr>
                  ▶ <tr>…</tr>
                  ▼ <tr>
                    ▼ <td valign="top" colspan="2">
                      ▼ <table width="450" align="left" valign="top" hspace="4">
                        ▼ <tbody>
                          ▼ <tr>
                            ▼ <td valign="top" width="150">
                                <img src="http://www.nba.com/mjretirement/images/
                                mj_buzzer_front.jpg" width="150" height="250" alt="Michael
                                Jordan">
                              ▶ <font size="1" face="arial" color="#666666">…</font>
                              ▶ <font size="1" face="arial, arial">…</font>
                                <p> </p>
                              </td>
                            ▼ <td>
                                <!--------------Main Callouts------------------>
                              ▼ <font size="5" face="arial" color="#990000">
                                ▼ <b>
                                    <a href="jordan_icon_thiel.html">Forever Michael</a> = $0
                                  </b>
                                </font>
```

And the problems don't end there. Just in what is displayed here, we can spot a lot of duplication… the background red is also used as the main color for links inside the content, but it's specified every time it's needed; that color



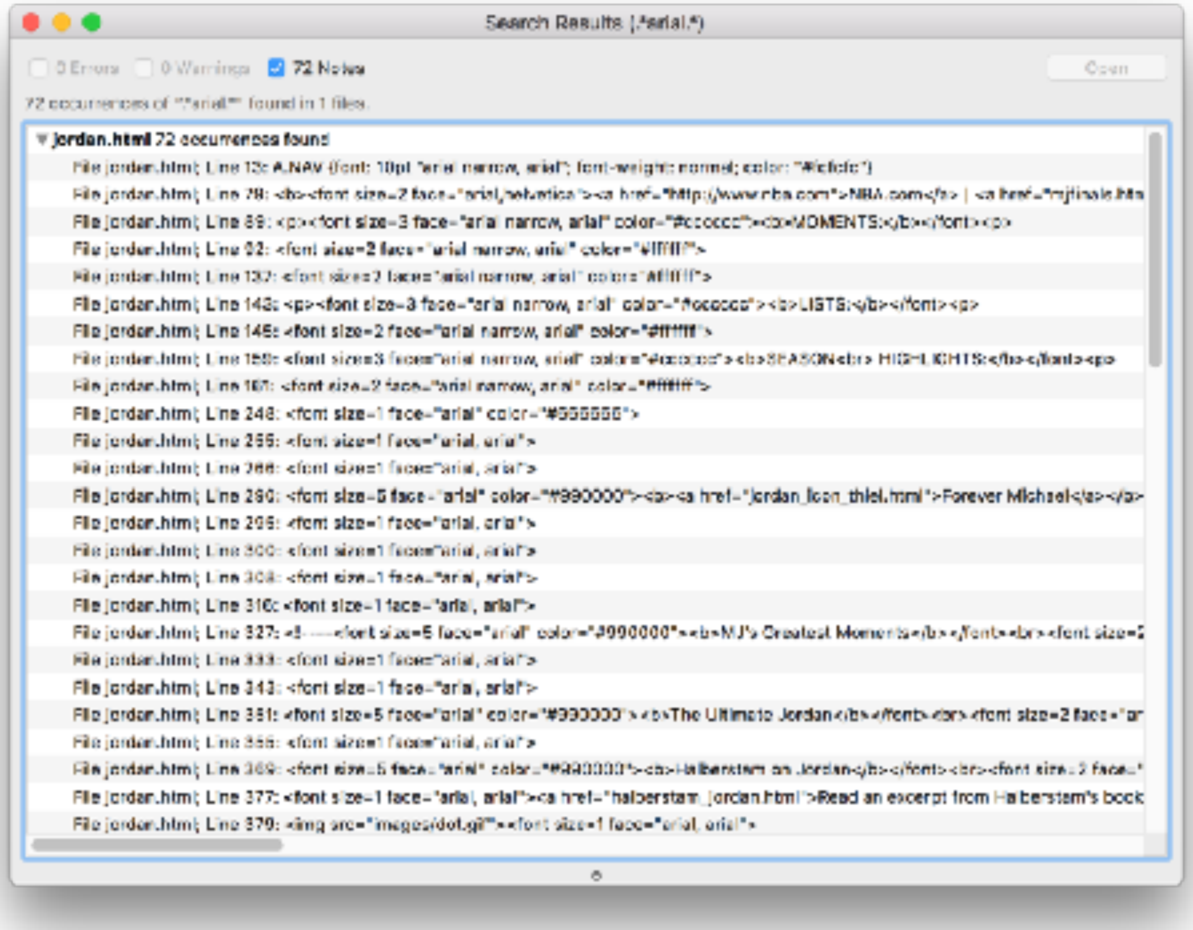
```
11 occurrences of "990000" found in 1 files.
▼ jordan.html 11 occurrences found
    File jordan.html; Line 23: <body bgcolor="#990000" link="#990000" vlink="#cc9999">
    File jordan.html; Line 57: <td colspan="7" bgcolor="#990000"><img src="http://www.nba.com/jordan/images/jordan_retrospe
    File jordan.html; Line 65: <td colspan=5 bgcolor="#990000" align=left width=600><nobr><img src="images/tv_ani.gif"><img
    File jordan.html; Line 290: <font size=5 face="arial" color="#990000"><b><a href="jordan_icon_thiel.html">Forever Michael<,
    File jordan.html; Line 327: <!-----<font size=5 face="arial" color="#990000"><b>MJ's Greatest Moments</b></font><br><font
    File jordan.html; Line 351: <font size=5 face="arial" color="#990000"><b>The Ultimate Jordan</b></font><br><font size=2 f
    File jordan.html; Line 388: <font size=5 face="arial" color="#990000"><b>Halberstam on Jordan</b></font><br><font size=2
    File jordan.html; Line 393: <font size=5 face="arial" color="#990000"><b>On Greatness</b></font>
    File jordan.html; Line 413: <font size=5 face="arial" color="#990000"><b>'Here Comes Mr. Jordan'</b></font><br><font size
    File jordan.html; Line 445: <!--<font size=5 face="arial" color="#990000"><b>World Bids MJ Adieu</b></font><br><font siz
    File jordan.html; Line 457: <!----<font size=5 face="arial" color="#990000"><b>Game-Winners</b><br></font>
```

is specified in eleven different places, eight of which are for those same major callout links. Furthermore, each one is also set to font size 5 - in these days, font sizes were relative to whatever you had set as your browser's default font size - and each one of them is also bolded. seems like there's a lot of duplication here… but it gets worse.



```
▼ <html>
  ▶ <head>…</head>
  ▼ <body bgcolor="#990000" link="#550000" vlink="#cc9999">
    ▼ <inlinereplace name="ad" type="banner">
      ▼ <table cellpadding="0" border="0" cellspacing="0" width="610">
        ▼ <tbody>
          ▼ <tr>
            ▼ <td>
              ▶ <map name="Map">…</map>
              ▼ <table width="610" border="0" cellpadding="0" align="left" cellspacing="0"
             bgcolor="#ffffff">
                ▼ <tbody>
                  ▶ <tr>…</tr>
                  ▶ <tr>…</tr>
                  ▼ <tr>
                    ▼ <td valign="top" colspan="2">
                      ▼ <table width="450" align="left" valign="top" hspace="4">
                        ▼ <tbody>
                          ▼ <tr>
                            ▼ <td valign="top" width="150">
                                <img src="http://www.nba.com/mjretirement/images/
                                mj_buzzer_front.jpg" width="150" height="250" alt="Michael
                                Jordan">
                              ▶ <font size="1" face="arial" color="#666666">…</font>
                              ▶ <font size="1" face="arial, arial">…</font>
                                <p> </p>
                              </td>
                            ▼ <td>
                                <!-------------Main Callouts----------------->
                              ▼ <font size="5" face="arial" color="#990000">
                                ▼ <b>
                                    <a href="jordan_icon_thiel.html">Forever Michael</a> = $8
                                  </b>
                                </font>
```

Going back to our element tree real quick, one other thing you'll notice is the number of times Arial is mentioned. While here you only see it four times, that's still a lot, and that's mostly because we're so deep in the document element tree that we can't actually see most of the content. If we do a search ala the red we looked at earlier we'll find that Arial is mentioned



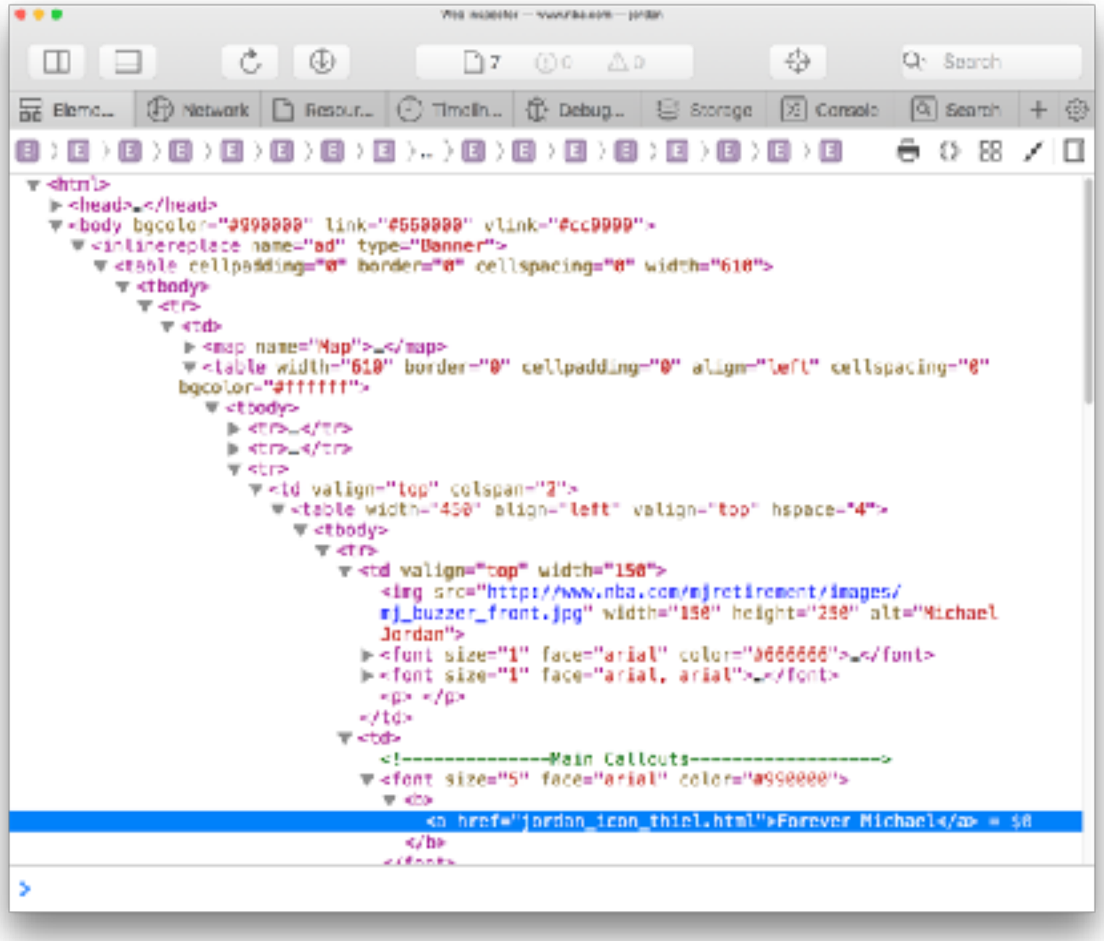
□ 0 Errors □ 0 Warnings ☑ 72 Notes

72 occurrences of ".*arial.*" found in 1 files.

```
▼ jordan.html 72 occurrences found
    File jordan.html; Line 12: A.NAV {font: 10pt "arial narrow, arial"; font-weight: normal; color: "#cfcfcf"}
    File jordan.html; Line 78: <b><font size=2 face="arial,helvetica"><a href="http://www.nba.com">NBA.com</a> | <a href="mjfinals.htm
    File jordan.html; Line 89: <p><font size=3 face="arial narrow, arial" color="#cccccc"><b>MOMENTS:</b></font><p>
    File jordan.html; Line 92: <font size=2 face="arial narrow, arial" color="#ffffff">
    File jordan.html; Line 132: <font size=2 face="arial narrow, arial" color="#ffffff">
    File jordan.html; Line 143: <p><font size=3 face="arial narrow, arial" color="#cccccc"><b>LISTS:</b></font><p>
    File jordan.html; Line 145: <font size=2 face="arial narrow, arial" color="#ffffff">
    File jordan.html; Line 158: <font size=3 face="arial narrow, arial" color="#cccccc"><b>SEASON<br> HIGHLIGHTS:</b></font><p>
    File jordan.html; Line 167: <font size=2 face="arial narrow, arial" color="#ffffff">
    File jordan.html; Line 248: <font size=1 face="arial" color="#999999">
    File jordan.html; Line 255: <font size=1 face="arial, arial">
    File jordan.html; Line 266: <font size=1 face="arial, arial">
    File jordan.html; Line 280: <font size=5 face="arial" color="#990000"><b><a href="jordan_leon_thiel.html">Forever Michael</a></b>
    File jordan.html; Line 295: <font size=1 face="arial, arial">
    File jordan.html; Line 300: <font size=1 face="arial, arial">
    File jordan.html; Line 308: <font size=1 face="arial, arial">
    File jordan.html; Line 316: <font size=1 face="arial, arial">
    File jordan.html; Line 327: <!-----<font size=5 face="arial" color="#990000"><b>MJ's Greatest Moments</b></font><br><font size=2
    File jordan.html; Line 333: <font size=1 face="arial, arial">
    File jordan.html; Line 343: <font size=1 face="arial, arial">
    File jordan.html; Line 351: <font size=5 face="arial" color="#990000"><b>The Ultimate Jordan</b></font><br><font size=2 face="ar
    File jordan.html; Line 355: <font size=1 face="arial, arial">
    File jordan.html; Line 369: <font size=5 face="arial" color="#990000"><b>Halberstam on Jordan</b></font><br><font size=2 face="
    File jordan.html; Line 377: <font size=1 face="arial, arial"><a href="halberstam_jordan.html">Read an excerpt from Halberstam's book
    File jordan.html; Line 379: <img src="images/dot.gif"><font size=1 face="arial, arial">
```

on 72 different lines! And this is just on one page; multiply this by the number of pages you have on the site. So if you want to change the font, or the background + link color, or any one of these things - you’re in for a massive find and replace job across your entire codebase.

