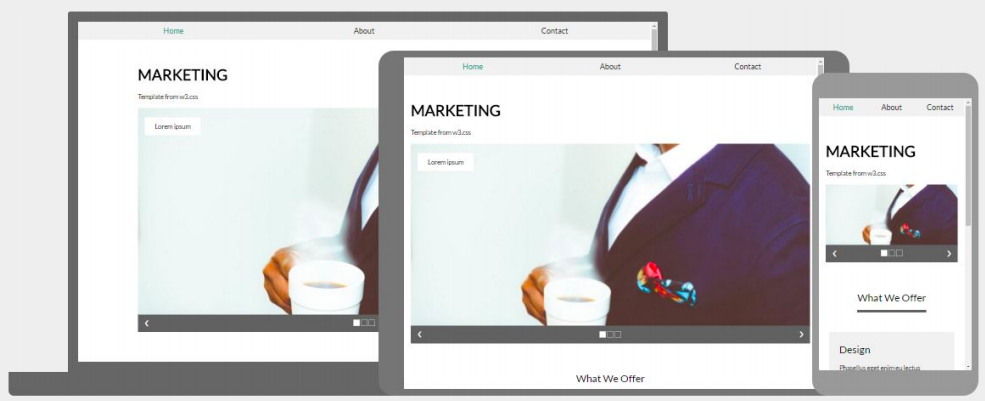
|  |
| --- |
| Tutorial 4: Part 1 – Responsive Web Design (RWD) |

**Part 1:**

Without a responsive layout, text gets too small to read on small devices. If a reader pinches and zooms, increases the page size on their phone, text gets cuts of forcing the reader to scroll back and forth to read. We introduced Responsive Web Design.



## What is a Responsive Web Design?

According to W2Schools, “Responsive web design is about creating web pages that look good on all devices! A responsive web design will automatically adjust for different screen sizes and viewports.”

### Why use RWD?

* The site adjusts its size to its platform desktop, laptop, or smart phone.
* Can reach many more customers.
* Designing for each viewport size makes your site design stronger.
* Improves your SEO (Search Engine Optimization): recommended by Google.
* Other reasons: <https://www.huffpost.com/entry/business-mobile-responsive-design_b_5267077>
* Also: <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>

### How to create a responsive website

* Hire a Web Designer

- Expensive

* Do it yourself from scratch

- Very daunting if not a trained Web Developer

* Content management systems like WordPress
* Still need some technical knowledge
* Number of choices for plug-ins can be overwhelming.
* E-commerce builder like Wix.com or Weebly
* More expensive than hosting your own HTML or WordPress site
* Limited options
* Use a template.

## Why use responsive web design templates?

* Templates are a great way to tap into the power of responsive design without having to become a web developer. Here are some other reasons:
* More design flexibility
* Reduced costs
  + Credit card processing can be done using relatively inexpensive methods such as: PayPal, Authorize.net
  + Can be combined with inexpensive shopping carts.

## Bootstrap

* One of the most popular templates for Responsive Web Design is Bootstrap.
* Developed by Twitter.
* Open source combines HTML, CSS, and JavaScript
* The best resource <http://getbootstrap.com/>
* Other resources:

<http://www.w3schools.com/bootstrap/>

<http://www.codecademy.com/learn>

## Overview of the Tutorial:

**In this Tutorial, you will build a responsive layout using the current form of your Tutorial 3. Any changes that you wish to move and integrate from Tutorial 3 is also allowed but not compulsory.**

For this Tutorial you will:

1. Use the Tutorial 3 as a starting point.
2. Build a responsive layout (1-, 2-, and 3-columns) using *max-width*, *%*, *multiple divs*, the *float* property, and *media queries*.
3. Re-use properties you used earlier (h1, h2, p, .intro, em)
4. Gain more experience using margin and padding.
5. Add any two kinds of images of your choice: content and decoration related.
6. Use a class to style where possible.

