Toplines from Responsive Web Design, by Ethan Marcotte

* In terms of front end work, there are three core ingredients to responsive web design:
    1. A flexible, grid-based layout
    2. Flexible images and media
    3. Media queries, a module from the CSS3 specification"

FLEXIBLE GRID:

* To create a typographic grid:
    - Implement a reset stylesheet, like those Eric Meyer uses, to get to a uniform starting point in terms of fonts, spacing, etc.
    - Set your body font-size to 100%. This will establish a baseline font size for all text.
    - All font size effects will then be a product, in ems, of the following formula:

        desired result (in ems) = target (in px) / context (in px)

    - Context is the element your target element inherits font attributes from.
    - Don't be tempted to unduly round off percentages--the browser can do the math, and the answer with a high degree of precision maintains proportionality best.

* To create a flexible layout:
    - Be prepared to take a pixel-specific layout from your comps and translate it into a set of proportional measurements in percentages.
    - Set a width of your outermost container against the body, then set internal containers using the formula

        desired result (in percent) = target (in px) / context (in px)

    - Set margins/padding proportionally to create flexible guttering.
    - When calculating, bear in mind context is different for padding/margins than width:
        1. When setting margins, context is the width of the element's container
        2. When setting padding, context is the width of the element itself
    - It is reasonable to set negative proportional margins to pull an element outside of its parent.

FLEXIBLE MEDIA

* To create flexible images:
    - To constrain fixed width media to the width of its container, use the following:

        ```
        img, embed, object, video { max-width: 100%; }
        ```

    - In IE6 and below, use (note this works poorly for small images):

        ```
        img, embed, object, video { width: 100%; }
        ```

    - In IE7 and below, images scaled with CSS render with artifacts. Use AlphaImageLoader (proprietary MSoft CSS extension) to correct for it

* To create flexibly tiled backgrounds for multi-column layout:
    - Create an extremely wide (several k px) background image that is relatively short
    - Using your layout comp, figure out the breakpoints where your background colors change, and then determine what percentage each column color represents of the whole background.
    - Find the pixel widths necessary to match the proportions in your very large background image, and paint backgrounds to those proportions. You will end up with a very wide, but proportionally correct, set of color backgrounds.
    - Set the very large background image on the element which contains the columns, so that it appears behind them. It will proportionally stretch at the same rate as the content it appears behind.
    - Other options include CSS3 'background-size' property (support's not great), jQuery Backstretch plugin, and using media queries to show different backgrounds at different widths.

* To create background images that do not rescale, but progressively appear at larger sizes of their containing element:

- Set the parent element to overflow: hidden
- Set the image to display: block and max-width: auto

MEDIA QUERIES

* To select style rules based on the capabilities / characteristics of the device viewing your content, use media queries.

* Media queries are made of two components:
  1. A media type, drawn from this list:
     all, braille, embossed, handheld, print, projection, screen, speech, tty, tv
  2. In parentheses, the query itself, which will be made of one or more features and values, separated by logical operators

* Media queries can be used in three basic ways:
  1. As a block declaration within a stylesheet:

```
@media screen and (min-width: 1024px) {
    body { font-size: 100%; }
}
```

  2. As a media argument to a <link> tag:

```
<link rel=stylesheet" href="wide.css" media="screen and (min-width: 1024px)" />
```

  3. As an argument to an @import statement:

```
@import url('wide.css') screen and (min-width: 1024px);
```

* Important to understand that:
  1. Each device has a 'display area' (the browser's viewport) and a 'rendering surface' (the entire display of the device)
  2. To test values above or below thresholds, many features accept min- and max- prefixes

* The testable features:

Feature	Definition	min-/max-
width	Width of the display area.	Y
height	Height of the display area.	Y
device-width	Width of the rendering surface.	Y
device-height	Height of the rendering surface.	Y
orientation	Accepts 'portrait' or 'landscape'	N
aspect-ratio	Ratio of display area's width/height.	Y
device-aspect-ratio	Ratio of rendering surface's width/height.	Y
color	Number of bits per color component of device.	Y
color-index	Number of entries in color lookup table for device.	Y
monochrome	Number of bits per pixel in a monochrome device.	Y
resolution	Density of the pixels in the device, in dpi.	Y
scan	For tv, measures 'progressive' or 'scan'.	N
grid	Tests whether device is a grid based display.	N

* Not all features are supported on all devices / browsers--research before using them.

* In 2007, Apple introduced the <meta> tag value 'viewport', which allows you to manipulate the size of the canvas a mobile browser uses to render web content to screen. For most purposes, it is reasonable to simply assert the following:

```
<meta name="viewport" content="initial-scale=1.0, width=device-width" />
```

* That declaration will make the width of the browser's viewport equal to the width of the device's screen, so an iPhone, for instance, won't lay out content at 980px wide, but instead 320px in portrait mode and 480px in landscape mode.

* If you set the viewport to width=device-width, you are more able to use max-width and min-width queries to look for the resolution range that a user is browsing at, and style your content accordingly. It also makes your content more likely to render

coherently across multiple mobile devices / screen sizes.

* Note that media queries are widely supported, but do NOT have support in IE8 or below.
  However, you can shim/patch that behavior with javascript in many cases.