Exercise 12 – Media Queries

Objective

In this exercise we will take our mobile first page and build three breakpoints for larger displays. In the second part of the exercise you will investigate print media queries to optimise the UI for other display types.

Overview

You are going to use media queries in the first part of the exercise to create a tablet desktop and maximum viewport size for our website. Repositioning and resizing where appropriate for different device screen size and capabilities.

In the second part of the exercise we will add and modify a print based stylesheet to optimise the website for that media type creating a print friendly version of our website. This exercise will take around ***45 minutes***.

Exercise Instructions

Part 1 – Creating a responsive page using media queries

Breakpoint 1 – Tablet

1. Open **mediaQuery.html** and **mobile.css** and check the page in Chrome. Use the mobile viewer and check the page for an iPad and check how it works with different orientations. It is OK at the moment but the block sized navigation is overwhelming on the screen. We’re going to rearrange how the navigation works and set the first breakpoint.
2. Locate the comment **Media Queries** this is where we will start the work. First of all, we need to create the media query block:

@media only screen and (min-width: 480px) {

}

1. The remainder of the code we will write in this part of the exercise will be placed within this block. The conditional statement means this will not apply to the document until the device or browser settings meets this minimum requirement.
2. The following CSS will reposition the <a> within the <nav> element but still maintain minimum button size:

nav a {

float: left;

width: 27%;

margin: 0 1.7%;

padding: 25px 2%;

margin-bottom: 0;

}

nav li:first-child a {

margin-left: 0;

}

nav li:last-child a {

margin-right: 0;

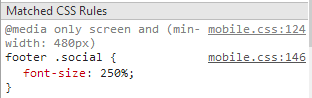
}

1. Note that we are using a lot of percentage based measurement units. This breakpoint will deal with landscape phones and most tablets, the resolutions will be varied so we need to cater for them. Percentages allow the width to adjust within the device perspective.
2. Save all files and check the page in a browser, note the difference in the browser behaviour now we have big buttons but not buttons that stretch all the way across the screen. Check the page in any phone device view and then rotate the device, you should see the difference.
3. Next we will turn our attention to the social icons at the bottom of the page. We are also going to use percentages to scale out our social icons

footer .social {

font-size: 250%;

}

1. Save and test the page. If you bring up the developer console in Chrome you should note the media query is applied. This will scale the text to 250% of its normal size.
2. The final part we will add to this breakpoint is a class that will set the text to two columns. The entire class will be defined within the media query this means it will only apply when the media query is met.

.two-col {

-moz-column-count: 2;

-moz-column-gap: 20px;

-webkit-column-count: 2;

-webkit-column-gap: 20px;

-moz-column-rule-color: #ccc;

-moz-column-rule-style: solid;

-moz-column-rule-width: 1px;

-webkit-column-rule-color: #ccc;

-webkit-column-rule-style: solid;

-webkit-column-rule-width: 1px;

}

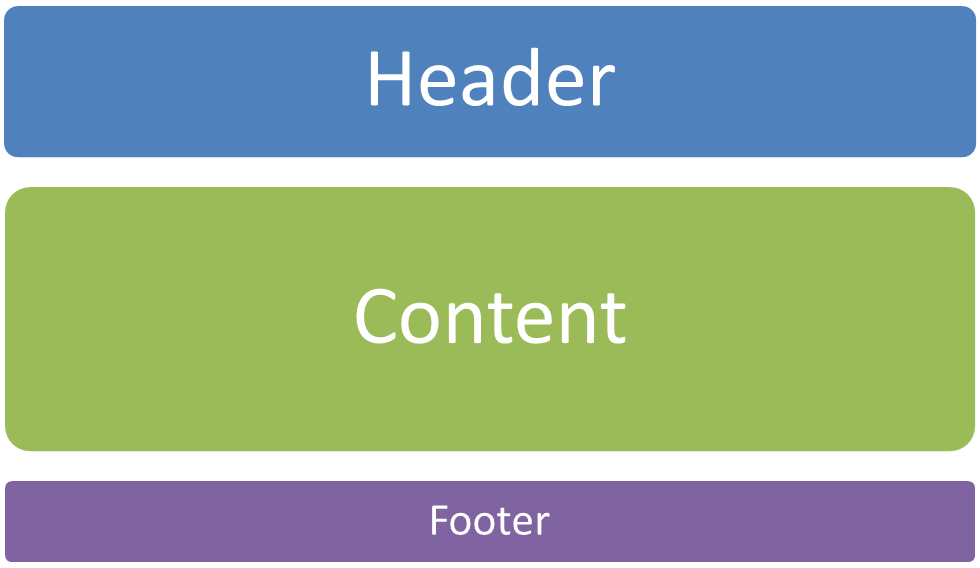
1. Save all page and check the text. Now when you switch into and out of the media query the navigation and columns of the text will switch.

Breakpoint 2 – Desktop

Moving to a ‘desktop’ sized device is a much more significant redesign than our first breakpoint. With the extra real estate the large screen provides we should rethink where our main sections go:

* header-container (containing the navigation)
* main-container (containing an aside)
* footer-container (containing social icons)

As our significant areas of content we now have a decision on where our content sits



Navigation

Aside

Content

This will make maximum use of the screen space we have moving the aside to the right of the screen within the main-container stack

1. Beneath the previous media query add the following media block:

@media only screen and (min-width: 768px) {

}

1. The remainder of the code for this section of the exercise will sit inside this media block. Note the min-width parameter.
2. We’re going to reposition the navigation first by moving the navigation to the right of its containing tile.