(sidenote: my favorites are the ones that say arial, arial. it’s the font so nice they requested it twice :)



```
▼ <html>
  ▶ <head>…</head>
  ▼ <body bgcolor="#990000" link="#550000" vlink="#cc9999">
    ▼ <inlinereplace name="ad" type="Banner">
      ▼ <table cellpadding="0" border="0" cellspacing="0" width="610">
        ▼ <tbody>
          ▼ <tr>
            ▼ <td>
              ▶ <map name="Map">…</map>
              ▼ <table width="610" border="0" cellpadding="0" align="left" cellspacing="0"
              bgcolor="#ffffff">
                ▼ <tbody>
                  ▶ <tr>…</tr>
                  ▶ <tr>…</tr>
                  ▼ <tr>
                    ▼ <td valign="top" colspan="2">
                      ▼ <table width="450" align="left" valign="top" hspace="4">
                        ▼ <tbody>
                          ▼ <tr>
                            ▼ <td valign="top" width="150">
                                <img src="http://www.nba.com/mjretirement/images/
                                mj_buzzer_front.jpg" width="150" height="250" alt="Michael
                                Jordan">
                              ▶ <font size="1" face="arial" color="#666666">…</font>
                              ▶ <font size="1" face="arial, arial">…</font>
                                <p> </p>
                              </td>
                            ▼ <td>
                                <!--------------Main Callouts------------------->
                              ▼ <font size="5" face="arial" color="#990000">
                                ▼ <b>
                                    <a href="jordan_icon_thief.html">Forever Michael</a> = $0
                                  </b>
                                </font>
>
```

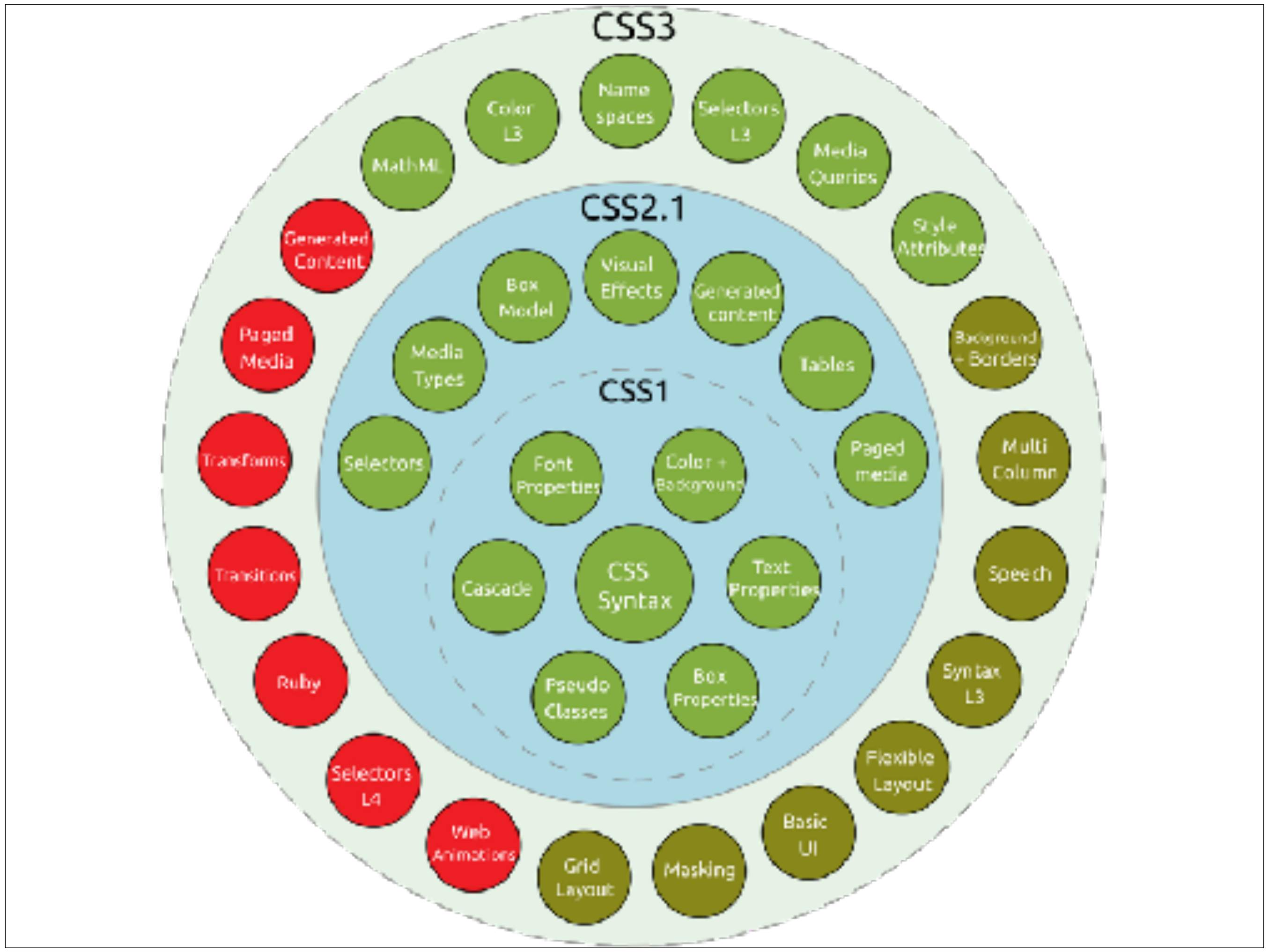
So. All of these tags. They're brittle, and hard to change. (go back a slide, actually my pet theory about "arial, arial" is that some of these cases were Arial Narrow, and they didn't bother to change it.) It also doesn't reveal much about the structure of our document; in fact, it hides it. Any meaning that can be derived from "this is a table" is completely lost when literally everything on a page is inside of a table. And finally, it's just inefficient. This isn't _that_ much by today's standards, but this page weighs in at 22kb; about a four-second download on a 56k modem, and that's not including all of the images it has to load.

This, I think, could be termed one of the first great hacks in web development. It's elegant in how easily it works, but as we've explored, has its downsides.

CSS

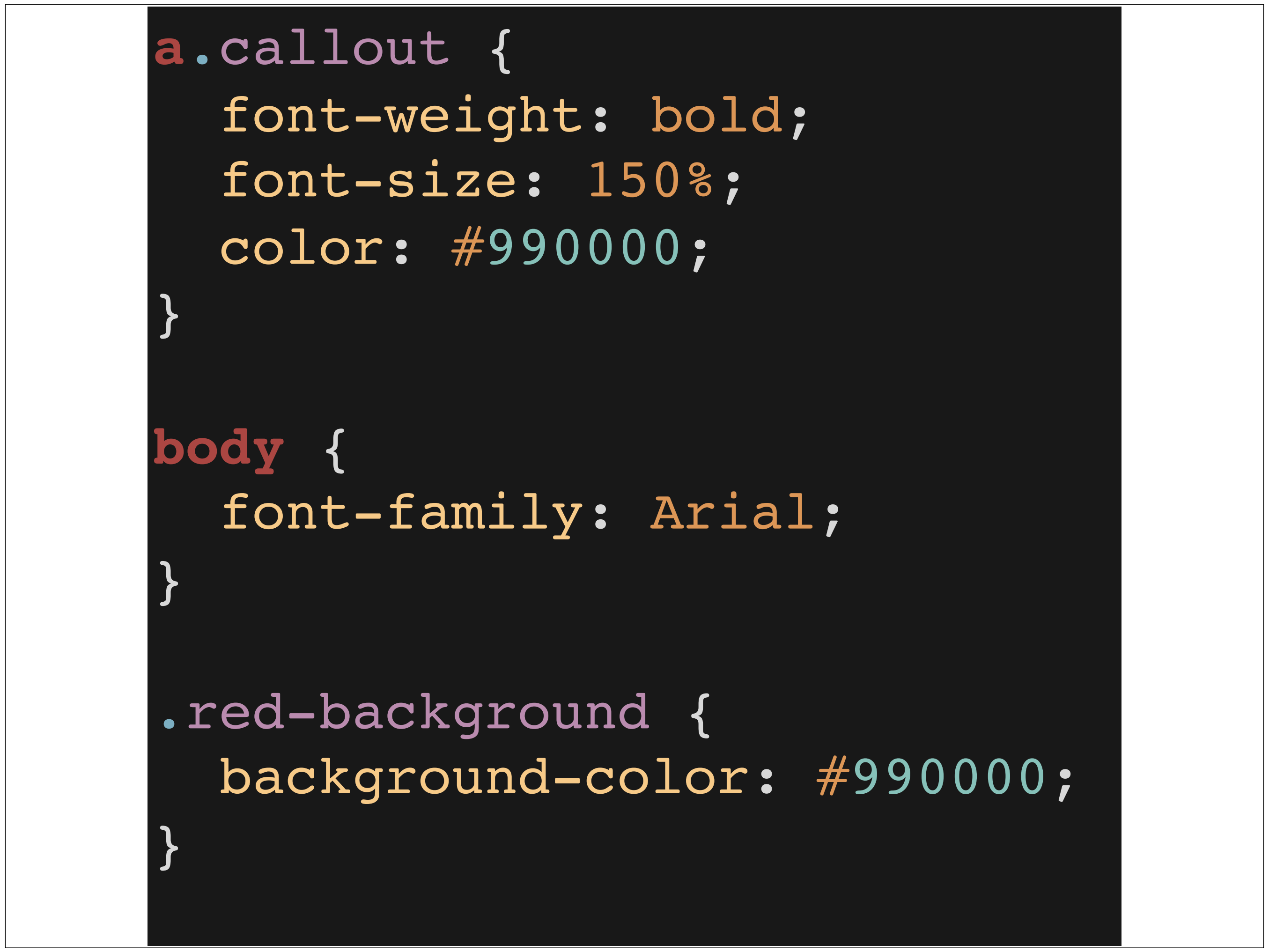
So all of that, for the reasons explored + more, led to the implementation of CSS, or Cascading Style Sheets. While the need for some sort of visual stylesheet for HTML was identified fairly early on in 1996 and small elements were adopted by earlier browsers, the actual first fully working implementation of CSS Level 1 wasn’t actually released until the year 2000.

Even then, CSS didn’t support all of this styling.



