RESPONSIVE WEB DESIGN

THE MOBILE WEB AND MORE

Jonathan Engelsma, Ph.D.

TOPICS

- Web Page Layout: a historical perspective
- Grid Layouts
- The Mobile Web / Mobile First
- Responsive Design

NO STYLE!

yahoo.com circa 1995

NO STYLE!

craigslist.org circa 1998

See: Why the Web Hasn't Birthed a Prettier Craigslist

EARLY ATTEMPTS AT STYLING

- Images: introduced in HTML 2.0 (1994)
- Image maps:
  - Mosaic 1.1: server side image maps (1995)
  - HTML 3.2 client side image maps. (1996)

CLIENT SIDE IMAGE MAPS

```
<!DOCTYPE html>
<html>
<body>

<p>Click on the sun or on one of the planets to watch it closer:</p>

<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap">

<map name="planetmap">
  <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">
  <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">
  <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">
</map>

</body>
</html>
```

- Used to be used on almost every web page!
  - Used to implement navigation schemes

LIMIT USE OF IMAGE MAPS

- Problems
  - Longer load times (think mobile)
  - Horrible from an accessibility perspective.
  - Painful to create.
  - Really out of style from a design perspective.

MORE ATTEMPTS AT STYLING

- Adobe Flash (introduced in 1996)
  - required a browser plug-in
  - allowed complex highly interactive websites to be built.
  - broke HTTP concept of addressability

Apple struck a very controversial chord when they banished Adobe Flash from their iOS platform. Was this primarily a technical decision or a business decision?

See Thoughts on Flash by Steve Jobs.

MODERN FLASH SITE EXAMPLES

- Flash is still being used… some interesting sites:
  - http://cloudsovercuba.com/
  - http://wechoosethemoon.org/

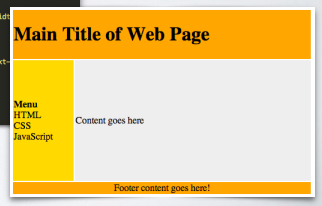
YET MORE ATTEMPTS AT STYLING

- Tables: introduced in "HTML 3.0" for display of tabular info. Quickly subverted for page layout. (1995)
- fonts, colors, background images, etc. HTML 3.2 (1996)

LAYOUT WITH TABLES

```
<!DOCTYPE html>
<html>
<body>

<table width="500">
    <tr>
        <td colspan="2" style="background-color:#FFA500;">
        <h1>Main Title of Web Page</h1>
        </td>
    </tr>

    <tr>
        <td style="background-color:#FFD700;width:100px;">
        <b>Menu</b><br>
        HTML<br>
        CSS<br>
        JavaScript
        </td>
        <td style="background-color:#eeeeee;height:200px;wid
        Content goes here</td>
    </tr>

    <tr>
        <td colspan="2" style="background-color:#FFA500;text-
        Footer content goes here! </td>
    </tr>
</table>

</body>
</html>
```

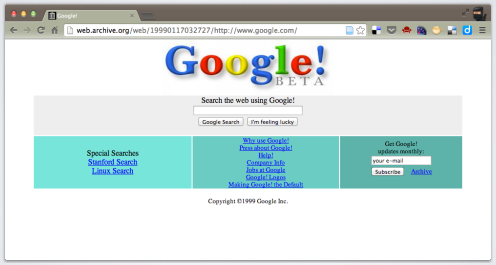
(Note that CSS wasn't widely used until years later.)



PRIMITIVE LAYOUT WITH TABLES

amazon.com circa 1996?

VINTAGE GOOGLE!



google.com circa 1999

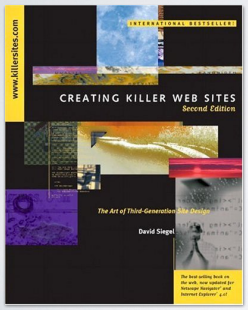
DON'T USE TABLES FOR LAYOUT!

- Using HTML Tables for layout purposes is NOT a best practice:
  - mixes presentational data in with your content. (why is this bad?)
  - makes redesigns of existing sites and content extremely labor intensive
  - makes it extremely hard (and expensive) to maintain visual consistency throughout a site.

DON'T USE TABLES FOR LAYOUT (CONTINUED)!

- Using HTML Tables for layout purposes is NOT a best practice (continued):
  - hard to get same content to display well on smaller screens.
  - does not lend itself well to accessibility.

