One-day Responsive Web Design : Build Examples

a_pixel_em

Set base line for calculations.

```
html {
    font-size: 100%;    /* Default 16 pixels */
}

body {
    font-size: 62.5%;   /* 1 EM = 10 pixels */
}
```

Set EMs based on containing element.

```
h1 {
    font-size: 2em;                 /* 20/10 *

h1 a {
    font-size: 0.8em;               /* 16/20 */

p {
    font-size: 1.4em;               /* 14/10 */

p span {
    font-size: 1.14285714285714em;  /* 16/14 */
```

- **jQuery** : why code at end of HTML?
- What does stopPropagation() method do?

b_margins_padding

- Can narrow browser beyond 400px with Firefox.
- How to make four nested DIVs remain **proportionate** in liquid layout.
- **Margin** is calculated based on parent/containing element.
- **Padding** is calculated based on element itself.

- But **Borders** cannot used %, and do not cascade.

```
web{
    width:  80%;        /* 400/500 = 0.8 */
    padding: 2.5%;      /* 10/400 = 0.025 */
    margin: 2%;         /* 10/500 = 0.02 */
    border: solid 1px black;
}

design {
    width:  62.5%;      /* 250/400 = 0.625 */
    padding: 4%;        /* 10/250 = 0.04 */
    margin: 2.5%;       /* 10/400 = 0.025 */
    border: solid 1px black;
```

c_fixed_fluid

- Convert a **fixed grid**.
- Understand 5 columns and gutters: (5 * 160) + (10*20) = 1000
- Convert widths to percentages
- Convert margins to percentages.
- Wrapper : change width to 100% and add max-width statement.

The column layout breaks. It exceeds 100% using default **box-sizing:content-box** model.

On IE8+, setting box-sizing equal to border-box, allows BORDERS and PADDING to be added, without breaking a percentage-based responsive column layout.

[Question] what are vendor prefixes?

- Add **box-sizing** with vendor prefixes.
- Add **padding** and increase **border**

- If we add image, need new rule to resize within its container:

```
img {
    max-width: 100%;
    border: 1px solid black;
```

```
}
```

- Style the **wrapper** div with a border,background color,margin,padding.
- The wrapper div **collapses** because it only contains floated items. Add a **clearFix** class to the wrapper DIV in the HTML.
- Does not include any **breakpoints** once text in columns becomes too narrow.

d_antarctic_grid

- Create a **responsive grid** using percentages and floats.
- Define float-based columns using **.grid-wrap** and **.grid-col** classes.
- Introduce mobile and tablet **breakpoints**
- **Naming convention** used combines name of breakpoint, and % width of column
- Apply **multiple classes** from multiple breakpoints to each column
- Add **debugging** using before/content CSS.
- Note: **Modernizr** and **YepNope** are used to provide a polyfill for older browsers that do not support media queries.
- Breakpoints are Brad Frost **-ISH** i.e. do not map directly onto known iPhone/iPad widths.

```
<div class="grid-col mobile-50 tablet-60">
<div class="grid-col mobile-50 tablet-40">

<div class="grid-col mobile-50 tablet-25">
<div class="grid-col mobile-50 tablet-25">
<div class="grid-col tablet-25">
<div class="grid-col tablet-25">
```

e_sentence

In **HTML**, Markup a multi-line heading with **spans**.

```
<h1>The Bank of England
    <span id="centralbank">(the UK central bank)</span>
```

```
    base rate
    <span id="mortgage">(used for mortgage and savings rates)
    <span id="mpc">, set by the MPC, </span>
    is 0.5%
    <span id="history">, and has been since 2009.</span>
</h1>
```

In **CSS**, hide all the spans, and then selectively reveal using media queries.

```
#centralbank , #mortgage, #mpc, #history {
    display: none;
}

@media screen and (min-width: 600px ) {
    #centralbank {
        display: inline;
        color: #8DD25A;
    }
}
```

f_table

- **Do nothing**. Table is recognisable, but awkward pinch/zoom.
- Could consider **dropping non-essential columns** for certain widths. But this breaks rule of serving all content to all users.

  ```
  @media screen and (max-width: 700px) {

      tr td:nth-child(5),
      tr th:nth-child(5) {
          display:none;
          visibility:hidden;
      }

  }
  ```

- **Scrollable tables** use a combination of a single left fixed column, and scrollable right columns. But scrollbars may not be visible on some devices.