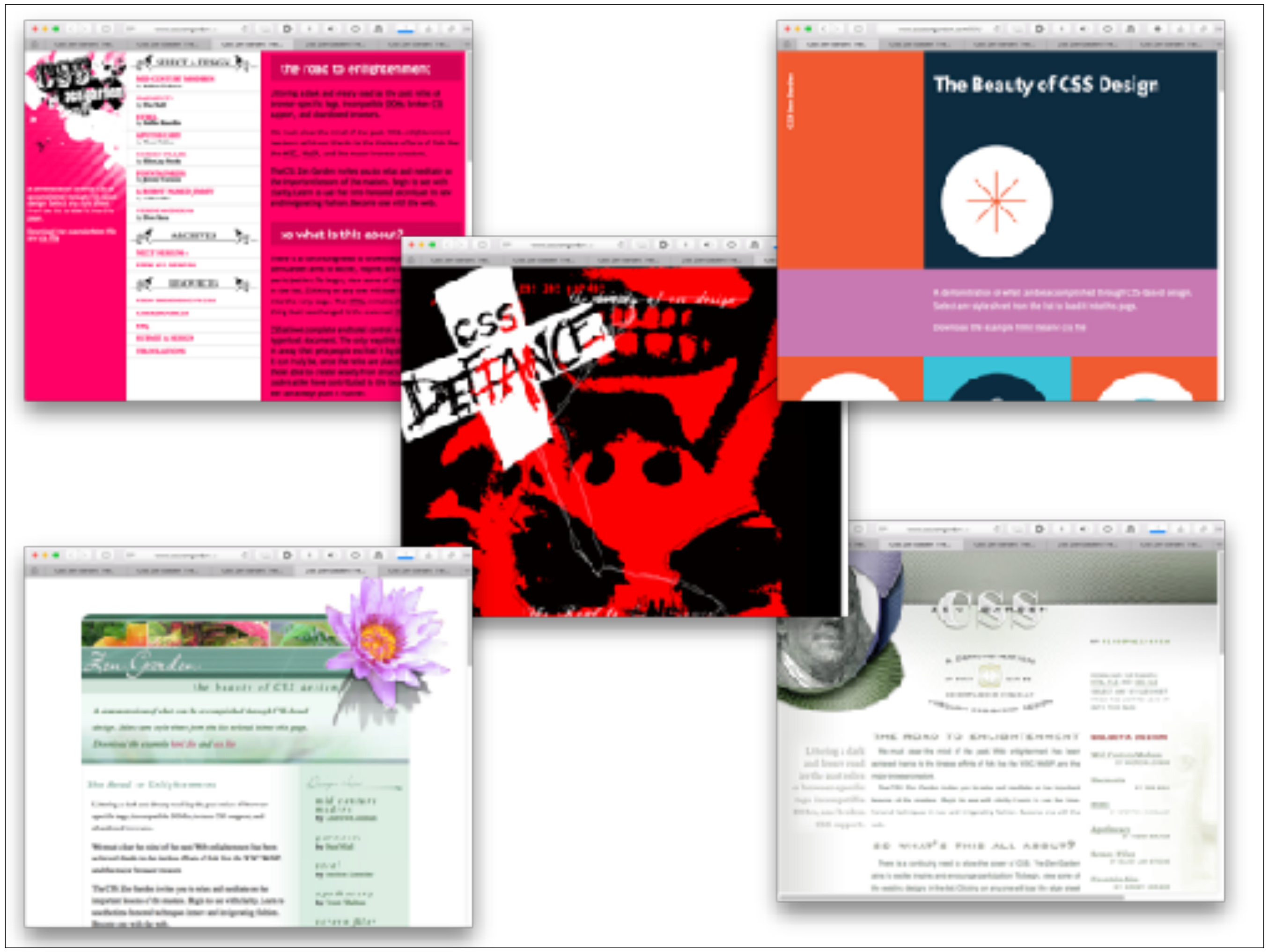
This chart shows roughly what was part of each level (although CSS 2.0 itself is missing). You can see that CSS1 mostly focused on the fonts, colors, backgrounds, and text properties. Pseudo classes are for things like hovering over a link, and cascade specifies what happens if there are multiple rules across multiple CSS files. (the colors around the outside dictate the current status of those proposals within the W3C's system.)

So let's look at what some CSS1 for that NBA page might look like.

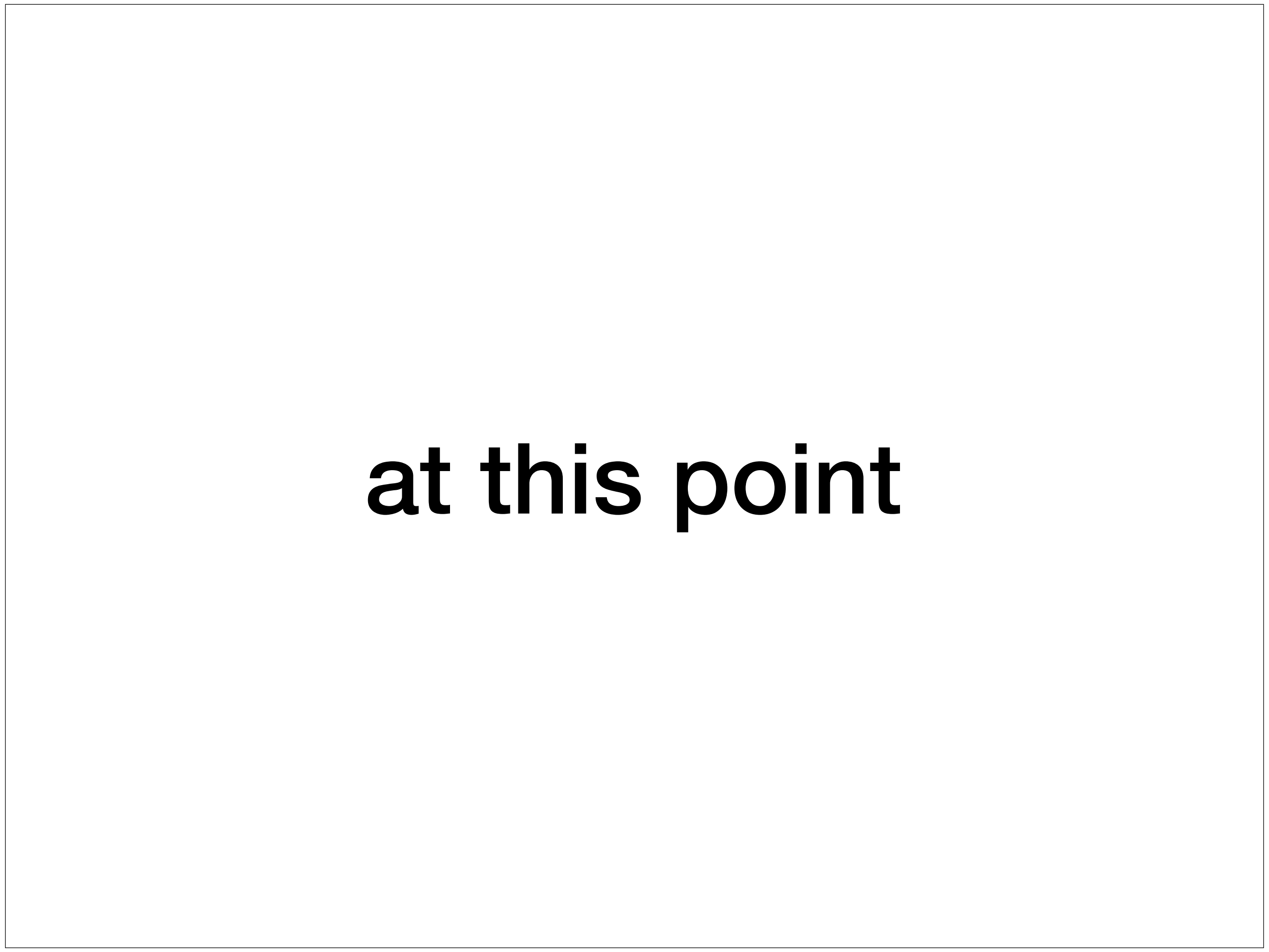


```css
a.callout {
  font-weight: bold;
  font-size: 150%;
  color: #990000;
}

body {
  font-family: Arial;
}

.red-background {
  background-color: #990000;
}
```

So this, combined with minor changes to the HTML to remove the now-unneccessary font tags and add the classes where necessary, is enough to completely get rid of every font-tag based complaint I had. If you're not familiar with CSS, (walk through CSS syntax, semantic CSS apology) The rest of it can be done with CSS2, though for the era we're talking about it can actually get weirdly complicated due to details of how the CSS box model operates. (while CSS was still an improvement, there was something to be said for how easily tables got the job done.) There could be a whole talk about liquid columns and faux columns and holy grail layouts of the mid-2000s, but for now let's just say that recreating our earlier layout is possible (if weird).


the road to enlightenment
so what is this about?
CSS DEFIANCE
The Beauty of CSS Design
Zen Garden
the beauty of CSS design
The Road to Enlightenment
CSS
THE ROAD TO ENLIGHTENMENT
SO WHAT'S THIS ALL ABOUT?

And people started to do wonderful things with it - these are just a few of some of the designs over the years posted to the CSS Zen Garden, still up. The idea is, everyone gets the same HTML; everything you do to it, you do with CSS. Even with the limited toolsets of earlier versions of CSS, the flexibility and ease in what you could do with just changing a few lines of code was a breath of fresh air.

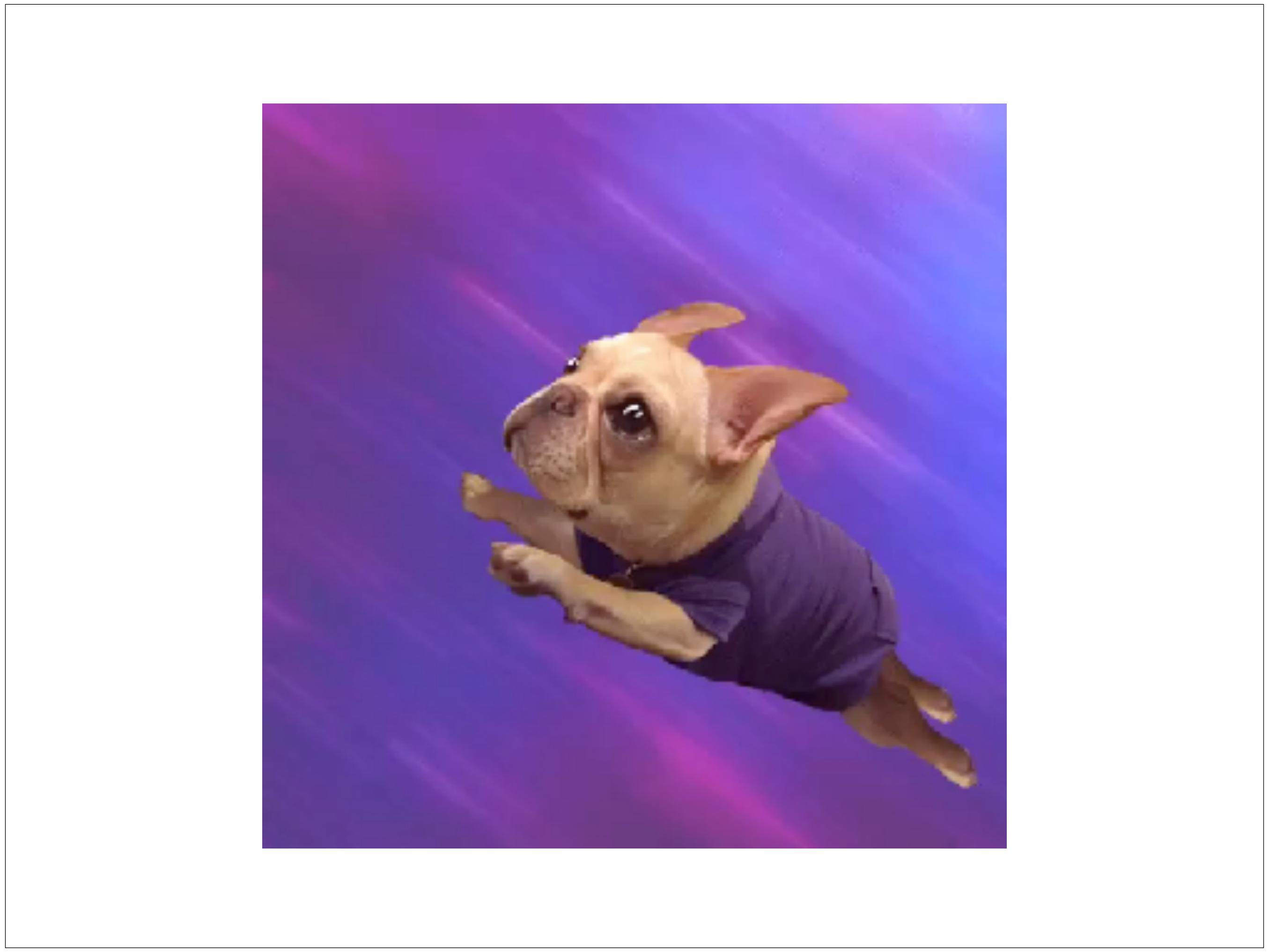


at this point

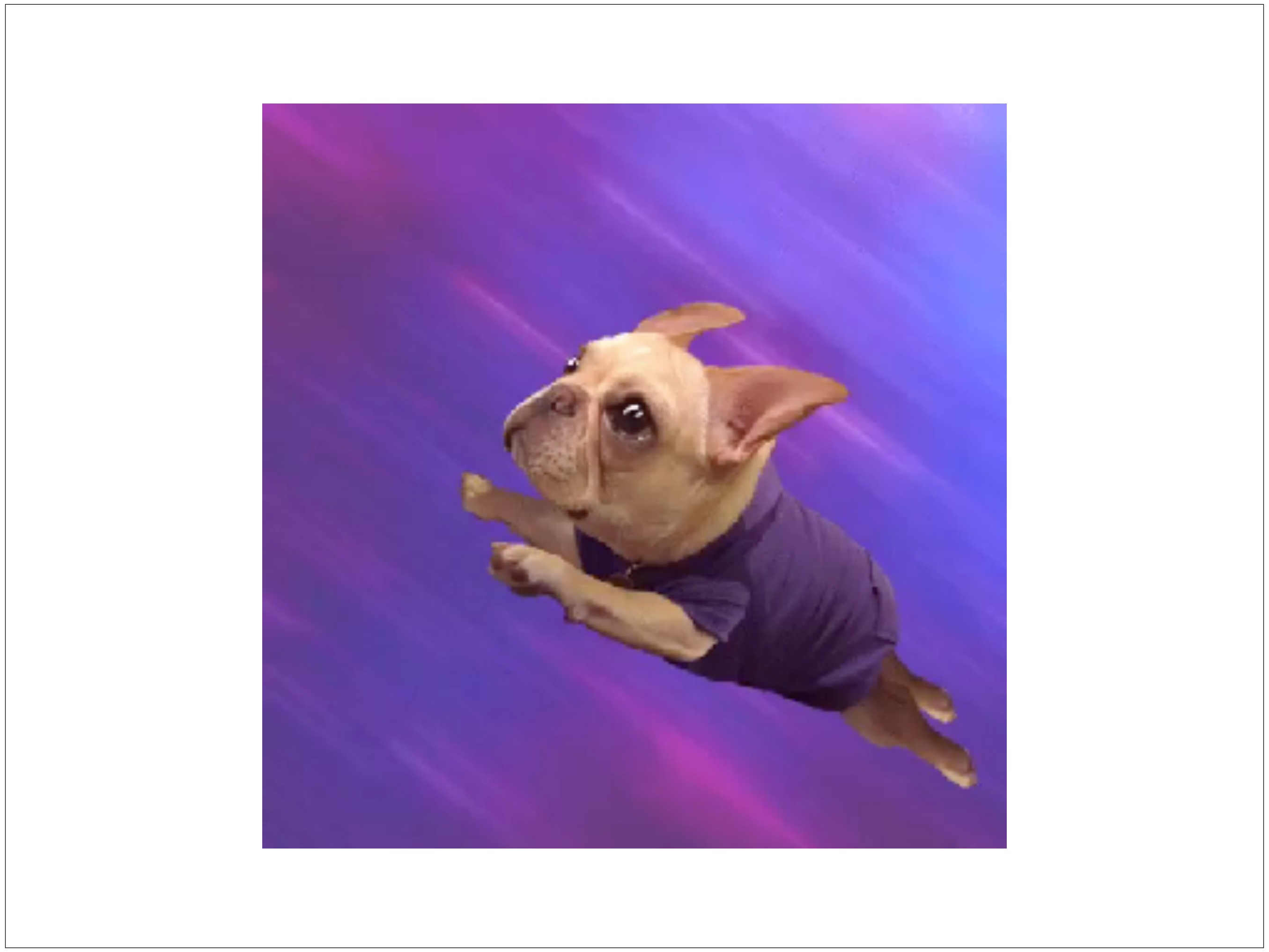
At this point, you might be thinking…

“ok, that’s great”

ok, that’s great. there were a series of standards and everyone went off into the sunshine and everything was puppies



puppies and rainbows, right?