/\* ============

WIDE: Menu

============ \*/

.title {

float: left;

}

nav {

float: right;

width: 38%;

}

1. For simplicities sake we have added the classes you’re using into the markup, conceptually the CSS is familiar in terms of work we have already investigated, it’s the Responsive and Mobile First techniques we’re aiming to master in this exercise.
2. Save all files and check the page in the browser, the navigation now appears to the left of the header and the title to the right.
3. The next step is to reposition the content in the **main-container** area for the available space

.leftCol {

width: 57%;

float: left;

}

.main article, .main figure {

float: left;

}

.main aside {

float: right;

width: 28%;

min-height: 425px;

}

1. The key to understanding the function of the code here is to consider what is happening with the containing blocks (we told you the box model was the most important part of CSS!). The container main-container is our ‘stack’ within it we can move content. In this scenario we have positioned items using floats and widths. If you save the page and check the difference, we are close to the planned outline.
2. Finally for this section of the exercise we are going to tidy up the appearance of the wide CSS view:

.three-col {

-moz-column-count: 3;

-moz-column-gap: 20px;

-webkit-column-count: 3;

-webkit-column-gap: 20px;

-moz-column-rule-color: #ccc;

-moz-column-rule-style: solid;

-moz-column-rule-width: 1px;

-webkit-column-rule-color: #ccc;

-webkit-column-rule-style: solid;

-webkit-column-rule-width: 1px;

}

footer .social {

font-size: 450%;

}

.header-container,

.main aside, .main figure {

-webkit-box-shadow: 0 5px 10px #aaa;

-moz-box-shadow: 0 5px 10px #aaa;

box-shadow: 0 5px 10px #aaa;

}

1. Save and check out your page and we’re about done.

Breakpoint 3 – Maximal Width

If you consider what we did earlier in the course, we constrained the width of the page. Currently our page will stretch right the way across the visible page. We may well get away with it on our current monitors but imagine this on a 40” TV! Let’s constrain it:

1. Create a media query with a minimum with of 1140px
2. Create a selector for the class **wrapper**
3. Set the width to 1026px and centre the wrapper using margins.

With this completed bring up the mobile display tool and check your three breakpoints.

Part 2 – Other Media Types

Media queries are not just for responsive design; they are also very effective for dealing with other media types including print. We’re going to add a print media sheet and utilise it to strip unnecessary parts of the page ready for print.

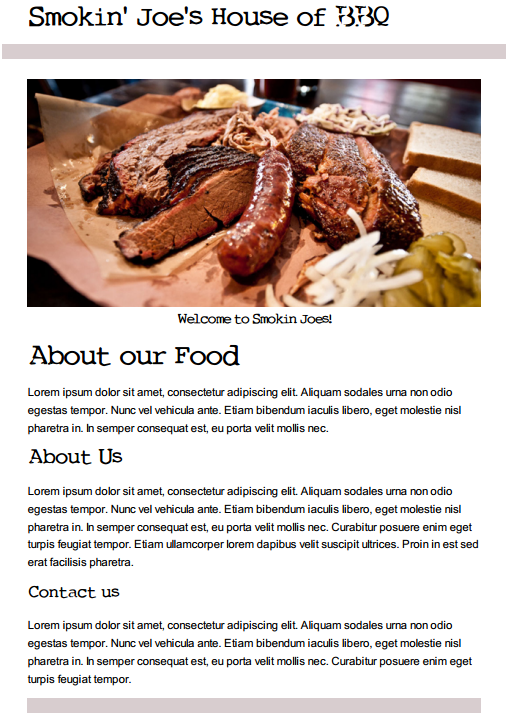
1. Open the page in Chrome and hit CTRL+P to print the page. The print preview automatically strips the background image away – we don’t want to print that anyway! The colours and the opacity are a mess; though the nice greyscale is good for screen design we aren’t going to need them. The print preview also ignores the media queries we’ve added as we specified them to be screen. This is good for us; the mobile first paradigm uses stacks of information in vertical order which is also brilliant for print!
2. Within the CSS folder you will find a file called **print.css** add a <link> as per usual but also add a **media="print"** attribute. Once you have done this open the **print.css** file and examine the existing content. Quite a few of the CSS styles already present include the **!important** selector which will override all other styles when we choose to print.
3. Hit CTRL+P once again and we should already see an improvement: the last line of the About Us section is no longer orphaned. We’re going to improve the print view a little more in the next few steps.
4. Locate the **\*{ }** selector; within this selector we are going to override the font colour for the document and take away the drop shadows we are using. They aren’t needed for print.

background: transparent !important;

color: #000 !important;

box-shadow: none !important;

text-shadow: none !important;

1. Save and check the print preview again; the text will now be black and the box shadows we’ve used are now removed.
2. The navigation buttons are not of use to a printed representation of the page. Locate the **header nav** selector and ensure the nav element does not display in the print style sheet.
3. Save the page and test for a final time you should find the page looks something like this: