Responsive Web Design (RWD) - Fluid Grids and Layout

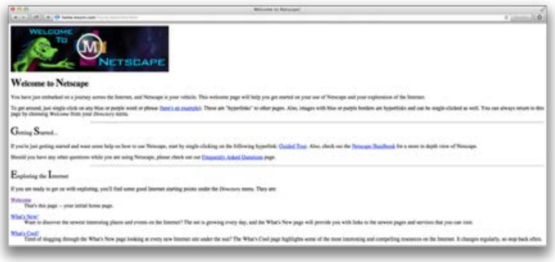
You don’t know what size the viewers screen is going to be. To be an excellent web designer your layout must respond to different window sizes and different devices.

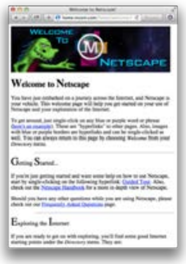
Don’t ever say, “I want the whole layout to fit on one page.” It tells the world you have not embraced RWD.

How did we get here?

The original Responsive Web Layout.

In the old days of the web, everything was responsive because every site just filled the width of the browser window.

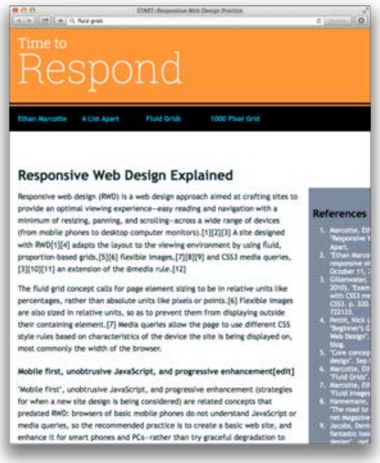




Old school sites were not pretty, but they fit in almost any sized monitor.

Grids

Unsatisfied with one wide column, web designers came up with <div>s and CSS layouts. The 960px grid was born—a handy tool helping designers create column grids that fit neatly in a 1024x768px computer screen. Designers pretended that the whole world viewed the web on a monitor at least 1024 pixels wide.



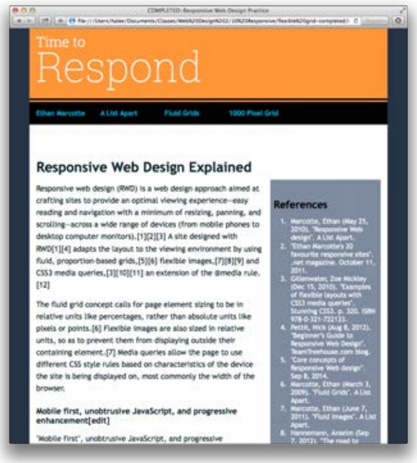
Responsive Web Design Explained

Responsive web design (RWD) is a web design approach aimed at crafting sites to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors).[1][2][3] A site designed with RWD[1][4] adapts the layout to the viewing environment by using fluid, proportion-based grids,[5][6] flexible images,[7][8][9] and CSS3 media queries,[3][10][11] an extension of the @media rule.[12]

The fluid grid concept calls for page element sizing to be in relative units like percentages, rather than absolute units like pixels or points.[6] Flexible images are also sized in relative units, so as to prevent them from displaying outside their containing element.[7] Media queries allow the page to use different CSS style rules based on characteristics of the device the site is being displayed on, most commonly the width of the browser.

Mobile first, unobtrusive JavaScript, and progressive enhancement[edit]

"Mobile first", unobtrusive JavaScript, and progressive enhancement (strategies for when a new site design is being considered) are related concepts that predated RWD: browsers of basic mobile phones do not understand JavaScript or media queries, so the recommended practice is to create a basic web site, and enhance it for smart phones and PCs—rather than try graceful degradation to



Left: If you view a static site on a monitor smaller than 1024 pixels wide, you may have to scroll sideways to see all the content.

Right: Fluid grids allow the layout to re-size to fit in any sized browser, or device.

Responsive Web Design is Born

Shortly after the birth of the 960px grid, the web gods admitted that not everyone was viewing the web through the same size monitor. Viewports stretched from about two inches to 27 inches and more. We needed to design sites that adapt to different sized devices. In 2010, a web genius named Ethan Marcotte coined the term “Responsive Web Design” in an article in the blog *A List Apart* (http://alistapart.com/article/responsive-web-design). Designers who embrace responsive web design (RWD) create layouts that present readable content in any sized viewport.

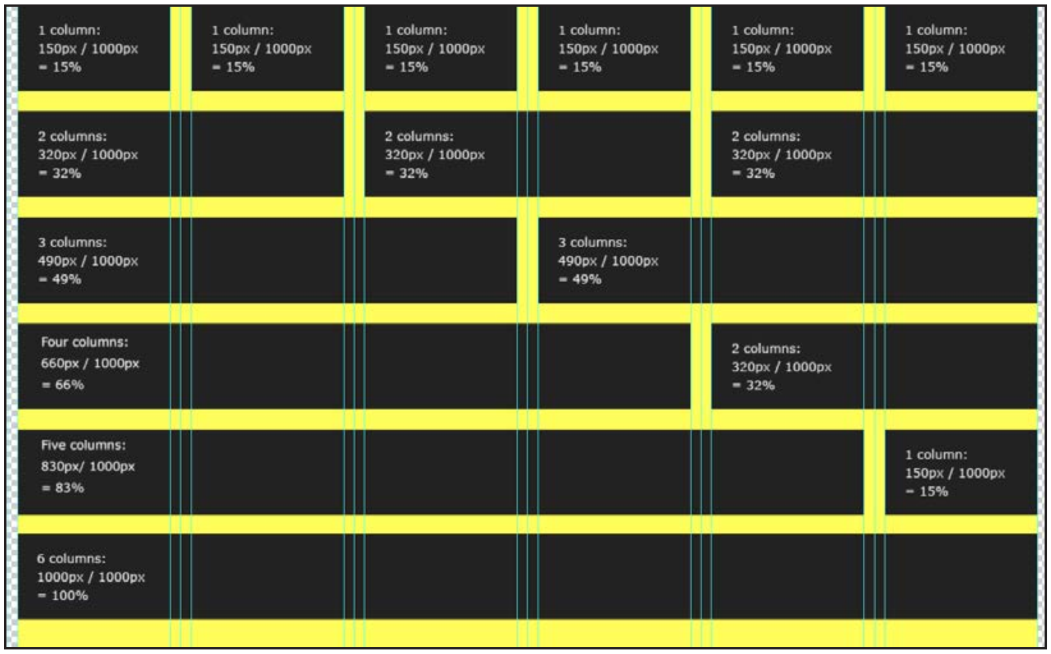
Fluid Grids

The first step toward responsive design, based on the 960px grid, is embracing relative units and converting our beloved pixel into to percentages and ems.