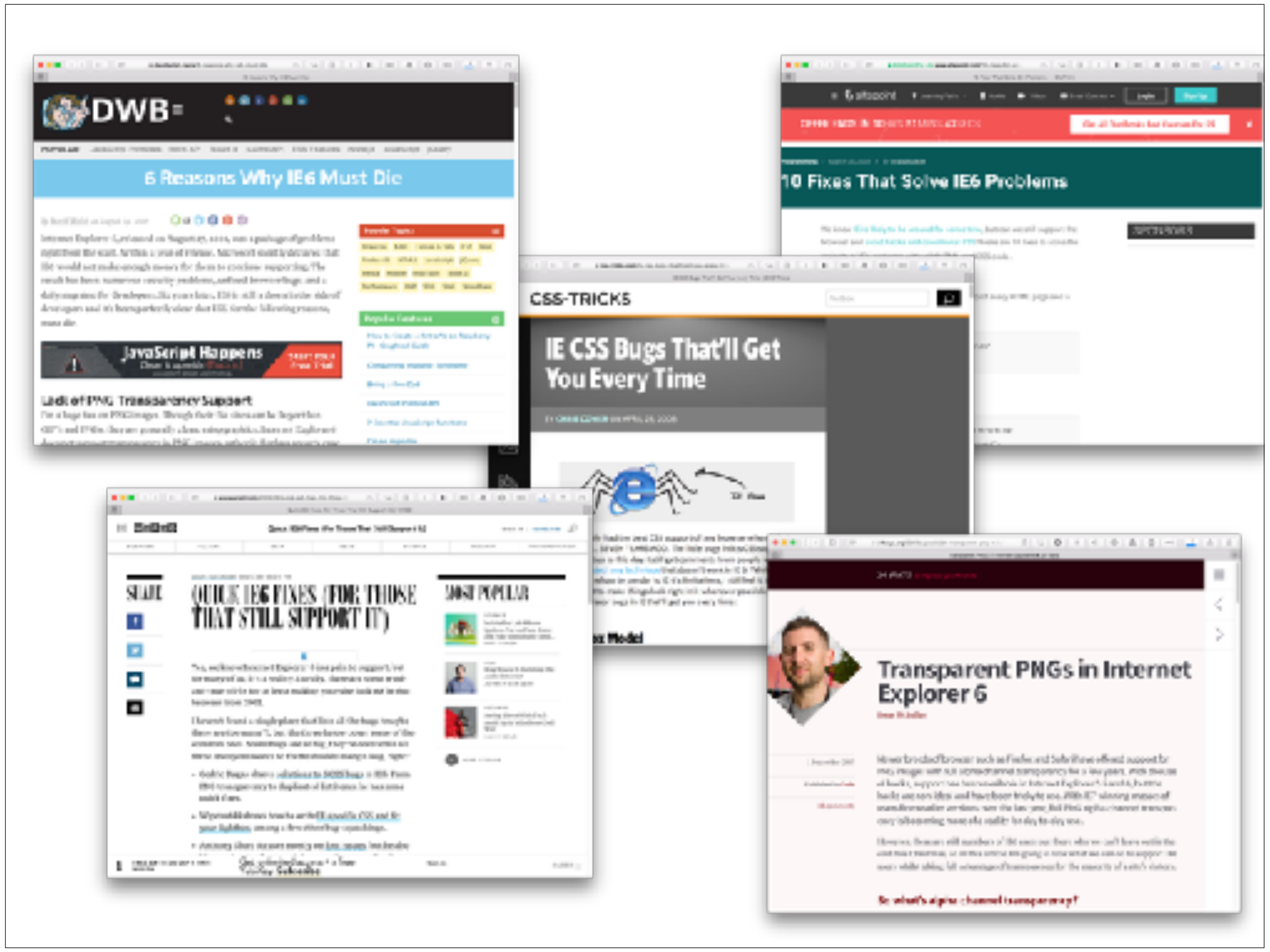
puppies and rainbows, right?

unfortunately… no. our story takes a dark turn. and a lot of it was due to

unfortunately… no. our story takes a dark turn. and a lot of it was due to

this.

Internet Explorer wasn’t actually always the worst browser when it came to CSS support - Netscape 4 was also around for quite a while and completely lacking in modern support. But when it came to “scourge of the web” status, nothing quite beat Internet Explorer 6.



DWB

6 Reasons Why IE6 Must Die

JavaScript Happens

Lack of PNG Transparency Support

10 Fixes That Solve IE6 Problems

CSS-TRICKS

IE CSS Bugs That'll Get You Every Time

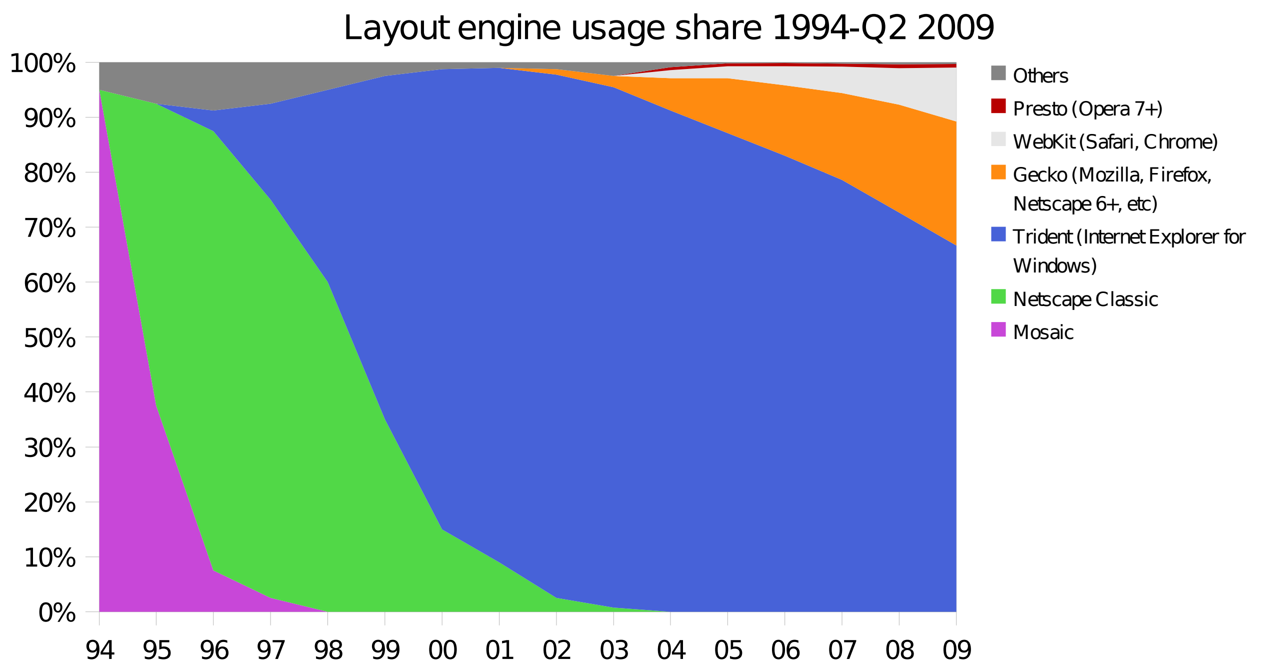
SHARE

QUICK IE6 FIXES (FOR THOSE THAT STILL SUPPORT IT)

MOST POPULAR

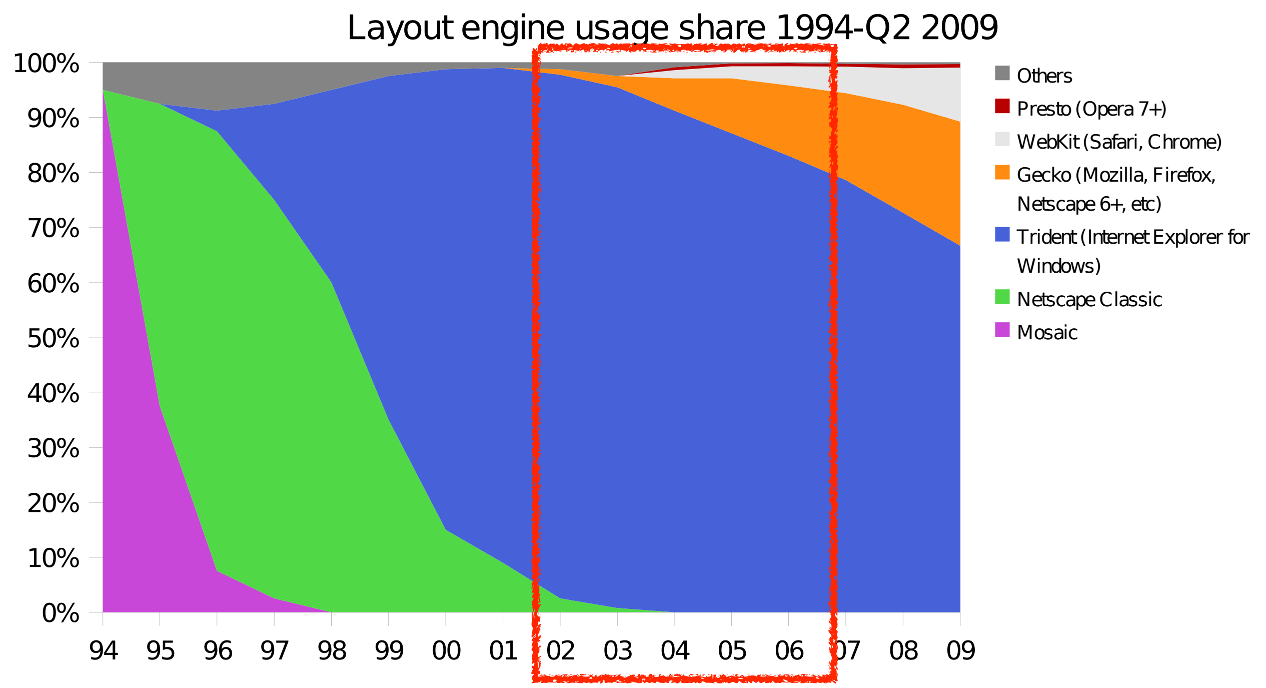
Transparent PNGs in Internet Explorer 6

These are just a few of the many articles out there talking about the bugs, quirks and brokenness of internet explorer 6. There were a wide variety of web standards that it bungled, but CSS bugs were the ones that were terrible. Broken selectors so you couldn’t select the same elements you could on other platforms, double margin bug, box model bug, no support for minimum widths or heights, adding hasLayout, no transparent PNGs without weird hacks… those are just a few of a 2000s-era web guru’s favorite things. and all of them required



And this was made worse by how popular it was. This is a chart of the major web rendering engines in use from 1994 to 2009. Internet Explorer 6… <build> was out for five years. From August of 2001 to October of 2006. At its peak, IE6 had over 90% browser market share. _90%_. there are a lot of reasons for this, but that’s _insanity_.

so what did we wind up doing?