HISTORICAL FOOTNOTE: A PREVIOUSLY POPULAR BOOK!



- Originally published in 1996 and one of the most popular books on the net.
- You still find people using Siegel's work arounds!

WEB PAGE LAYOUT TODAY

- HTML + CSS has largely eclipsed using HTML table elements or Flash for laying out web pages.

What are the advantages of using HTML/ CSS instead of HTML tables, etc.?

REASON #1

- By removing presentational markup from your pages, redesigns of existing sites and content is much less labor intensive.
- To change the layout of the site, all you need to do is change the style sheets; you do not need to edit the pages themselves at all.

Check out the CSS Zen Garden.

REASON #2

- Easy to maintain visual consistency throughout a site.
- Since pages use the same CSS document for their layout, they are all formatted the same.
- Improve usability and overall quality of the site.

REASON #3

- Improved accessibility for users with disabilities and users accessing with smartphones / tablets.
- separating content from presentation boosts device independence.

REASON #4

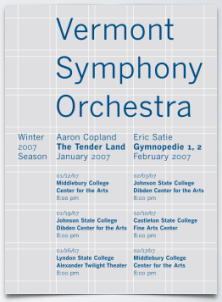
- Using HTML/CSS reduces the file sizes of your pages, as users no longer need to download presentational data with each page they visit.
- Stylesheets can be cached
- Faster page loads and less data used (big + for mobile)!

REASON #5

- minimizing markup and using header tags properly **improves search engine ranking**.
  - reduce the ratio of code to content
  - use keywords in the header tags
  - replace header GIFs with actual text

GRID BASED LAYOUTS

- Technique developed for and widely used in traditional print publishing.
- Popularized in web design by a number of recent web UI frameworks

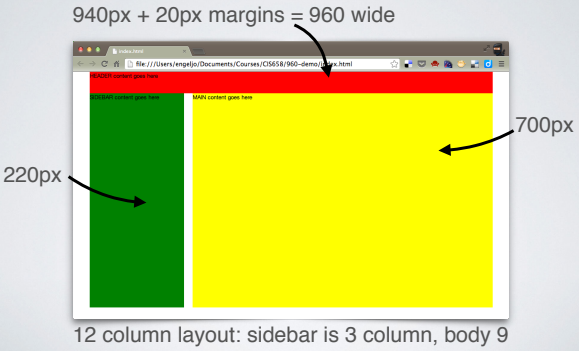


Source: http://en.wikipedia.org/wiki/Grid_(graphic_design)

960 GRID SYSTEM

- Popular CSS layout framework.
- modern monitors support at least 1024 × 768 pixel resolution.
- 960 is divisible by 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 32, 40, 48, 60, 64, 80, 96, 120, 160, 192, 240, 320 and 480, making it a flexible number to work with.

960 GRID LAYOUT EXAMPLE



960 GRID LAYOUT CODE EXAMPLE

```
<link rel="stylesheet" type="text/css" media="all" href="css/reset.css" />
<link rel="stylesheet" type="text/css" media="all" href="css/text.css" />
<link rel="stylesheet" type="text/css" media="all" href="css/960.css" />
```

```
<div id="container" class="container_12">
  <div id="header" class="grid_12">
    HEADER content goes here
  </div>

  <div id="sidebar" class="grid_3">
    SIDEBAR content goes here
  </div>

  <div id="mainContent" class="grid_9">
    MAIN content goes here
  </div>
</div>
```

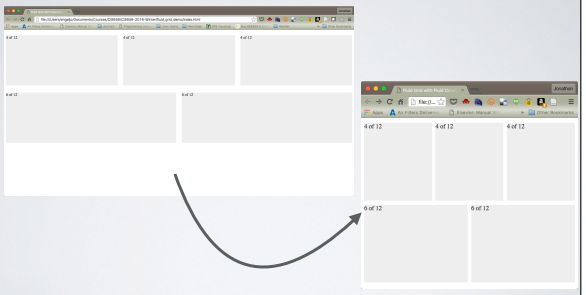
GRID LAYOUT TAKEAWAYS

- Much less verbose than table layouts
- Styling separate from content
- But what if screen size > 960 or screen size < 960?

FLUID GRIDS

- graceful (fluid like) support for different size screens.
- designer specifies a max container width
- layout is scaled in proportion to the container (widths specified as percentages).

EXAMPLE: FLUID GRID



Example coded with: http://www.responsivegridsystem.com/calculator/

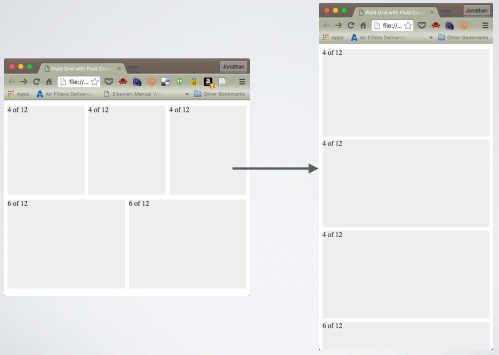
EXAMPLE: FLUID GRID

```
<div class="section group">
    <div class="col span_4_of_12">
    4 of 12
    </div>
    <div class="col span_4_of_12">
    4 of 12
    </div>
    <div class="col span_4_of_12">
    4 of 12
    </div>
</div>
<div class="section group">
    <div class="col span_6_of_12">
    6 of 12
    </div>
    <div class="col span_6_of_12">
    6 of 12
    </div>
</div>
```

- What happens if content cannot be scaled any further?

Example coded with: http://www.responsivegridsystem.com/calculator/

RESPONSIVE DESIGN



THE BIG LAYOUT CHALLENGE

Smartphones
Tablets

Computers

Big Screens

Small — Medium — Large

How can we develop a single web app and have it look good on every possible screen?

What are our options for getting content onto mobile devices and tablets? Native app ecosystems are hugely popular. Do we even need to use the web to deliver content to smartphones and tablets?

MOBILE APP DEVELOPMENT

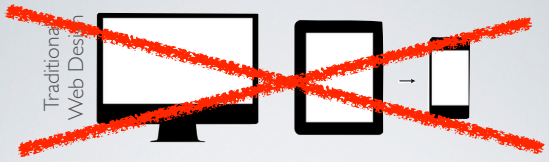
Mobile Web Apps	"Hybrid" Apps	Native Apps
HTML5	NimbleKit	Android SDK
Sencha Touch	PhoneGap	iOS / xCode
CSS3	Titanium Mobile	Visual Studio
jQuery Mobile	Rhodes	WinPhone
Javascript		
Web app targeting handheld devices	Framework that generates native code and/or uses webviews in native.	Apps written for the native platform.

GO NATIVE OR NOT??

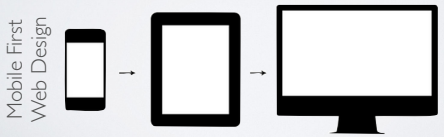
- Gartner's "Rule of Three": use Mobile Enterprise Application Platform (MEAP) when:
  - there are 3 or more mobile applications
  - there are 3 or more targeted operating systems or platforms
  - they involve the integration of 3 or more back-end systems

ENGELSMA'S TAKE

- Economics are important, but be user centric (as budget allows):
  - If possible, go native for consumer or customer facing apps
  - Hybrid / Web approach should be considered when you have a "captive audience", e.g. employee-only app.
  - Web vs. Hybrid: "app store" distribution is most familiar to end users today.



Definition: Mobile First - when designing a new web app, first begin with the mobile design and then work up to a larger desktop version.



MOBILE FIRST

- Why?
  - Mobile web is very important:
    - 1.2B worldwide users in 2013.
    - 25% of users in USA are mobile-only.
    - 85% of new handset sales have web access.
  - Why not?
    - Designing for all screens is difficult!

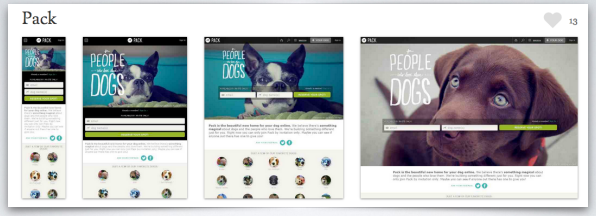
Source: http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats

RESPONSIVE WEB DESIGN

- a design approach aimed at crafting sites to provide an optimal viewing experience - easy reading and navigation with minimum resizing, panning, and scrolling, across a wide range of devices, from small screens to large screens.

Source: http://en.wikipedia.org/wiki/Responsive_Web_Design

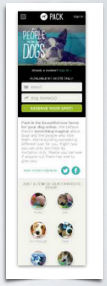
RESPONSIVE WEB DESIGN



Source: http://mediaqueri.es/

KEY COMPONENTS OF RESPONSIVE WEB DESIGN

- Fluid grid (relative sizing)
- Flexible images (see http://alistapart.com/article/fluid-images)
- Media queries to switch out styling rules based on viewing device.



MEDIA QUERIES (1)

- CSS 2.1 introduced a concept called *media types*:

```
<link rel="stylesheet" type="text/css" href="style.css"
      media="screen" />
<link rel="stylesheet" type="text/css" href="print.css"
    media="print" />
```

- Other media types included "braille", "speech", "handheld" "tv" and more.
- Most were never implemented by browsers!

Reference: http://www.w3.org/TR/CSS21/media.html#media-types

MEDIA QUERIES (2)

- CSS3 introduced a concept called *media queries*:
- Finer control, e.g., target not only the device class, but its physical characteristics.

Reference: http://www.w3.org/TR/css3-mediaqueries/#media1

MEDIA QUERIES (3)

- Media Query example:

```
<link rel="stylesheet" type="text/css"
    media="screen and (max-device-width: 480px)"
    href="mobile.css" />
```

- Media query contains 2 items:
  - a media type (in this case "screen")
  - a query referring to a media feature, in this case max-device-width and the value 480px.

MEDIA QUERIES (4)

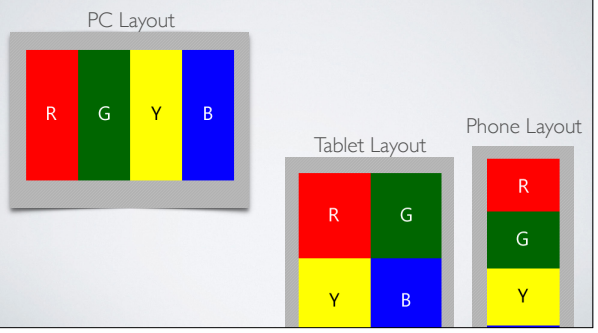
- Multiple device features can be chained in a single media query:

```
<link rel="stylesheet" type="text/css"
    media="screen and (max-device-width: 480px) and (resolution: 163dpi)"
    href="mobile.css" />
```

- Media queries can also be embedded right in the stylesheet:

```
@media screen and (max-device-width: 480px) {
    .sidebar {
        float: none;
    }
}
```

SIMPLE RESPONSIVE EXAMPLE



SIMPLE RESPONSIVE EXAMPLE (2)

```
@media screen
    and (min-width:800px)
{
  /*custom styling for screens
    greater than 800px. */

  #red, #yellow, #green, #blue {
    width: 25%;
    display: inline-block;
    white-space: nowrap;
  }

}
```

If screen width is > 800px

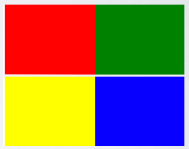
The content:

```
<div id="mainContent">
  <div id="red"></div><div id="green"></div><div id="yellow"></div><div id="blue"></div>
</div>
```

SIMPLE RESPONSIVE EXAMPLE (3)

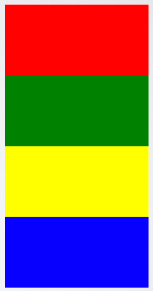
- If 540px <= width <= 800px we get 4 X 4 layout:

```
@media screen
    and (max-width:800px)
    and (min-width:540px)
{
  /* this is the breakpoint where
     we transition from the first
     layout, of four side-by-side
     columns, to the square layout
     with 2X2 grid */

  #red, #blue, #green, #yellow {
    width:50%;
    display: inline-block;
  }
}
```



MEDIA QUERIES (4)

- if width < 540px, neither of the media queries match, so blocks are laid out in the room available (vertically)



RESPONSIVE FRAMEWORKS

- The techniques discussed are widely implemented in popular CSS frameworks. Some examples:
  - Bootstrap: http://getbootstrap.com/
  - Pure: http://purecss.io/
  - Foundation: http://foundation.zurb.com/
  - HTML5 BoilerPlate: http://html5boilerplate.com/

CONCLUSIONS

- Huge progress has been made in efficiently creating highly styled and maintainable web apps.
- There's no excuse for having a website that doesn't look decent on every screen!