ARCH A4988

Coding for Spatial Practices

Responsive Web

Learning Objectives

Learning Objectives

1. Use layout techniques with width, min-width, max-width, vh and vw.
2. Understand and apply Image Optimization techniques such as min-width, max-width, retina display, etc.
3. Identify breakpoints and apply media queries to adjust layout.
4. Effectively use media queries to adjust and adapt your page layouts.
5. Understand how media queries work in the cascade.
6. Understand that Responsive Design is dictated by visual break points and not device sizes.

Agenda

1. Hello, World!
2. Responsive Web – The Basics
3. Exercise

The Basics

Responsive Web

Background

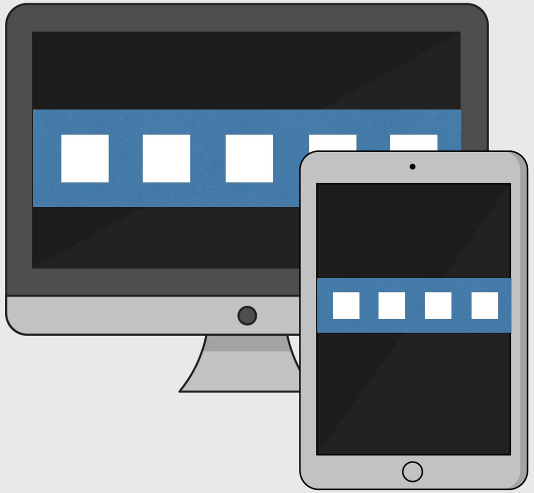
- Tablet/mobile phones have vertical The iPhone, unveiled in 2007, changed how people used the internet. It took until ~2009 to become popular.
- This was first addressed by creating mobile-specific sites (started with m.site.com), which meant you had to make multiple sites.
- As browsers evolved, it became possible (by 2010) to build one site where you could adjust styles based on screen widths, creating one site for all devices.
- Ethan Marcotte pioneered this technique with an article on the subject: http://alistapart.com/article/responsive-web-design

Responsive Web

Background

Laptops and desktops typically have horizontal orientations.

Designers can place several elements next to each other with plenty of room.

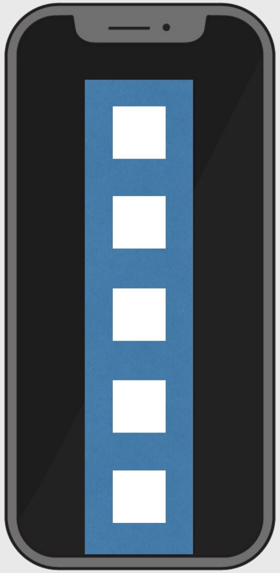


Responsive Web

Background

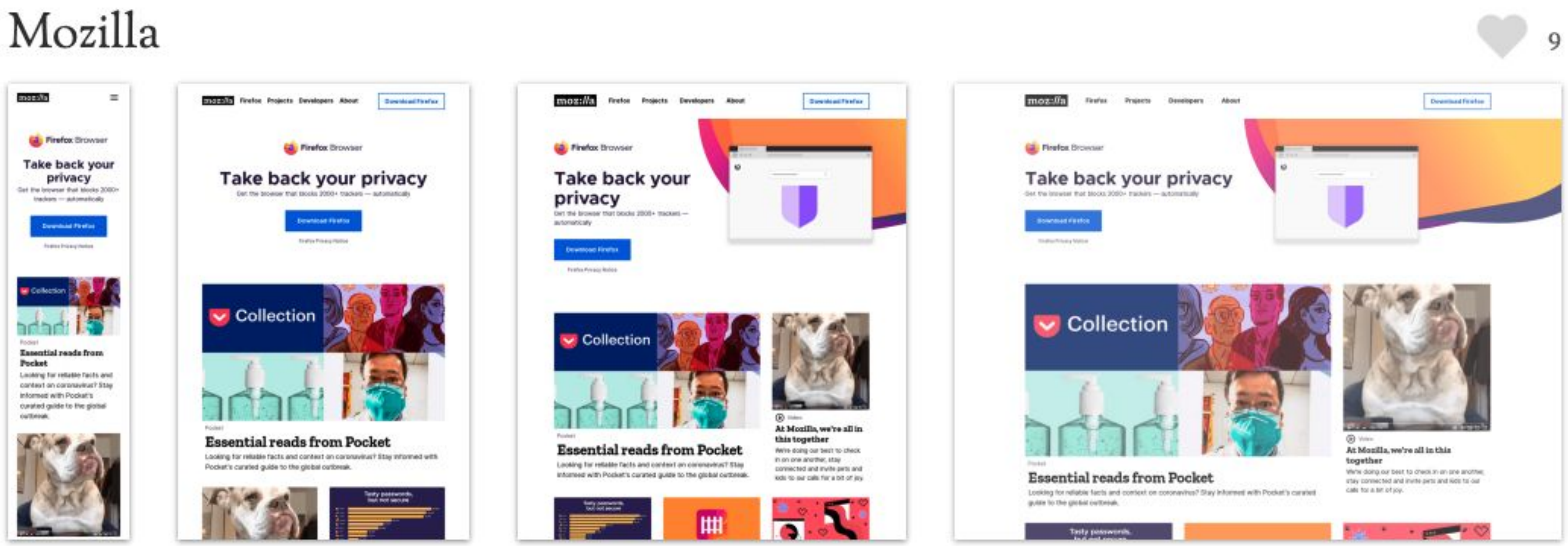
Tablet/mobile phones have vertical orientations.