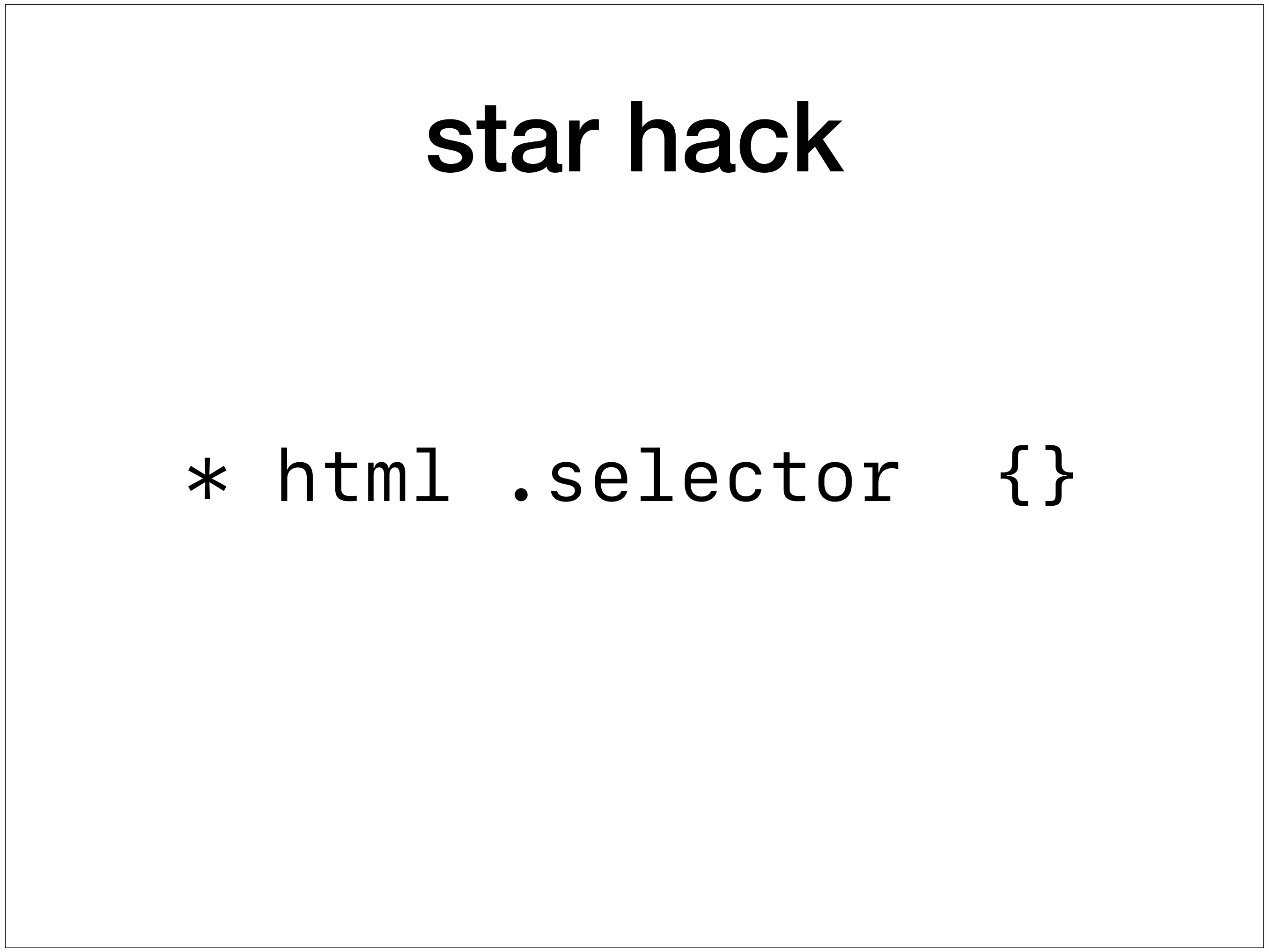


And this was made worse by how popular it was. This is a chart of the major web rendering engines in use from 1994 to 2009. Internet Explorer 6… <build> was out for five years. From August of 2001 to October of 2006. At its peak, IE6 had over 90% browser market share. _90%_. there are a lot of reasons for this, but that’s _insanity_.

so what did we wind up doing?

hacks

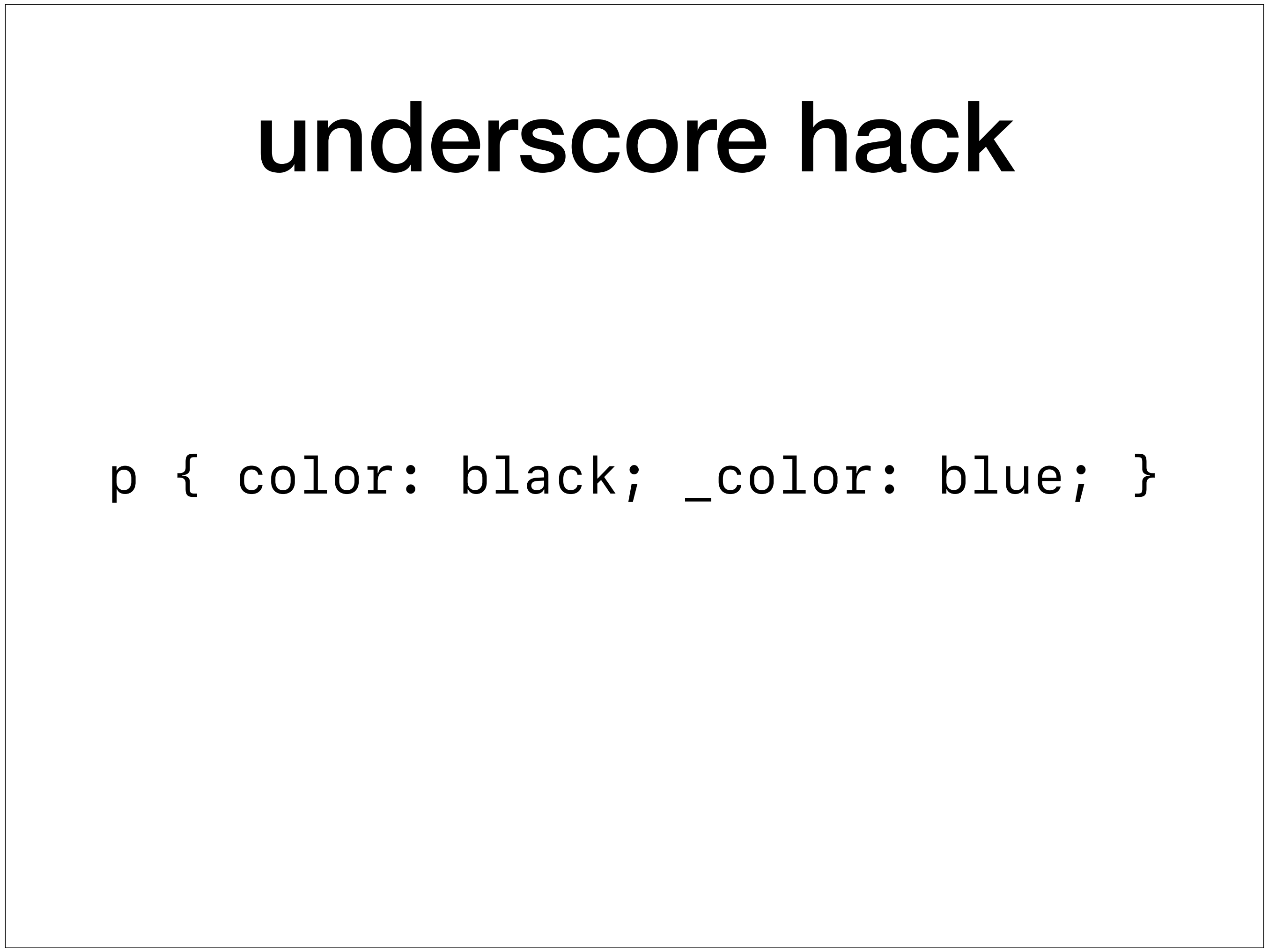
hacks. sometimes we had to feed special CSS to Internet Explorer to override bad assumptions or broken functionality.



star hack

```
* html .selector  {}
```

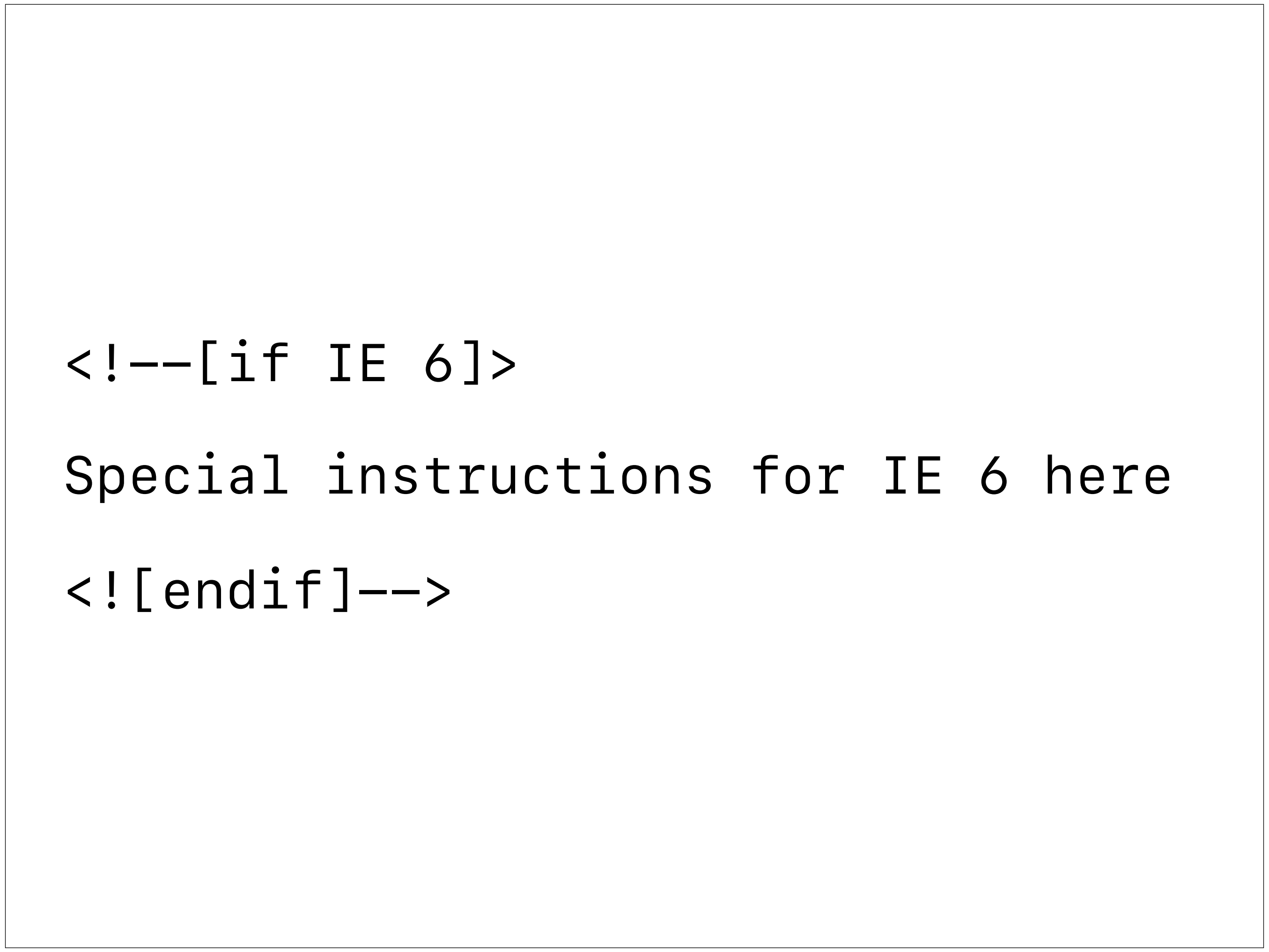
This is one of my favorites, it’s called the star hack. This wouldn’t select anything in a properly standards compliant browser; it’s first selector is looking for an element above the HTML tag, which is always the surrounding tag in an HTML document, so it will never find anything. But, Internet Explorer’s CSS implementation is just broken enough that it, and only it, will lead any CSS that’s in this tag. Perfect for putting in different rules it will understand while keeping away from those pesky standards-based browsers.