It is very easy (not). You just divide the widths of your columns by the width of the whole layout. For example a 300px wide column would be divided by the 960px width of the layout: 300px ÷ 960px = 0.3225 x 100 = 32.25%. Easy, right? No.

The 1000 Pixel Grid — Saving You From Difficult Math

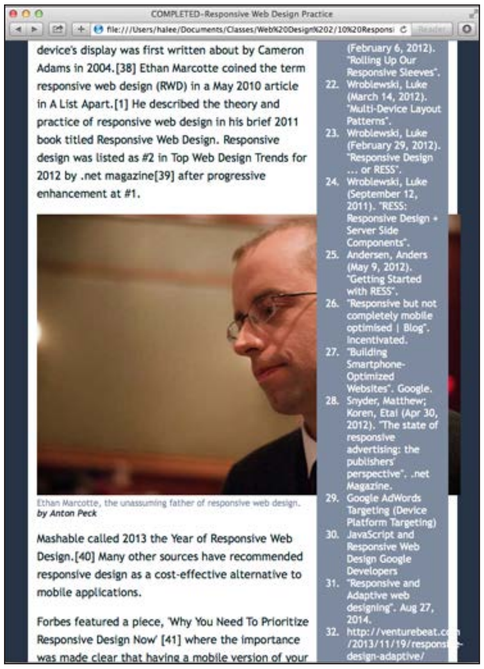
Lately, another web guru came up with the 1000 pixel grid (http://www.elliotjaystocks.com/blog/a-better-photoshop-grid-for-responsive-web-design/). Designing on the 1000 pixel grid allows for easier math. If you have a column that is 320px wide, all you need to do is take the zero off the width of the column and add a percent symbol; 32%. Use the ejs-1000-pixel-grid.psd file when planning your layout and the math is done for you.


1 column: 150px / 1000px = 15%
1 column: 150px / 1000px = 15%
1 column: 150px / 1000px = 15%
1 column: 150px / 1000px = 15%
1 column: 150px / 1000px = 15%
1 column: 150px / 1000px = 15%
2 columns: 320px / 1000px = 32%
2 columns: 320px / 1000px = 32%
2 columns: 320px / 1000px = 32%
3 columns: 490px / 1000px = 49%
3 columns: 490px / 1000px = 49%
Four columns: 660px / 1000px = 66%
2 columns: 320px / 1000px = 32%
Five columns: 830px/ 1000px = 83%
1 column: 150px / 1000px = 15%
6 columns: 1000px / 1000px = 100%

Above: The 1000 pixel grid template helps you make fluid layouts without complex math.
If using the supplied .psd template, use the first two columns to put together your mobile first design, and all 6 columns to put together your desktop design.

How Do You Make a Fluid Image?

As the browser window becomes smaller, columns may become narrower than the images in them.



COMPLETED–Responsive Web Design Practice

file:///Users/halee/Documents/Classes/Web%20Design%202/10%20Responsi

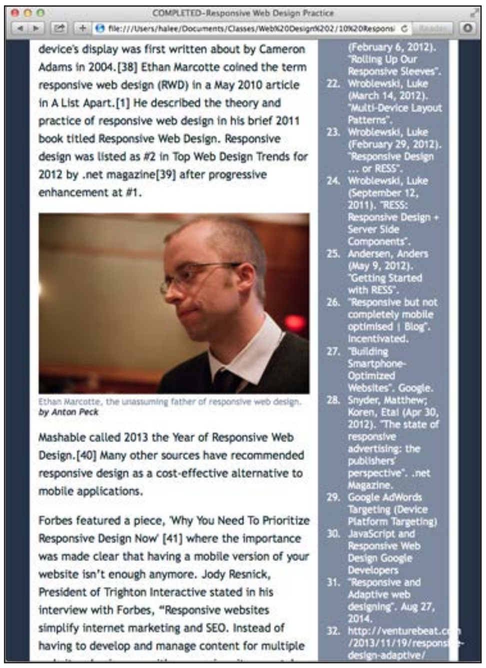
device's display was first written about by Cameron Adams in 2004.[38] Ethan Marcotte coined the term responsive web design (RWD) in a May 2010 article in A List Apart.[1] He described the theory and practice of responsive web design in his brief 2011 book titled Responsive Web Design. Responsive design was listed as #2 in Top Web Design Trends for 2012 by .net magazine[39] after progressive enhancement at #1.

Ethan Marcotte, the unassuming father of responsive web design.
by Anton Peck

Mashable called 2013 the Year of Responsive Web Design.[40] Many other sources have recommended responsive design as a cost-effective alternative to mobile applications.

Forbes featured a piece, 'Why You Need To Prioritize Responsive Design Now' [41] where the importance was made clear that having a mobile version of your

(February 6, 2012). "Rolling Up Our Responsive Sleeves".
22. Wroblewski, Luke (March 14, 2012). "Multi-Device Layout Patterns".
23. Wroblewski, Luke (February 29, 2012). "Responsive Design ... or RESS".
24. Wroblewski, Luke (September 12, 2011). "RESS: Responsive Design + Server Side Components".
25. Andersen, Anders (May 9, 2012). "Getting Started with RESS".
26. "Responsive but not completely mobile optimised | Blog". Incentivated.
27. "Building Smartphone-Optimized Websites". Google.
28. Snyder, Matthew; Koren, Etai (Apr 30, 2012). "The state of responsive advertising: the publishers' perspective". .net Magazine.
29. Google AdWords Targeting (Device Platform Targeting)
30. JavaScript and Responsive Web Design Google Developers
31. "Responsive and Adaptive web designing". Aug 27, 2014.
32. http://venturebeat.c /2013/11/19/respons -design-adaptive/



COMPLETED–Responsive Web Design Practice

file:///Users/halee/Documents/Classes/Web%20Design%202/10%20Responsi

device's display was first written about by Cameron Adams in 2004.[38] Ethan Marcotte coined the term responsive web design (RWD) in a May 2010 article in A List Apart.[1] He described the theory and practice of responsive web design in his brief 2011 book titled Responsive Web Design. Responsive design was listed as #2 in Top Web Design Trends for 2012 by .net magazine[39] after progressive enhancement at #1.

Ethan Marcotte, the unassuming father of responsive web design.
by Anton Peck

Mashable called 2013 the Year of Responsive Web Design.[40] Many other sources have recommended responsive design as a cost-effective alternative to mobile applications.

Forbes featured a piece, 'Why You Need To Prioritize Responsive Design Now' [41] where the importance was made clear that having a mobile version of your website isn't enough anymore. Jody Resnick, President of Trighton Interactive stated in his interview with Forbes, "Responsive websites simplify internet marketing and SEO. Instead of having to develop and manage content for multiple

(February 6, 2012). "Rolling Up Our Responsive Sleeves".
22. Wroblewski, Luke (March 14, 2012). "Multi-Device Layout Patterns".
23. Wroblewski, Luke (February 29, 2012). "Responsive Design ... or RESS".
24. Wroblewski, Luke (September 12, 2011). "RESS: Responsive Design + Server Side Components".
25. Andersen, Anders (May 9, 2012). "Getting Started with RESS".
26. "Responsive but not completely mobile optimised | Blog". Incentivated.
27. "Building Smartphone-Optimized Websites". Google.
28. Snyder, Matthew; Koren, Etai (Apr 30, 2012). "The state of responsive advertising: the publishers' perspective". .net Magazine.
29. Google AdWords Targeting (Device Platform Targeting)
30. JavaScript and Responsive Web Design Google Developers
31. "Responsive and Adaptive web designing". Aug 27, 2014.
32. http://venturebeat.c /2013/11/19/respons -design-adaptive/

Left: In this image the photo is not playing along with the flexible grid, but is breaking out of its containing element.
Right: With the addition of the magical max-width property, photos will never be wider than the div that contains them.

To keep photos from breaking out of their columns, add the following code to your CSS:

```
img {
    max-width: 100%;
    height: auto;
    }
```

The 100% max-width tells the browser to re-size images so they are never wider than the element that contains them. It also prevents them from being "up-scaled" larger than their intended size

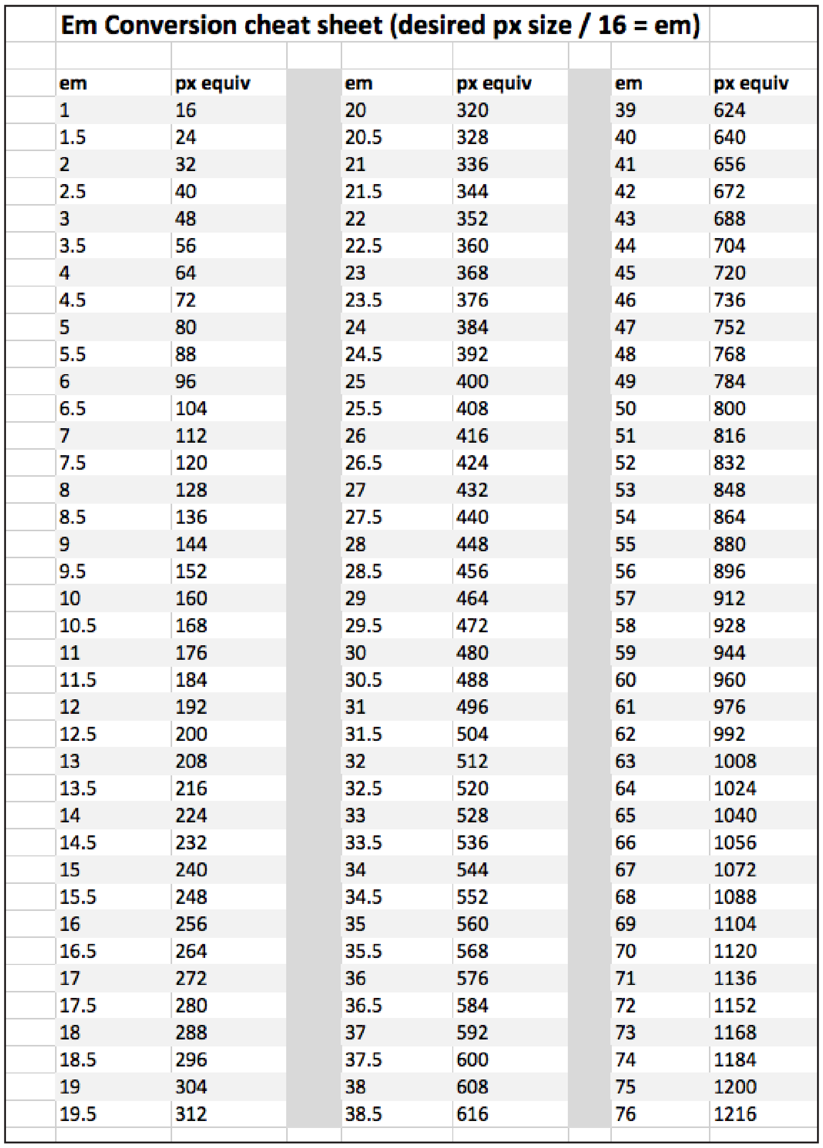
How do you make Fluid Text?