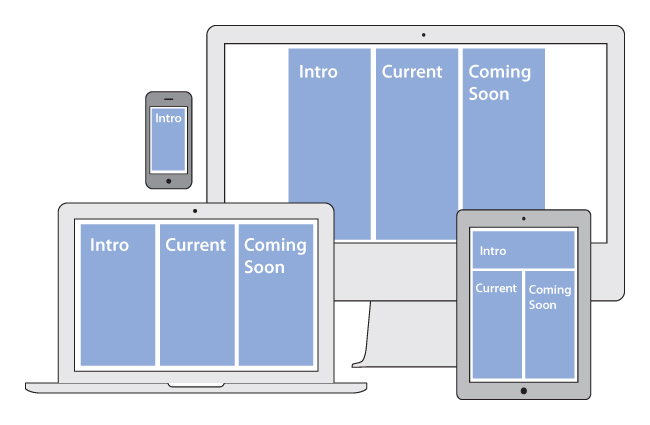
## Outcomes:

* Learn about the importance of Responsive Web Design.
* Review your options when it comes to do-it-yourself responsive e-Commerce sites.
* Learn about Responsive Web Design templates.
* Test your RWD skills on SOLOLEARN: <https://www.sololearn.com/learning/1162>

## RWD Steps:

You have created the initial set up in your previous HTML, CSS and HTML5 tutorials. Create four divs of your choice, **example**: the main container, X-column, Y-column, and a Coming Soon column (The main container + three columns).

The three columns will respond to browser size, moving around as needed, example:



## Style the Divs in your CSS file:

To make your columns respond to the width of your browser, you need to stop setting widths in pixels and start setting them in percentages.

The only div that will use pixels to set a max-width (e.g., the widest an element should ever get) is the main\_container. Set the other divs to 94% width, like so (new example syntax in bold):

Graphical user interface, text, application

Description automatically generated

### Save and View Your Page

You should now see the three divs, in a column, with an equal amount of space on either side.

The left and right margins now look equal. Left margin (3%) + width (94%) = 97%.

That leaves a 3% “extra” space to the right of the divs. Recommendation is to always leave an “extra” space instead of setting both left and right margins. It keeps your layout flexible. Percentages are not the same as pixels and cannot be controlled with the same amount of precision.

### Add Content

You have a basic mobile layout structure, you can now add content (.doc) into the divs. The word document tells you what content to put in each div. **Create your own content (explaining a product or a section of your product).**

A reminder: content goes in the HTML document. And each column is defined by a pair of div tags. Simply replace the words currently being used as placeholders.

### Define the Content

You have the choice to define and add any content and redesign it accordingly to your own preference if it can be responsive in your final design. Possible changes could be: headings and paragraphs etc in the way you want it to look.

## Create Your First Breakpoint

The 94% (plus left margin = 97%) wide columns are fine for a phone but are too wide for a larger screen. When the columns get too wide to comfortably read information, the layout should change. One option is to change the layout to a two-column page. Your layout will have the intro div along the top, with the section A and Section B columns side-by-side below it, example:

A picture containing chart

Description automatically generated

**Note:** It is hard to tell exactly where this layout change should happen. But you can estimate it.

## Determine a Possible second breakpoint

The next larger layout has two columns side-by-side. Estimate the point where you think two columns might work—not so narrow that two-columns get too narrow for the text, but not so wide that the Intro column is too wide.

Add a Media Query in the CSS

You can use a media query to tell the CSS to notice the browser width.

At the bottom of the CSS file, type in the following syntax (In the below example we use 690px, if you have estimated a different width, use your measurement):

@media (min-width: 690px) {

}

**What does this mean?** The syntax says, “pay attention to the media, if the width of the browser is equal to or more than 690px, do this.”

Except it does not actually do anything yet. Because you have not told it what to do.

What do you want the layout to do on browsers equal to or larger than 690px?