underscore hack

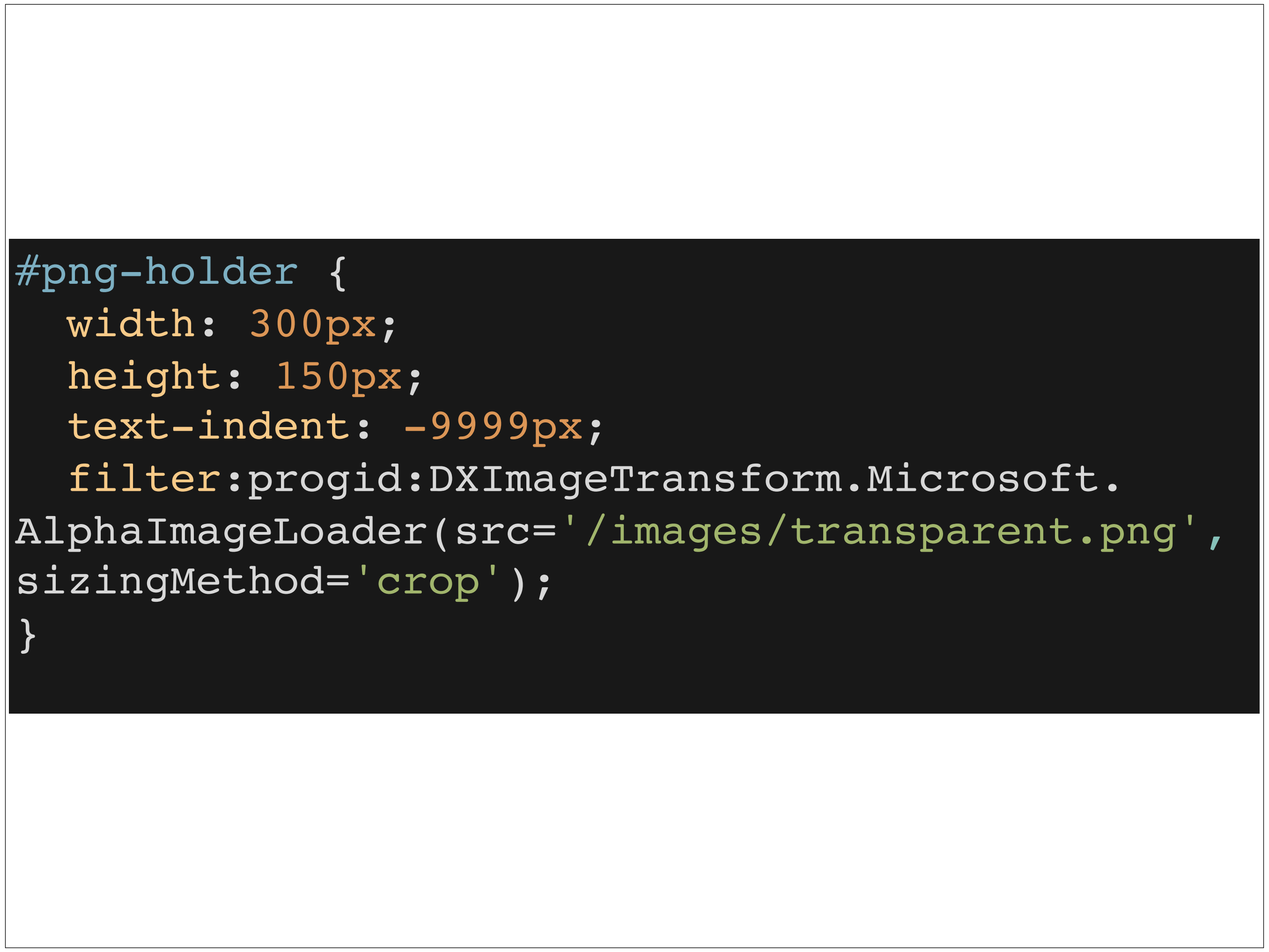
```
p { color: black; _color: blue; }
```

This is another great one. In any standards-compliant browser, it will read the color of a paragraph tag as black. It sees the underscore as part of the blue attribute as a completely different thing. But again, Internet Explorer is just broken enough to read them both as “color”, and the second one wins out. Now there were other ways to do this



```
<!--[if IE 6]>

Special instructions for IE 6 here

<![endif]-->
```

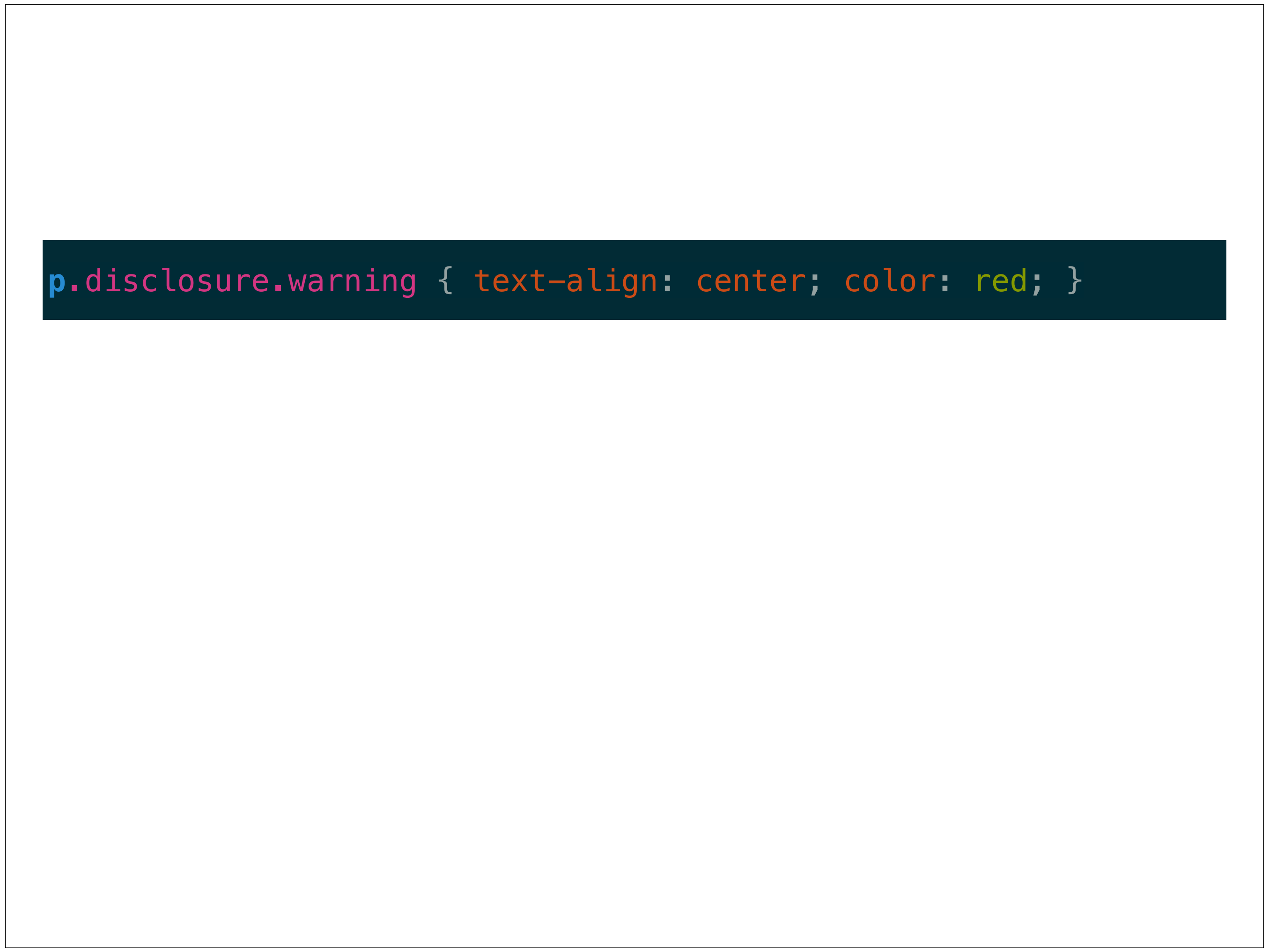
Conditional comments were the less fun, more sensical way to selectively apply custom CSS to Internet Explorer. And today we have browser vendor prefixes for features that are experimental or not fully implemented. But that didn’t get around the parts of Internet Explorer that were genuinely broken, like



```
#png-holder {
  width: 300px;
  height: 150px;
  text-indent: -9999px;
  filter:progid:DXImageTransform.Microsoft.
AlphaImageLoader(src='/images/transparent.png',
sizingMethod='crop');
}
```

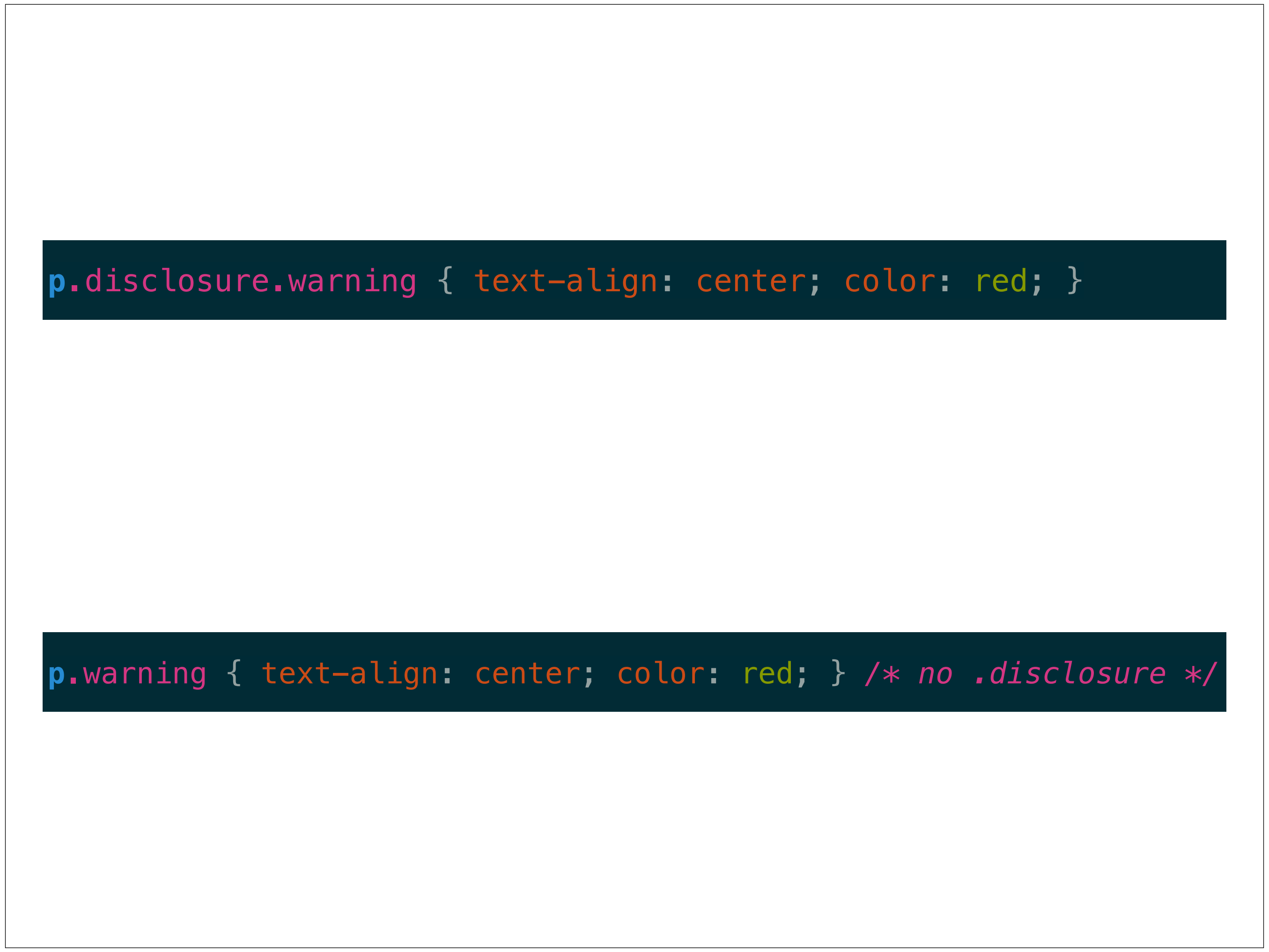
transparent PNGs. IE6 didn’t support transparency in PNGs natively, so you either had to use a GIF - often not desirable, since they’re only 8 bit color and don’t support alpha transparency - or do this. What this is actually doing is, the element on the page contains an image tag, and this indents the image way off the page with the text-indent, then adds a proprietary Microsoft-only attribute to apply a DirectX filter that _just so happens_ to be able to read a PNG correctly.

This was ridiculous, and this particular had a lot of downsides that made it not always suitable. But people found ways around all this stuff. There were worse things that were broken, though.