When it comes to sizing your text, think in terms of the relative unit, ems.

- 16px is the default font size for most browsers.
- 1 em is equal to 16px.

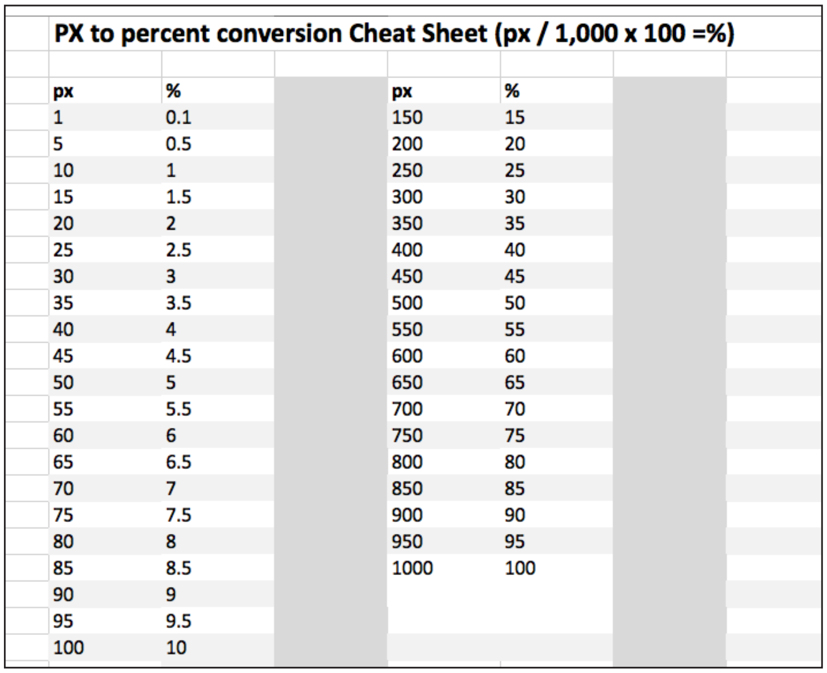
Why ems?

As users scale their sites, the text will scale with it, proportionate to the rest of the content.



Em Conversion cheat sheet (desired px size / 16 = em)

em	px equiv	em	px equiv	em	px equiv
1	16	20	320	39	624
1.5	24	20.5	328	40	640
2	32	21	336	41	656
2.5	40	21.5	344	42	672
3	48	22	352	43	688
3.5	56	22.5	360	44	704
4	64	23	368	45	720
4.5	72	23.5	376	46	736
5	80	24	384	47	752
5.5	88	24.5	392	48	768
6	96	25	400	49	784
6.5	104	25.5	408	50	800
7	112	26	416	51	816
7.5	120	26.5	424	52	832
8	128	27	432	53	848
8.5	136	27.5	440	54	864
9	144	28	448	55	880
9.5	152	28.5	456	56	896
10	160	29	464	57	912
10.5	168	29.5	472	58	928
11	176	30	480	59	944
11.5	184	30.5	488	60	960
12	192	31	496	61	976
12.5	200	31.5	504	62	992
13	208	32	512	63	1008
13.5	216	32.5	520	64	1024
14	224	33	528	65	1040
14.5	232	33.5	536	66	1056
15	240	34	544	67	1072
15.5	248	34.5	552	68	1088
16	256	35	560	69	1104
16.5	264	35.5	568	70	1120
17	272	36	576	71	1136
17.5	280	36.5	584	72	1152
18	288	37	592	73	1168
18.5	296	37.5	600	74	1184
19	304	38	608	75	1200
19.5	312	38.5	616	76	1216

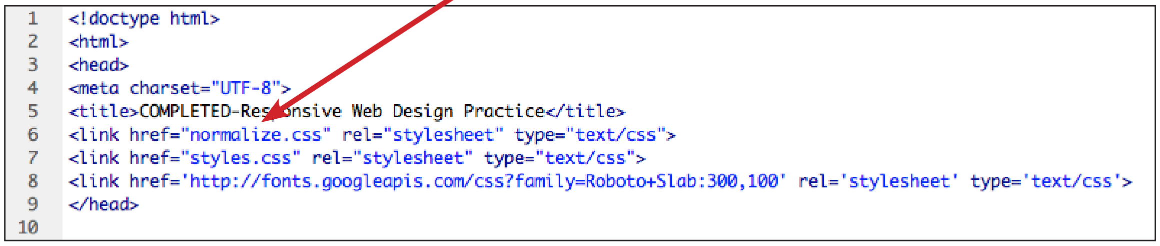


PX to percent conversion Cheat Sheet (px / 1,000 x 100 =%)

px	%	px	%
1	0.1	150	15
5	0.5	200	20
10	1	250	25
15	1.5	300	30
20	2	350	35
25	2.5	400	40
30	3	450	45
35	3.5	500	50
40	4	550	55
45	4.5	600	60
50	5	650	65
55	5.5	700	70
60	6	750	75
65	6.5	800	80
70	7	850	85
75	7.5	900	90
80	8	950	95
85	8.5	1000	100
90	9		
95	9.5		
100	10		

What is normalize.css?

You may notice there is a style sheet named "normalize.css" linked to the practice html document. Normalize.css is a CSS reset.



```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>COMPLETED-Responsive Web Design Practice</title>
<link href="normalize.css" rel="stylesheet" type="text/css">
<link href="styles.css" rel="stylesheet" type="text/css">
<link href='http://fonts.googleapis.com/css?family=Roboto+Slab:300,100' rel='stylesheet' type='text/css'>
</head>

```

Above: The link to the normalize.css should come before the link to your styles.css. Whatever is lower in the cascade, will overwrite whatever is higher. So remove browser defaults with normalize.css, and add styling back in, in your own styles.css.

The reason your site looks a little different in different browsers is because each browser has a different default style sheet. A true CSS reset removes all padding, margin and formatting from every HTML element. Then you, the designer, must define every attribute of every element in your styles.css. Defining every attribute makes your site render more consistently in every browser.

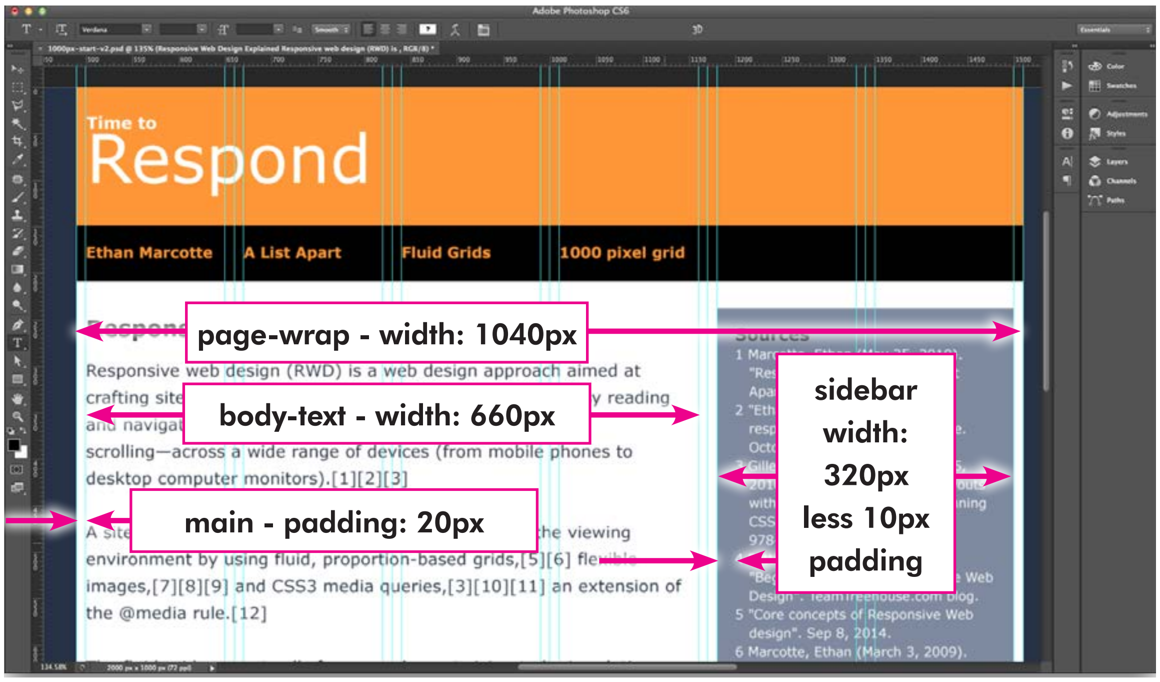
The practice site coming up uses the Normalize reset, which is a little different. Normalize resets some styles and then adds some formatting back in. Your site renders consistently in most browsers, but you don't have to define every single attribute in your styles.css. According to the Normalize web site (http://nicolasgallagher.com/about-normalize-css/): "Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing."

In your HTML code, your styles.css link must come after the normalize.css link, because style sheets lower in the list will override style sheets higher in the list.

So from now on, let's add this normalize.css file to our root folders, and make sure it's linked to all of our pages.

Activity

Get the “Fluid Grids Start” folder from the Shared Items server. Convert the following layout from a fixed-width, pixel based layout, to a percentage based, fluid layout. As you work re-size the window to see your site responding to different viewport widths.



Here is an example equation converting the .body-text selector properties from px to %.

660px ÷ 1000px =__________ x 100% = __________%

As you work on this activity, use the formula above to convert all the pixel width measurements in the CSS to percentages. Or figure the problems in your head.

More Responsiveness to Come

Next time, we will explore how to change the layout and type size for the different window sizes and different devices using something called in-line media queries.

Portage Trail Project Requirement: You need to keep your design flexible by defining the widths of your divs, articles, sections, asides and other containing elements in percentages rather than pixels. You’ll also need to use ems in your type styles.

Responsive Web Design—Media Queries

Responsive Web Design (RWD) Review

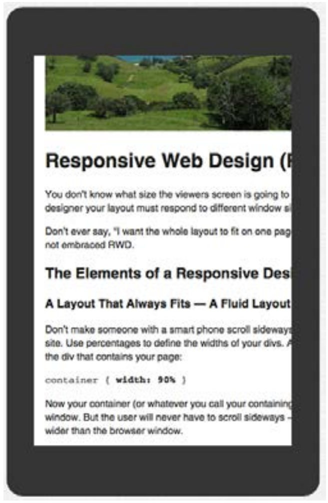
A responsive web design is one that adapts, or responds, to any sized device. Your site adapts to the user, your user doesn't adapt to your site.

Fluid layout—A Layout That Always Fits

Use percentages to define the widths of your layout elements. For example, you can apply this rule to the div that contains your page:

```
.page-wrap { width: 90% }
```

Now your page-wrap (or whatever you call your containing div) will fill the browser window. But the user will never have to scroll sideways — the container will never be wider than the browser window.



Responsive Web Design (F

You don't know what size the viewers screen is going to
designer your layout must respond to different window si

Don't ever say, "I want the whole layout to fit on one pag
not embraced RWD.

The Elements of a Responsive Desi

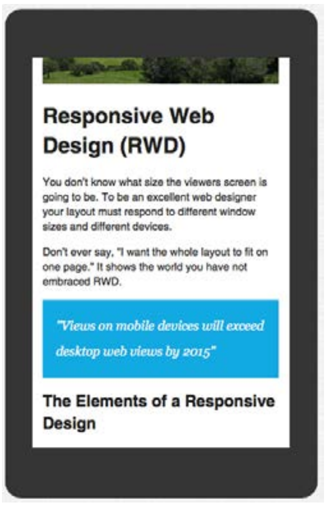
A Layout That Always Fits — A Fluid Layout

Don't make someone with a smart phone scroll sideways
site. Use percentages to define the widths of your divs. A
the div that contains your page:

```
container { width: 90% }
```

Now your container (or whatever you call your containing
window. But the user will never have to scroll sideways –
wider than the browser window.

The layout does not fit and the viewer must scroll sideways to read.



Responsive Web Design (RWD)

You don't know what size the viewers screen is going to be. To be an excellent web designer your layout must respond to different window sizes and different devices.

Don't ever say, "I want the whole layout to fit on one page." It shows the world you have not embraced RWD.

"Views on mobile devices will exceed desktop web views by 2015"

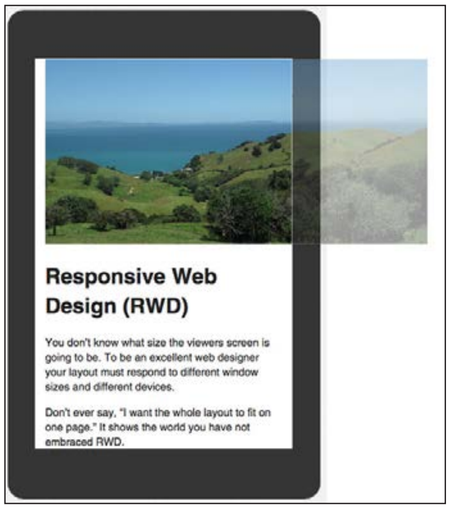
The Elements of a Responsive Design

The layout fits because all the element widths are defined in percentages.

Scalable Images

With the following code in the CSS, images will never go outside their container—they will scale down if necessary.

```
img {
   max-width: 100%;
   height: auto;
   }
```



Responsive Web Design (RWD)

You don't know what size the viewers screen is going to be. To be an excellent web designer your layout must respond to different window sizes and different devices.

Don't ever say, "I want the whole layout to fit on one page." It shows the world you have not embraced RWD.

Use max-width to keep images from busting out of their containing element

Media Queries

- Media Queries are a line of CSS that tell the browser to use different styles for different circumstances.
- Media queries can target things like screen size, device orientation (landscape or portrait), screen resolution, print, projection and many other aspects. Study them all at http://www.w3schools.com/cssref/css3_pr_mediaquery.asp if you are interested.
- We are going to concentrate our efforts on screen width, because with screen width we can style layouts that look good on both tiny phone screens and wide desktop screens.

Coding a Media Query

- Media queries can be coded in the head of the document and load different style sheets for different situations. This method causes more requests to the web server (known as http requests).
- We will be using in-line media queries to target viewport width.
- In-line media queries are located in the flow of the styles.css style sheet and do not require multiple http requests.

First, we code all the styles to make our page look good in a small mobile device. Then, below all those styles, we will type something like the following:

```
@media only screen and (min-width: 42em) {

/*styles between these curly brackets will override the properties above*/

}
```

The media query above identifies 42em as a breakpoint. “If the screen is wider than 42em, apply the styles defined between these brackets. In a media query, “min-width: 42em” means the screen, at minimum, has to be 42em wide before the styles are applied.

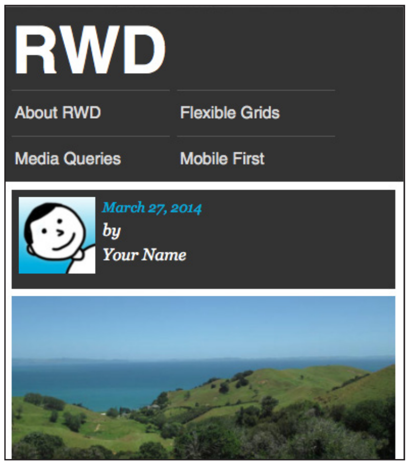
Why use ems and not pixels to define our breakpoints?

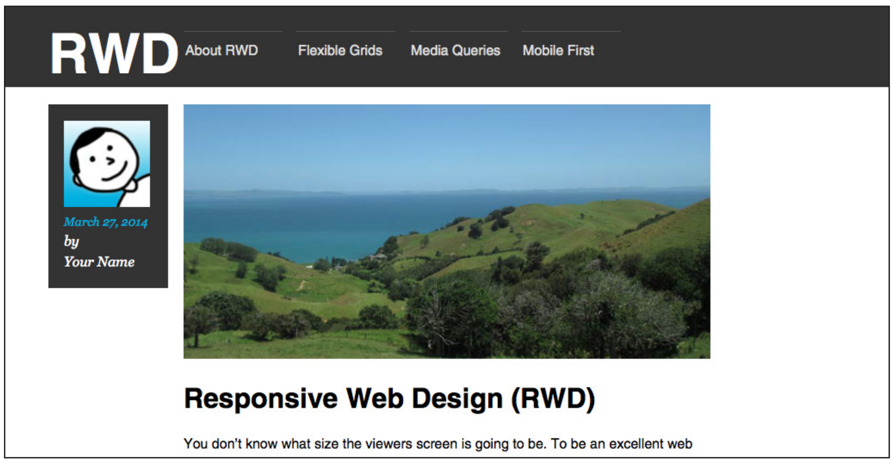
Reason 1: Content and comfortable line length readability should define the breakpoints. Using a relative size ensures that when people zoom in (aka change the default type size) the browsers will react to the media queries as expected. Fixed pixel media queries tend to break when a user zooms.

Reason 2: All the web hotshots do it, so I play along.

Common media query breakpoints to start with:

```
@media only screen and (min-width: 42em) {
          /* This will generally target a tablet */
   }
@media only screen and (min-width: 62.5em) {
          /* This will generally target a desktop*/
   }
```





Responsive Web Design (RWD)

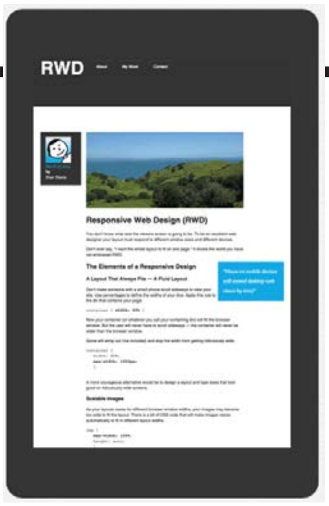
You don't know what size the viewers screen is going to be. To be an excellent web

Above right: When the screen is wide enough, the browser applies styles that float the navigation next to the logo.

Example:

```
@media only screen and (min-width: 62.5em) {
   .logo {
         float: left;
         width: 15%;
   }
   .nav-main   {
         float: left;
         width: 83%;
         padding-left: 2%;
         padding-top: 0.5em;
   }
}
```

The media query above says, "If the screen is wide enough, float the navigation next to the logo." When the screen gets wider than 62.5em the styles between the braces will override the styles defined above in the CSS.


RWD
Responsive Web Design (RWD)
The Elements of a Responsive Design

Without the "viewport" meta tag, small screens will "pinch" your layout to fit.

The Magical Meta Tag

(without which, none of this works on mobile devices)

Many mobile browsers will shrink every web layout down to fit on the screen. For instance a phone with a 320pixel wide viewport might display your page as if the viewport were 1024pixels wide. The result is a tiny layout the user must zoom in to read. If you include the following meta tag in the head of your HTML document, the viewport will not pretend to be 1024pixels wide.

The code below says, "Please acknowledge that the browser window is not huge, but is really the size of the device viewport."

```
    <head>

        ...

<meta name="viewport" content="width=device-width, initial-scale=1">

    </head>
```

Tip: Add this line of code to every page of your websites from now on.

Preview Your Site on Pixel Tuner

Upload your site to a web server, then go to http://responsive.pixeltuner.de and input your url. Pixel Tuner shows how your site will look in several different devices. The only problem with Pixel Tuner is it does not require the <meta viewport...> tag.

Test Your Site on Real Devices

Upload your responsive site to a web server and test it on several real devices. Look at it on your smart phone (or your classmate's smart phone). Go to the tablets in the Creative Center and view your site in portrait and landscape orientation. Simply shrinking the window on a desktop computer or using Pixel Tuner is not a realistic test of a mobile experience. Also, view your site on Macintosh and Windows computers and in different browsers.

Further reading

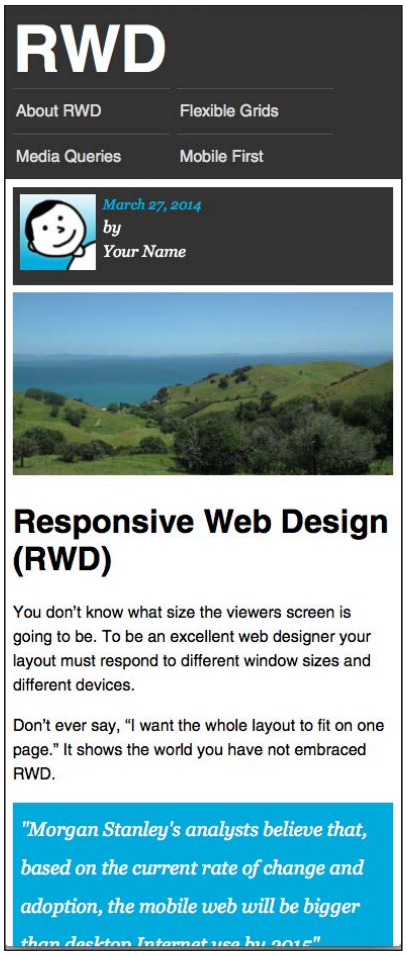
Another good article—a little more technical: http://www.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/

Examples of beautiful RWD: http://mediaqueri.es/

In Class Assignment

1. Get the "Media Queries Start" folder from the Shared Items Server.
2. Create the demo responsive layout shown on this and the following page.
3. Employ CSS in-line media queries that make the mobile layout adjust for a 42em wide and a 62.5em wide screen. Use the hints below and your design and CSS knowledge to create the layouts.
4. **Portage Trail Project:** Design your project site to be mobile first, beginning in your Photoshop comps. You'll need, at minimum a mobile layout and a wide screen layout. You may have more breakpoints if you want.

Demo Responsive Layout: (continued on next page)



RWD

About RWD | Flexible Grids
Media Queries | Mobile First

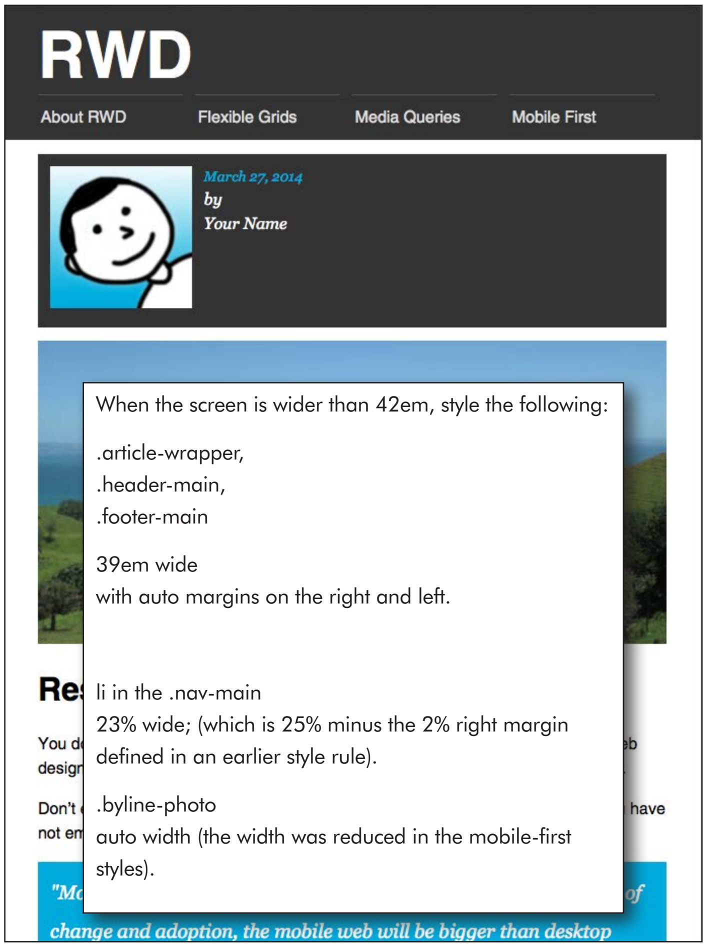
March 27, 2014
by
Your Name

Responsive Web Design (RWD)

You don't know what size the viewers screen is going to be. To be an excellent web designer your layout must respond to different window sizes and different devices.

Don't ever say, "I want the whole layout to fit on one page." It shows the world you have not embraced RWD.

"Morgan Stanley's analysts believe that, based on the current rate of change and adoption, the mobile web will be bigger than desktop Internet use by 2015"



RWD

About RWD | Flexible Grids | Media Queries | Mobile First

March 27, 2014
by
Your Name

change and adoption, the mobile web will be bigger than desktop

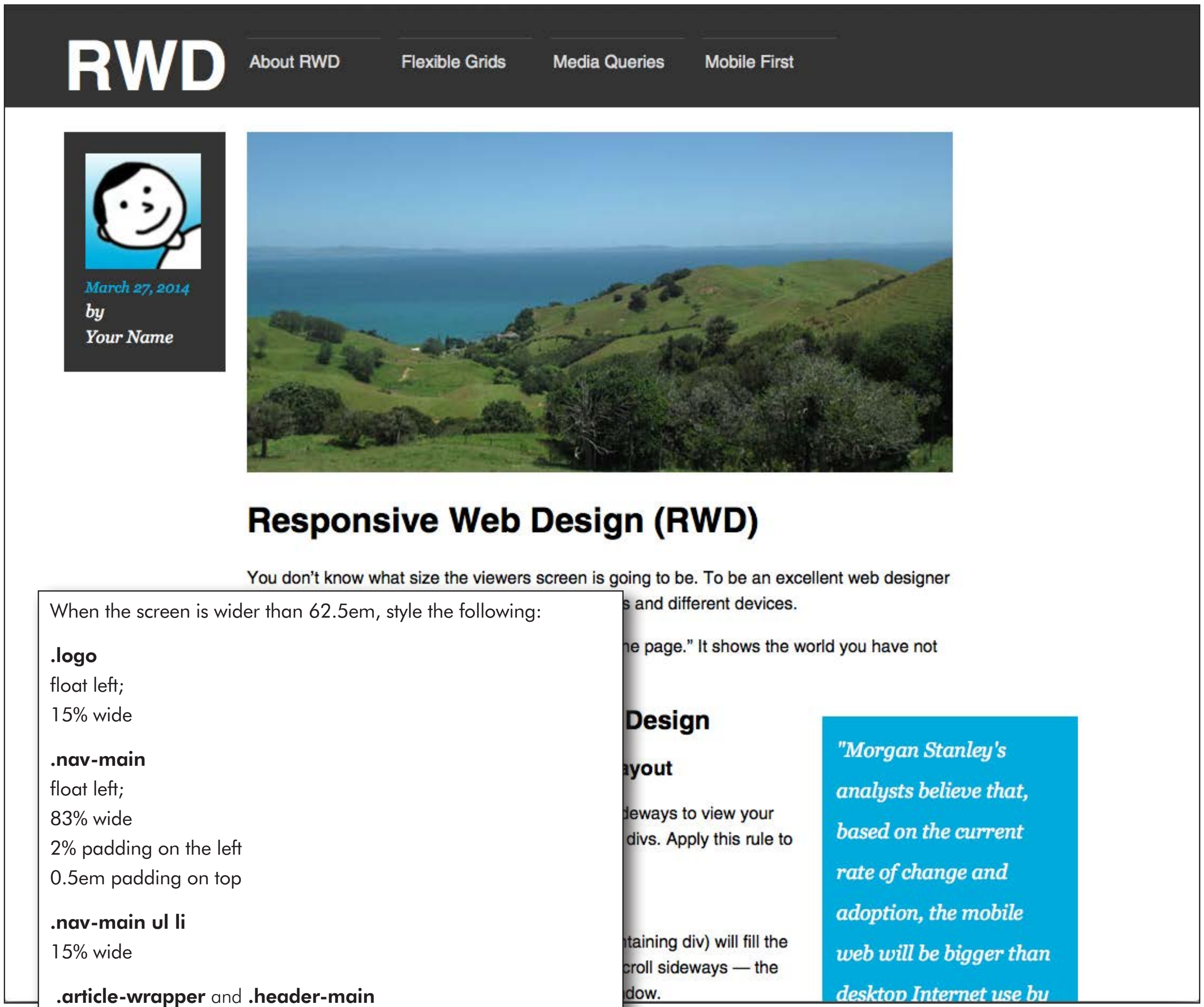
When the screen is wider than 42em, style the following:

.article-wrapper,
.header-main,
.footer-main

39em wide
with auto margins on the right and left.

li in the .nav-main
23% wide; (which is 25% minus the 2% right margin defined in an earlier style rule).

.byline-photo
auto width (the width was reduced in the mobile-first styles).



Responsive Web Design (RWD)

You don't know what size the viewers screen is going to be. To be an excellent web designer
s and different devices.

he page." It shows the world you have not

Design

ayout

deways to view your
divs. Apply this rule to

taining div) will fill the
croll sideways — the
dow.

"Morgan Stanley's analysts believe that, based on the current rate of change and adoption, the mobile web will be bigger than desktop Internet use by

When the screen is wider than 62.5em, style the following:

.logo
float left;
15% wide

.nav-main
float left;
83% wide
2% padding on the left
0.5em padding on top

.nav-main ul li
15% wide

.article-wrapper and **.header-main**
90% wide

.byline-box
float left;
11% wide (15% – 2% padding all around)
2% right margin

.byline-photo
float none (to undo the float left in an earlier style rule)

.content
float left;
66% wide

.pull-quote
float right
32% wide
2% margins on the top, bottom and left.
Negative 17% margin on the right (to move it to the right one column)

Bonus:

When the screen is very wide, keep the type line lengths from getting insanely long by making the default type size bigger

When the screen is wider than 80em,
set the default type size to 125%