You want to:

1. Make the current\_column and coming\_soon\_column turn into a two-column layout.

Thus, the two elements you will be changing in the media query are the current\_column and the coming\_soon\_column. Add those columns *inside* the media query like so (new syntax in bold):

@media (min-width: 690px) {

**#current\_column{**

**}**

**#coming\_soon\_column{**

**}**

}

**Note: In the above example there are two closing curly brackets at the end.** This is correct! One closes the coming\_soon\_column div, the other closes the media query.

## Place the Columns Side-by-Side with the Float Property

Divs are block elements. By default, they show up as a single column of elements. You want two columns to live side-by-side. To fix this, make use of the float property.

It is best to float all the divs (so they recognize each other), and then clear the float when you want to create a new row of elements.

In a 690px media query, start by adding the Intro Column (it is not changing size, but you want to float it). Then add the line float:left; to all three divs, like so (new syntax in bold):

@media (min-width: 690px) {

**#intro\_column{**

**float:left;**

**}**

#current\_column{

width:47%;

**float:left;**

}

#coming\_soon\_column{

width:47%;

**float:left;**

}

}

Next — because this is the div you want to move down to the next row — add a clear:left; to the current\_column. *The clear must be before the float* — after you clear the float:left relationship, you want to start a new one with the div that follows it. New syntax in bold:

@media (min-width: 690px) {

#intro\_column{

float:left;

}

#current\_column{

width:47%;

**clear:left;**

float:left;

}

#coming\_soon\_column{

width:47%;

float:left;

}

}

### Fix any issues with the Margin

The margin between the Intro column and the Current column is too wide. And the space on the right is too narrow.

Keep the 3% on the Intro column and a 2% on the other two columns. This will leave a remaining 3% of space on the right and will balance everything out.

How can you fix this?

Go in to the 1015px Media Query in your CSS, leave the Intro column as is (it already has a 3% left margin), and change the Current column. You can leave the Coming Soon column alone because it already has a 2% left margin. New syntax in bold:

Text

Description automatically generated

Add a Media Query in the CSS

Two images belong in this web page. They are part of the content and will be added in the HTML page. Make sure to save the images in the same folder.

### Add the images in the HTML File.

Depending on your headings style, make sure to use the below syntax:

<h2>Your heading</h2>

**<img src="images/Your Image1.jpg">**

<h3>The exame</h3>

...

<h2>Coming Soon Exame</h2>

**<img src="images/Your image2.png">**

<h3>example</h3>

Make the Images Responsive

For this tutorial, you will use the quickest, easiest method. This is not always the best approach, because it loads large images (and larger amounts of data) then shrinks the images down to size as needed. This can cause web pages to load more slowly on phones.

In the CSS file, add the following syntax so it works at all sizes. To make it work at all sizes, put the syntax before the first media query. All wider browsers will continue to use the styling — unless you tell them otherwise.

A picture containing graphical user interface

Description automatically generated

### Mobile View

Now there might be a few things you will need to fix when considering the requirements of display and design for mobile devices.

Add space between paragraphs

It is recommended that you add a margin-bottom on the **p** in the mobile view (e.g., before the first media query) section of your CSS.

Increasing the line-height adds equal amount of space above and below **an** element. Its recommended using a margin-top and margin-bottom instead of adding line-height. Again, make the changes to your headings in the mobile view section of your CSS.

Add Space Above the Section Headings

Use margin-top and margin-bottom. And make the changes in the mobile view section of your CSS.

### Adjust Columns to Fix Ragged Edge (or Not)

You might encounter some very ragged edges in the text. This happens when a column is too narrow for the text inside it.

*One solution would be to make the column a little bigger.* But you might not have a lot of wiggle room here. The divs already take up 94% of the width of the screen.

*Another option would be to make the text 1px smaller.* But you do not want to compromise legibility.

*For now, I recommend leaving the ragged edges.* You have not seen the page on an actual phone yet. You are probably on a desktop or laptop. Since this is a *one-page exercise*, finish the page, then test it on a couple of phones and tablets to see if you need to make any final tweaks to column widths or text size.