```css
p.disclosure.warning { text-align: center; color: red; }
```

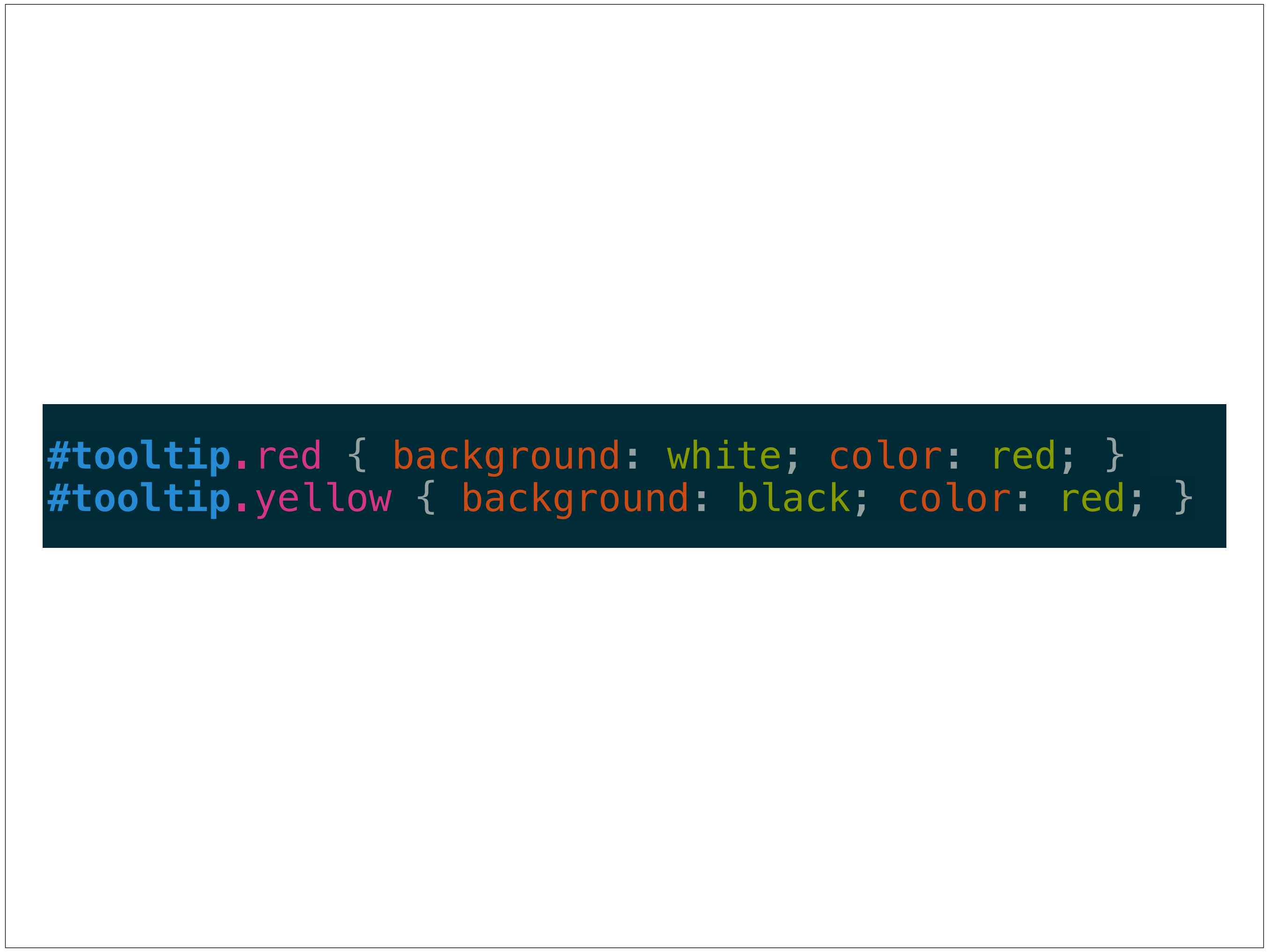
For instance, this should select a paragraph tag with a class attribute of both disclousure and warning. <build> But IE6 actually reads it as just warning, and ignores the disclosure.



```
p.disclosure.warning { text-align: center; color: red; }
```

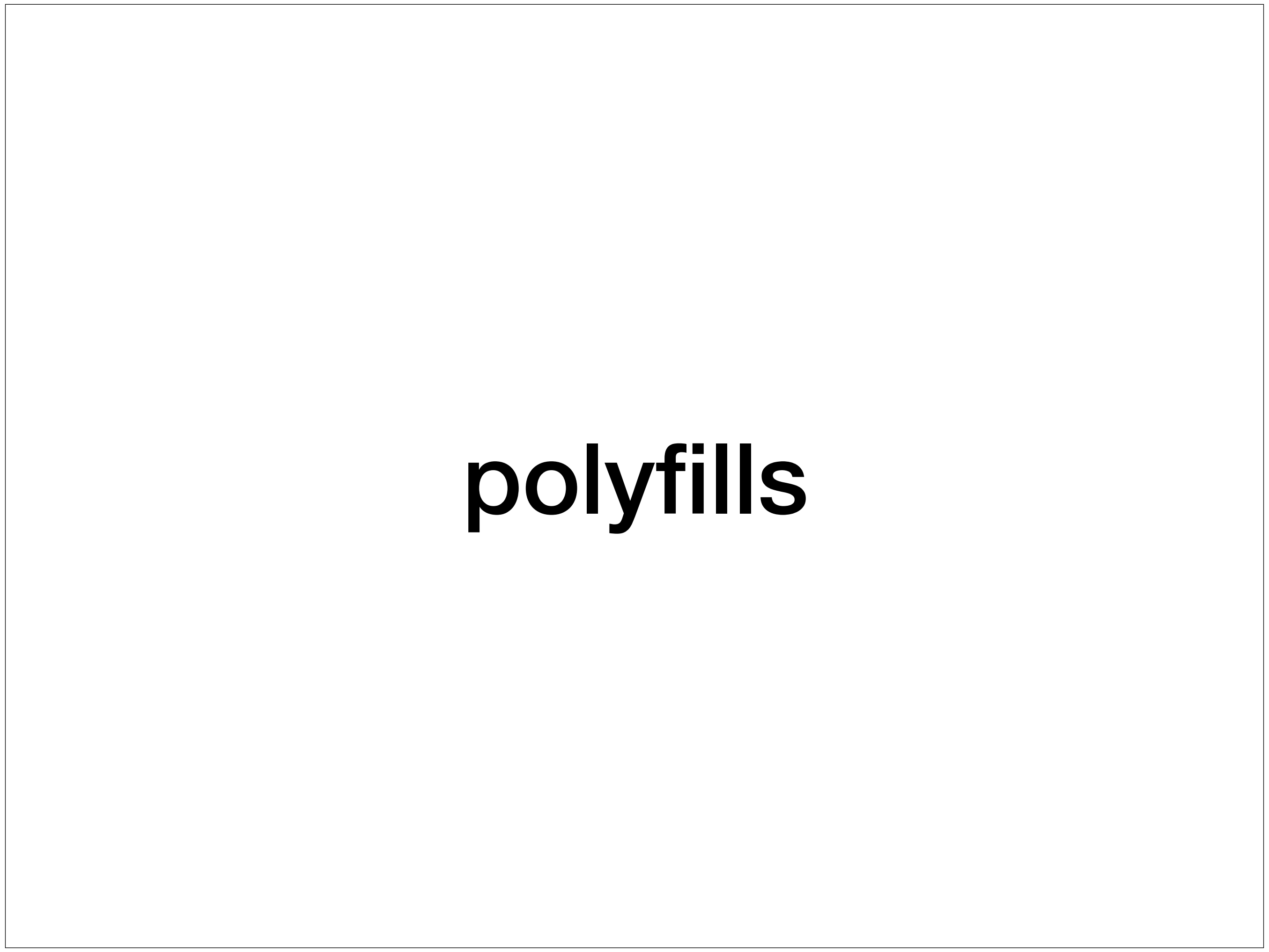
```
p.warning { text-align: center; color: red; } /* no .disclosure */
```

For instance, this should select a paragraph tag with a class attribute of both disclousure and warning. <build> But IE6 actually reads it as just warning, and ignores the disclosure.



```
#tooltip.red { background: white; color: red; }
#tooltip.yellow { background: black; color: red; }
```

Another bug is that IE6 can only see the first one of these. These are both completely valid selectors, selecting an element that has an ID of tooltip and a class of red or yellow. Yet IE6 will only be capable of reading the first one. The only way around these types of problems was to change your HTML structure such that you don't need to do this, adding additional useless structure and clutter to your page.



polyfills

This environment of one dominant but sorely lacking browser is where the polyfill was born. A polyfill, for those who are unaware, is a library (typically Javascript) that adds functionality that is standardized, but not available or yet standardized across all platforms. Ideally, a polyfill should provide its functionality as close to the standard as possible in any browser that doesn’t support it - so close that you don’t even have to think about what it’s doing behind the scenes - but not even activate itself when running in a browser that it does. Something of a hack in itself.

html5shiv

The most popular of these for a time was html5shiv. HTML5 - among a whole host of huge improvements in specification and documentation, and providing a more living specificification - also introduced a whole bunch of semantic tags for common portions of a page, like article, header, footer, aside, nav. Most browsers simply accepted these as HTML tags, but Internet Explorer (up until version 8), by default, didn’t apply CSS to these new elements. But it turns out if you called just the right small snippet of Javascript, Internet Explorer would suddenly recognize these elements, with no detriment to usability.

CSS3 PIE

One of the crazier polyfills was CSS3 PIE (short for Progressive Internet Explorer). Even after Internet Explorer 7 was released in 2006, IE6 stuck around for a long time, and exciting new CSS standards were being introduced for things like rounded corners, and gradients. It took Microsoft a long time to catch up to these with version 9 in 2011, so most designers wrote progressive fallbacks and just didn't display these in IE. If you wanted them to appear, you could either write the javascript and CSS modifications yourself… or you could insert this javascript library that would automatically scan through your standards compliant syntax and add whatever was neccessary to make your site work correctly. (Of course with a lot of potential side effects, but still. It was impressive.)

eventually IE6 went away

now. eventually IE6 went mostly away, thankfully. though for some of us writing apps for corporate users it hung on much longer than we would’ve liked. (this guy right here).

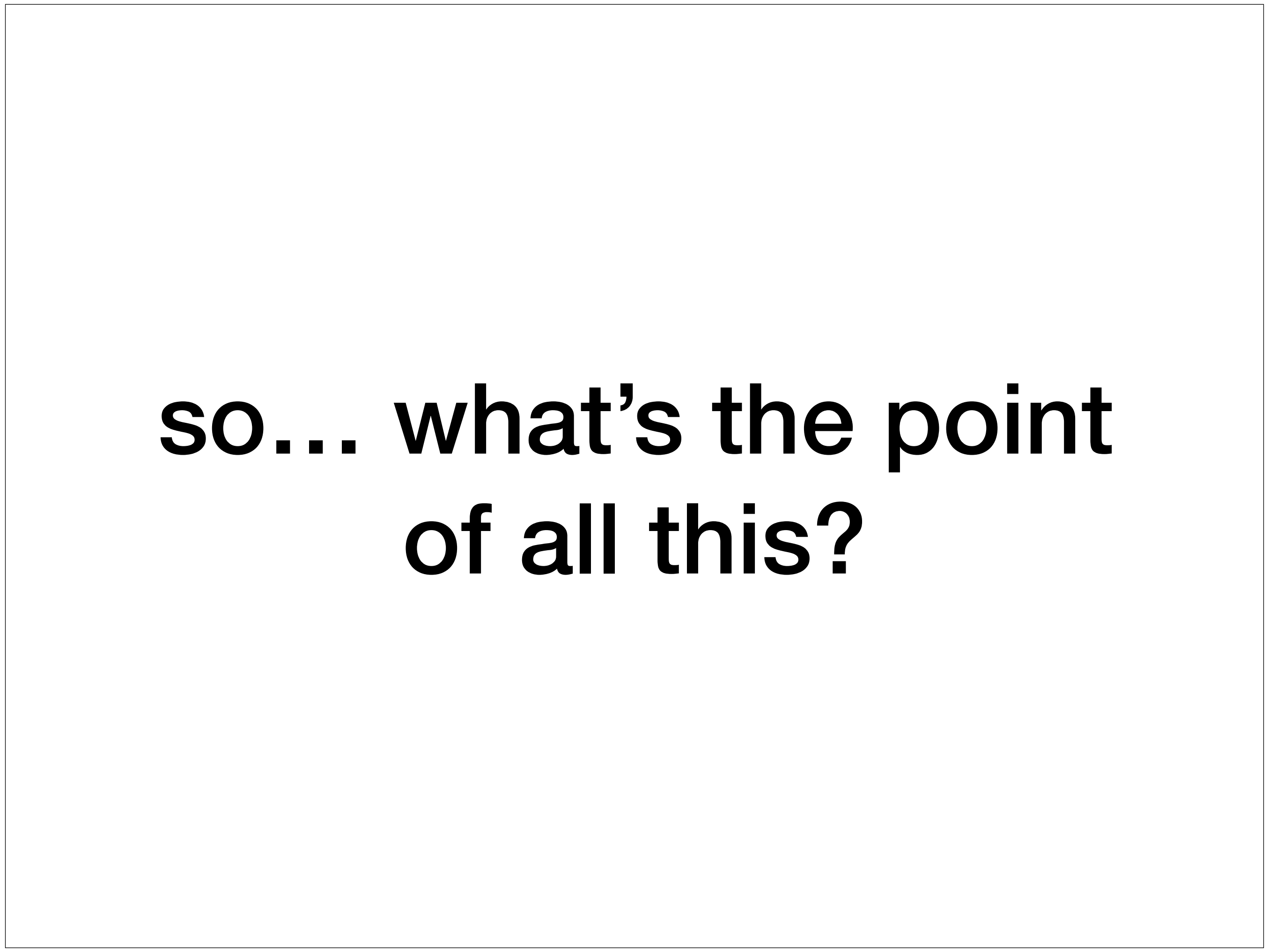
(fun fact - that first browser I was talking about with CSS support? IE 5 for Mac. It used a completley different rendering engine. useless for previewing how things would work on windows.)

but the concept of the polyfill is still with us, implementing new features or papering over partially implemented standards wherever neccessary. They aren’t just limited to CSS, offering up all sorts of helpful features for users of Javascript or HTML as well. They help us not have to worry about the lower level details.

but, despite the best efforts, CSS was still kind of a pain to use for layouts. Because the web originated as just a flowing amount of text, most of CSS's first steps toward adding layout control assumed that a designer wanted to think in that way. elements could be positioned on the page, either absolutely to the bounds of the page itself, or relative to other elements on the page, but would then be taken out of the flow of the page, and sizing things based on other elements - or even the contents of those elements was tricky. Or, you could float an element to the left or the right of the page, but that would also make its position shift based on other elements in the container… it required a lot of knowledge, and often a lot of balanced math to make sure elements didn't overlap or cause page flow to go crazy. So, we did what we often do

abstractions

and built abstractions. frameworks and compiled languages. While there are absolutely earlier CSS frameworks that popped up to help deal with this, Bootstrap is the most popular and familiar today. and while its default markup can be controversial, having a whole bunch of this work automated out of your way so you don’t even have to think about it has proven to be a rather popular idea. And similarly, compiled languages like SASS or Less or postprocessors like PostCSS have helped us. Sass initially became hugely popular because of its ability to do math, set variables, transform colors, and iterate over elements, and is still part of a framework like boostrap today. And the features it introduced are slowly bbeing introduced into CSS itself in new standards.



so… what's the point of all this?

When I first had the idea for this talk, I was originally thinking about the evolution of tech stacks on the web, and how and why each layer in the stack evolved, and the story of CSS was just one story that I wanted to tell. But as I started to flesh out that part of the story, I sort of realized that the story of CSS is a perfect microcosm of the web at large.

The spiraling complexity followed by a moment of clarity, the gradual drift to a standard that is then improved upon… Javascript, RESTful APIs, canvas and WebGL, WebRTC, offline progressive web applications… all of these started as hacks or unconventional, were improved into a diaspora of different standards, and then - often - eventually standardized upon. A small fractal of the larger story that is the web's history.

And I think this the story of the web is particularly interesting because the web is the only ubiquitous open platform we have. Every other major platform, desktop or mobile, has its destiny controlled by a single corporation. And there are certainly many large and powerful interests out there that are steering the direction of core technologies. But I'd argue, this is the closest thing we have to modern software at its most unencumbered.

(I know there are Linux users out in the audience internally screaming at me, and I love Linux. I've built tons of servers with it, and dabbled with it on the desktop off and on since I was a teenager. But the next year has been the year of Linux on the desktop for at least the last 20 years; and even setting aside the joke, for a lot of people it would only be a viable solution _because_ of the web.)

So, what can we take away from this?

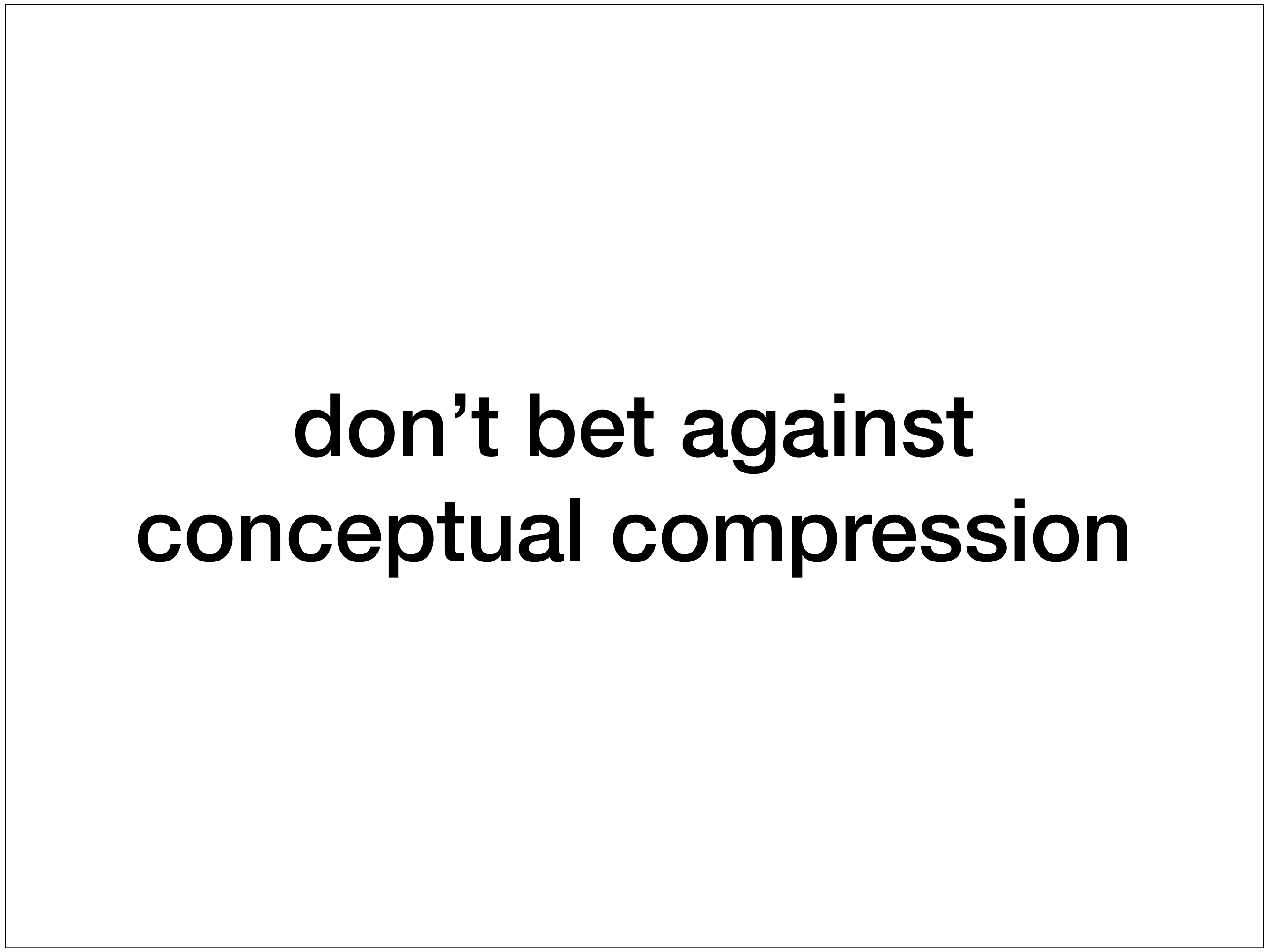
standards will win

First, standards will win. <build> eventually. And note, I’m explicitly not saying that _one standard_ will win. There’s still plenty of examples of multiple ways to get a thing done. But generally speaking, if someone builds a technology that works and is useful, developers will demand it. And once that happens, the pressure is on. Although one browser does come scarily close, there’s no longer one browser that controls the entire ecosystem, and that means that browsers - and their respective platform holders - still need to keep up.

standards will win

(eventually)

First, standards will win. <build> eventually. And note, I’m explicitly not saying that _one standard_ will win. There’s still plenty of examples of multiple ways to get a thing done. But generally speaking, if someone builds a technology that works and is useful, developers will demand it. And once that happens, the pressure is on. Although one browser does come scarily close, there’s no longer one browser that controls the entire ecosystem, and that means that browsers - and their respective platform holders - still need to keep up.



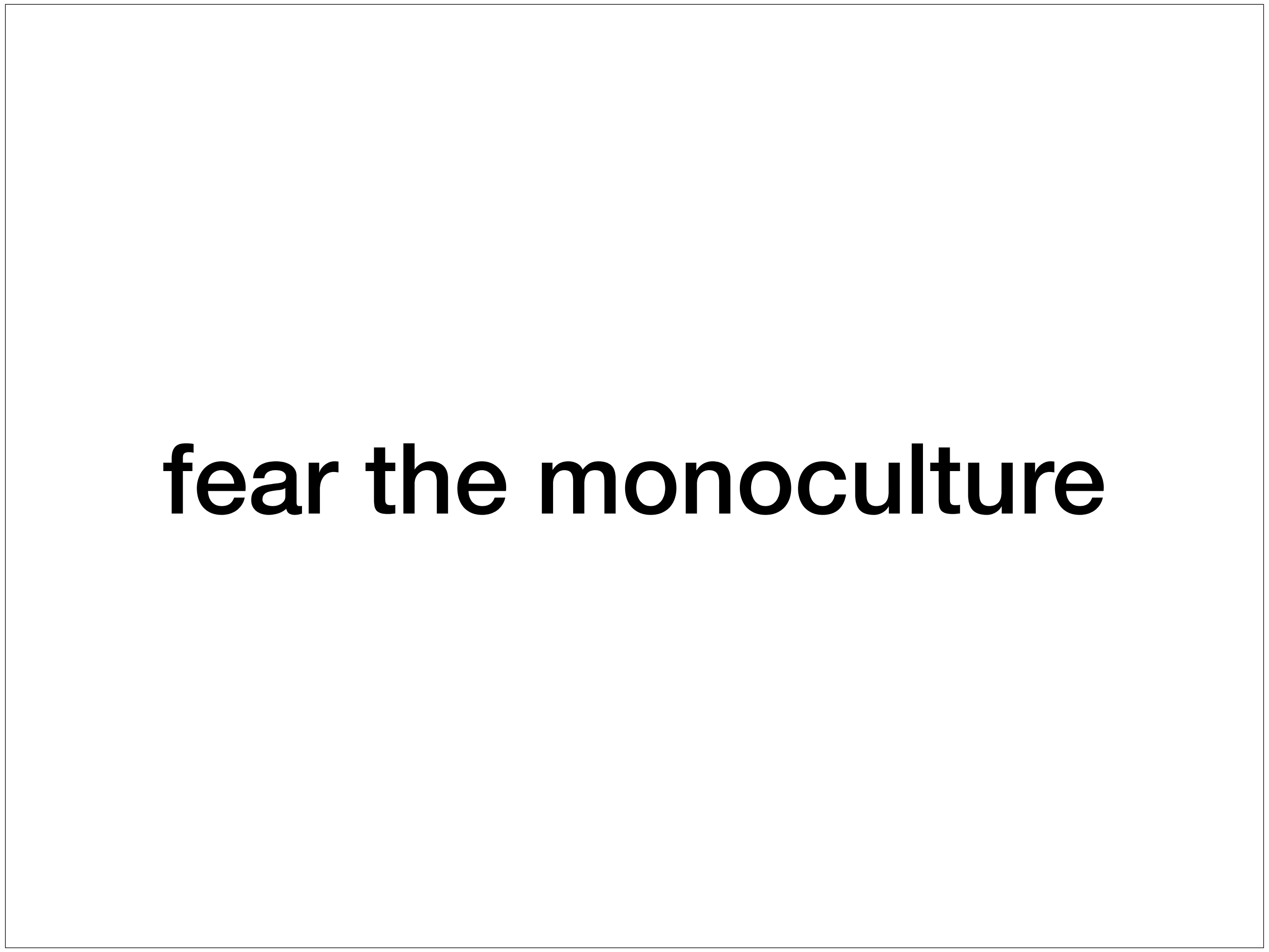
don’t bet against conceptual compression

In David Heinemeier Hanson’s keynote at this years RailsConf, David coined the term “conceptual compression” to refer to lower-level concepts that most people no longer need to know to be effective when developing software. (It was actually a remarkably powerful talk with a clear-eyed vision of how conceptual compression _might_, if we’re very lucky, help save us all. I’d really suggest watching it if you haven’t had a chance.)

Taking the timeline of computing at large, this isn’t really news… just look at what the difference an operating system from 1988 and an operating system in 2018 does. But the microcosm of the web offers us a recent and fascinating look at exactly how this evolves in the open. Nobody even _dreamed_ of the utility of a compiling CSS framework in 1991. Yet now there’s multiple. Granted you wouldn’t have _needed_ one in 1991, but look at how much more the web does now, and how large of a role it has on our everyday lives. The fact that I can download Bootstrap and start making a software design that works on billions of devices of all different shapes and sizes is _amazing_ and powerful, and it’s thanks to conceptual compression that we can do so.

The next is that conceptual compression compresses, kind of like

layers of rock. First something starts off difficult, starts off complicated. Perhaps then a framework comes by and adds abstractions to make it easier, or a library that comes in and provides the just tool you need. Eventually, enough people are doing the thing you want that a standard is proposed, and if it's accepted, the layers that were built are no longer required; it's part of the bedrock now. (maybe you need to do some shoring up with a polyfill, but hey, it's beter.) Finally



fear the monoculture

now, granted, Google is… not the worst steward for a browser, and at their current pace they’d ship 30 versions in the time it took Microsoft to ship one. but Chrome has almost 60% of the worldwide desktop browser market share, and 55% of the global mobile market share. and there are already occasional experiments or features that Google has released that don’t just need features only Chrome has yet implemented, but refuse to run on any browser but Chrome. I don’t really want to go back to 2003. I lived in that time and the legacy of building sites for IE6 for years afterwards, and while we did find clever hacks to get around this… I don’t really want to go back. none of us do. now I’m not saying you need to switch your personal browser of choice, switch mobile platforms, whatever… but maybe try a different browser. and definitely test in all of them, and have graceful degradation or fallbacks if neccessary. People have all sorts of different reasons for using the browsers they use, and just shrugging our shoulders and saying “use this one” at people’s needs isn’t very inclusive. The web being open is a precious gift… let’s try not to hand control over to any one company. Thank you.

thanks!

Rufo Sanchez
@rufo
me@ru.fo