Responsive Pages

Setup

For this exercise, you can use the completed *grouping* HTML and CSS files from the earlier exercise. You can either work on them directly, or make copies of them, the choice is yours.

Open the HTML page in your browser and activate the responsive design /device preview tool

- Chrome: Open developer tools, then toggle the device toolbar (Ctrl/Cmd+Shift+M)
- Firefox: Ctrl/Cmd+Shift+M

Make sure you view the page at different widths, down to the smallest possible.

The Issues

- As the width of the screen decreases:
- The nav bar content starts to overflow its container
- The 2 articles become too narrow to be read comfortably

At the smallest screen sizes, it becomes apparent that we are wasting space on the left and right sides of the page (the orange areas)

Fixing the nav bar

The nav bar *breaks* when the screen is approximately 460px wide. So we will create a media query that targets devices 480px or wider (the extra 20px is wiggle-room). When we convert 480px to *ems*, we get **30em** which is what we should use in our expression.

Follow the steps outlined in the presentation (pages 29 - 32) to move the display: flex declaration to a media query.

When done, test your page in the browser again. Does the nav bar change its layout at the correct size?

Fixing the articles

The 2 articles start to become hard to read at between 500px and 600px. If there was more content in them, the issue may present itself at larger screen sizes too. We will use 600px as our break point, which equates to **37.5em** .

In *grouping.css* , locate the rule targeting main. The two articles are displayed side-by-side due to the display:grid declaration in this block. There are also other declarations in the block which only apply to grid layout.

Repeat the process you followed for the nav bar, but this time:

- Place your media query immediately after the main declaration block
- Change the **30em** in the media query's expression to **37.5em**
- Move the display, grid-template-columns and column-gap declarations into the media query

When done, test your page in the browser again. Does the layout of the articles change at the correct size?

Article polish

You may notice that there is no space between the articles when they are stacked one above the other. We can fix this with bottom margin on the HTML article, but should be careful when doing so.

- We want to apply the margin, only when screen width is less than 37.5em

  (600px)
- We do not want bottom margin on larger screens (it might disturb our layout)

We can approach this in several ways:

- Outside the media query, set margin on the HTML article, inside the media query, reset the margin to 0
- Create a media query with *max-width* as opposed to *min-width* in its expression, using the same breakpoint as above.

We will use the second method as it is the simpler approach (i.e. less code for us to write and maintain!).

Locate the rule targeting .html-feature in your CSS file. Immediately after this block, add a media query like the previous ones, changing the expression to **max-width: 37.5em** .

Inside the media query, add a CSS rule targeting .html-feature. Inside this rule, set the bottom margin to 1em;

Test your page in the browser again. Is there a space when the articles are stacked?

Better use of space

The final problem to solve is the wasted space on the left and right sides of the page (i.e. the orange region), and arguably at the top of the page too. When we are displaying pages on small screens, we don't have much space in which to display our content and shouldn't waste space unnecessarily.

If you recall, the orange region surrounding the page was created when we added rules to the .page-container selector in our CSS. Specifically:

- Setting its *width* to 90%
- Setting its left and right *margin* to *auto*

So, as before, we simply need to add a media query that determines when these styles should apply.

Unlike some of the other issues we have solved, this one does not cause the layout to break, so the *break point* must be determined by us, subjectively. i.e. at which size do we feel the space is appropriate.

We will use the same break point we used for the articles (600px aka 37.5em) so that the 2 layout changes occur at the same time.

1. Locate the .page-container selector in your CSS file. Immediately after its declaration block, add a media query targeting **screen** devices with a **min-width** of **37.5em** .
2. Add a new declaration block inside the media query, with the .page-container selector
3. Move the **width** and **margin** declarations from the original declaration block to the one you just added to the media query.

Save your files and preview the page in your browser. You should notice that, on small screens, the amount of space has been reduced, but not eliminated entirely. Can you think what might be causing this?

We are not setting any margins on page-container for small screens, so it can't be them. There is padding that is applied on small screens, but that is responsible for the spacing *inside* the page-container box. The orange space is *outside* the page-container box.

Debugging issues like this is not trivial.

- The spacing *could* be caused by descendants of page-container, such as the h1(e.g. if it had a large top margin).
- Or it *could* be caused by ancestors of page-container, which in this case is the <body>and <html>elements (e.g. if <body>had padding or margin applied to it).

When we examine our CSS, we can see that the <h1> has a top margin value, but it is set to 0, so it is probably not guilty.

We can also see that <body>has no padding or margin rules at all, so it is probably not guilty too… or is it?

Remember, some elements have *default styles* which are applied by the browser internally! Could <body>be such an element?

Let's debug the code!

In your CSS file, locate the body selector and add a new declaration to it: padding: 0;. Preview the page in your browser. There will be no change, which tells us that padding probably isn't the issue, so we can remove that declaration from our CSS.

Next, we will see if *margin* is the cause. In the body rule in your CSS, add a new declaration: margin: 0;. Save the files and preview the page in your browser.

You should now see that, on small screens, there is no longer any orange areas on the top, left and right of the screen. This tells us that the phantom space is indeed caused by default margin which the browser is applying to body.

However, you will also notice that, on larger screens, we have lost the orange space at the top of the page, which looks odd.

Therefore, we need to do use a media query to apply the styles conditionally.

The end result we want is:

- On small screens, body should have no margin
- On larger screens, body should have a small margin at the top
  - More specifically, this should apply when the width of page-container is set to 90%

So…

1. In the body rule, we set margin to 0 on all sides (which we have just done)
    - In other words, we are overriding the browser's default style for this element, and the style will apply to all screen sizes
2. Locate the media query that sets the *width* of page-container to 90%. Inside the media query, add a new declaration block with body as the selector. Add a declaration to the rule, setting the *margin-top* to 1em.
3. In other words, when the page-container switches into "reduced width" mode, we are re-enabling the margin on the top of body

Save your files and preview the page in your browser. There should be no orange space visible on the top, left and right when viewed on small screens. As the screen gets larger, the orange space should reappear on the top, left and right sides of the page.

You *may* see orange space at the bottom of the page on small screens, but this is entirely different to the margin issue we just resolved. See if you can resolve this issue yourself (hint: it is all about the background colour applied to the body element)

Summing Up

At this point, you should have a responsive web page which adapts its layout and appearance, depending on the environment it is being viewed in. And you have achieved this without altering the HTML in any way. This is an example of why *separation of concerns* is such an important part of software development.

When adding the media queries, we have taken the approach of adding the query immediately after the rule we are modifying, resulting in media queries scattered throughout our stylesheet, some of which have the exact same *expression*. Be aware that there are other ways that we could organise the code, each with their own pros and cons. Which approach you take is largely a personal preference.

Further exercises

1. Make a printer-friendly version of your page. i.e. use the techniques we saw in the presentation to apply different CSS when the user attempts to print the page. You may use a separate stylesheet, or an *at-rule*, whichever you prefer. The things you want to change in the Print stylesheet are:

- Remove all background colours (or set them to white)
- Make sure all text, except links, is black
- Hide the navigation bar and footer elements (use display: none; for this)
- Make the HTML and CSS articles sit one on top of the other, not side-by-side (i.e. the same thing we have just done for small screens)

2. Revisit your previous projects (Balloons, Families in Dublin, Etc) and see if you can make them responsive too.