- Turn each row into multiple columns. Note: nth-of_type() uses 1-indexing (IE9+)

```
  @media screen and (max-width: 700px) {

      table, thead, tbody, th, td, tr {
          display: block;
      }

      table th {
          display: none;
      }

      tr{
          border-bottom: 0.5em solid #FFFFFF;
      }

      td:nth-of-type(1) {
          background-color: rgba( 120,222,156,0.2);
      }

      td:nth-of-type(1):before { content: "Author";}
      td:nth-of-type(2):before { content: "Title"; }
      td:nth-of-type(3):before { content: "Price";}
      td:nth-of-type(4):before { content: "Publisher"; }
      td:nth-of-type(5):before { content: "Reviewer"; }

      td:before {
          padding-right: 1em;
          text-transform: uppercase;
          color: rgba( 120,222,156,1);
      }
  }
```

g_table_custom

In **HTML**, construct a 1 row data table.

```
<tr>
    <td>UK male life expectancy, decades 1910-80.</td>
```

```
    <td class="data">58</td>
    <td class="data">61</td>
    <td id="sparkline"><span>58,61,62,65,68,70,74,78</span></
</tr>
```

Load **jQuery** and the **PIETY** jQuery plug-in, to convert spans of numbers to canvas charts.

In **Javascript**, create a **matchMedia** object, and listen for changes to its associated media query.

```
var mq = window.matchMedia("screen and (min-width: 40em)");
mq.addListener( monitorMQ );
```

Hide and reveal table rows, based on screen width:

```
var tablet = mq.matches;

$("#sparkline").toggle( !tablet );
$(".data").toggle( tablet );
```

Create a bar chart canvas, when the screen is narrow:

```
var spark = $("#sparkline").find("span");
spark.peity( "bar" , {  colours: "black" } );
```

h_image_pf

PictureFill is a POLYFILL, which uses DIVs and HTML5 data-attributes to simulate the semantics of the future **picture** element.

Add CSS to make image resize to its container (body) element.

```
    <style>
        img {
            max-width: 100%
        }
    </style>
```

Construct a DIV that defines **breakpoints** and a **fallback** case for browsers, with JS turned off.

```
<div data-picture data-alt="desert">

    <div    data-src="imgs/small.jpeg"></div>

    <div    data-src="imgs/medium.jpeg"
            data-media="(min-width: 500px)"></div>

    <!-- Fallback -->
    <noscript>
        <img src="imgs/fallback.jpeg" alt="desert..">
    </noscript>
</div>
```

i_navigation

- Create responsive hide/reveal menu using CSS and jQuery.
- Review the image in DOCS.

```
img {
    max-width: 100%;
```

- Understand this. The menu is initially hidden

```
ul {
    display:none;
}
ul.reveal {
    display: block;
}
```

- jQuery **toggleClass** will add/remove a class to this element.

```
$("#menu").on( "click", function() {
    $("ul").toggleClass( "reveal");
```

- This works, but menu links unstyled. Unsuitable for **touch** interface on a

phone. Style menu.

```
ul li {
    background-color: rgba(112,64,64,0.4);
    border-radius: 2px;
    margin-bottom: 4px;
    padding: 0.75em;
    text-align: center;
    text-transform: uppercase;
}

ul li a {
    color: rgba(112,64,64,1);
    letter-spacing: 0.2em;
}
```

- Add a **media query** to hide MENU button, turn navigation back into horizontal list, float navigation, turn sections into 2 columns.

j_ajax

[Class exercise] Test out **matchMedia** expressions.

Note: **Chrome** does not run AJAX from the desktop. On PC, open it with conditional flags at command line **--allow-file-access-from-files**

On MAC, could use the installed Python web server:

```
python -m SimpleHTTPServer 1200
http://localhost:1200/pathname
```

Index.html contains an empty DIV, into which the HTML of another file will be loaded using AJAX.

```
<div id="extra"></div>
```

In Javascript, listen for **window resize** events (or use matchMedia approach)

```
window.addEventListener( "resize" , ajaxLoad );
```

When window expands beyond certain width, load content from extra HTML file into empty DIV:

```
if ( document.body.clientWidth >= breakPoint )
    $("#extra").load( "extra.html" );
```