***Note: if you face a choice between readability and a ragged edge, always choose readability****.*

### Use the Break (br) Tag

When two or more lines of text belong together (e.g., lines of a poem, lines of an address) but need to break in a specific place, you can use the break tag.

Break tags should be used sparingly, only when the break is meaningful, and never to fix a ragged edge of text.

When using the break tag, the two lines need to be in the same paragraph, like this:

<p>Through April 16, 2021**<br>**

Tutorial 4. Almost there</p>

### Fix or add any new spaces between columns

Fixing the space between the two columns takes a little *math*.

* The total of all margins and columns in a row must = 100%
* If you want the two columns equal width, their sum must be an even number (e.g., 90%)
* For the total width of the two columns to be even, the sum of the margins in a row must be even.

The image below shows an example on how to make margins and widths changes. New measurements are in red.

Diagram

Description automatically generated with low confidence

Notice that if you add the measurements across the top row (the Intro column): 5+90+5 (leftover) = 100%. If you add the measurements across the bottom row: 5+43+4+43+5 (leftover) = 100%. These measurements will (theoretically) work. **This is an example and might not apply to your current design if you make your page responsive for the web.**

## Fix any Image(s) that might cause an issue with your design

There are different ways to make sure your images have the same width:height ratio. This is you can do it:

1. Make a copy of an image you will be modifying, and work on the original version.
2. Open the image(s)
3. Test various crop-tool ratio settings until the setting matches. Use any crop tool installed on your machine or see: <https://www.adobe.com/africa/photoshop/online/crop-image.html>
4. Assuming since you did not change the name of the image, the newly cropped image should automatically load when you refresh the page.

### Tell Devices to Look at their Viewport Size

Some mobile devices will show a web page at its largest size regardless of media queries.

Why? Since many sites are not yet optimized for mobile, some modern mobile browsers are purposely set to a larger viewport — meaning they act as if their browser is larger than it is. This way, readers see the entire page (just small) and can pinch-to-zoom to see the content they want to read. Theoretically, this is better than only seeing a portion of a web page when it loads.

Since your responsive layout is optimized for mobile browsers, you can use the viewport meta tag to tell browsers to use their real device width for the viewport.

Put the viewport meta tag into the **head of your HTML document**. If you are using Google fonts, make sure your ***Google font link remains first*** in the head element. To keep things clean, put the meta tags near each other. Syntax example:

<head>

<meta charset="utf-8">

**<meta name="viewport" content="width=device-width">**

<title>My Petstore in Responsive X</title>

<link href="inf3014f\_x.css" rel="stylesheet" type="text/css">

</head>

**PLEASE NOTE THAT YOU ARE REQUIRED TO USE YOUR OWN DESIGN HENCE THERE ARE NO DESIGN TEMPLATES. MAKE SURE TO BE CREATIVE.**

* Make sure all HTML files, CSS files, and images should be within one main folder.
* Save your files (i.e., images etc.) with the correct file extensions, and make sure you zip your folder before submission!
* Zip all files in one folder and submit on Vula.

Submit this Tutorial under *Assignments* 4 on Vula.

|  |  |
| --- | --- |
| **Tutorial 4:**  **Part 1 Marksheet** | |
| Style Divs in CSS stylesheet | 8 |
| Added additional content in Divs | 8 |
| Created first breakpoint | 5 |
| Added a Media Query in the CSS | 5 |
| Used the Float property | 5 |
| CSS works fully (this is subjective, 4 of the 8 marks are for creative use of CSS, or doing more than the standard tags [<p>, <body>, <table>,  <h1>, <h2>, etc.]) | 8 |
| Created second breakpoint | 5 |
| Added another Media Query in the CSS |  |
| Added newly cropped images (used crop feature) | 5 |
| Adjusted columns to fix Ragged edges | 5 |
| Added any other adjustments | 6 |
| **Total marks:** | **60 marks** |

# Finished!