Content should rely on having many rows of one item each. There’s not enough room for more!



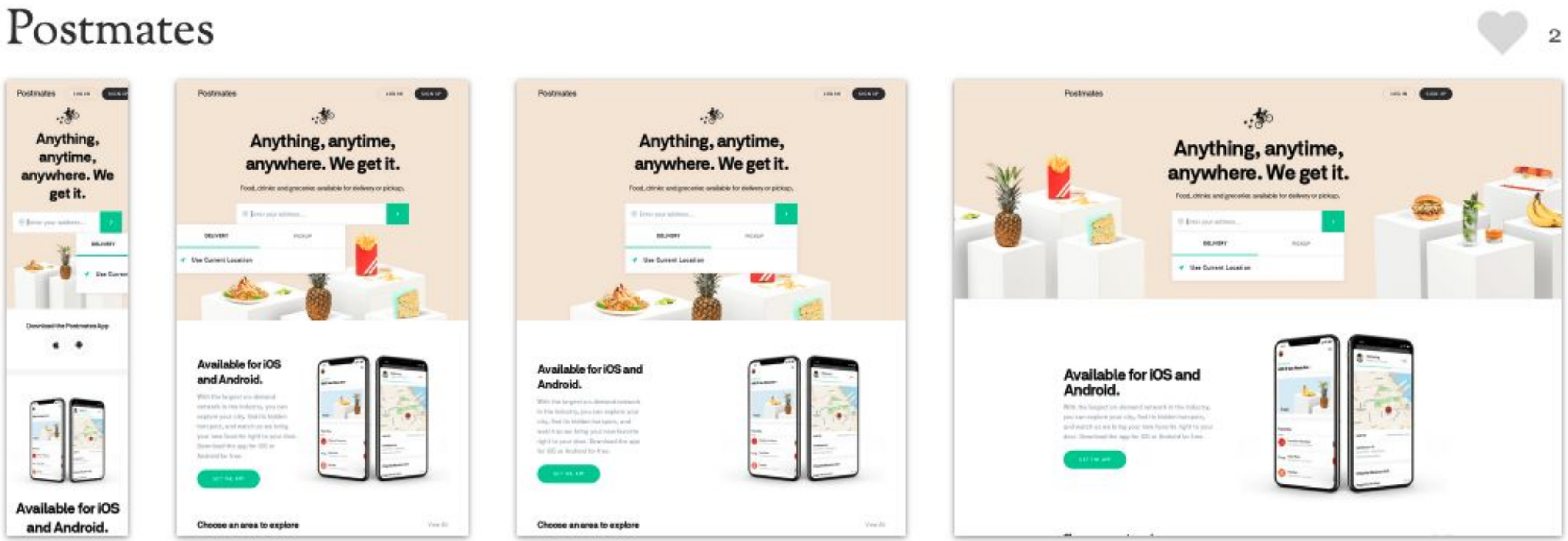
Real Cases:

Good Responsive Design: Mozilla


Mozilla
9
Firefox Browser
Take back your privacy
Get the browser that blocks 2000+ trackers — automatically
Download Firefox
Firefox Projects Developers About
Collection
Essential reads from Pocket
Looking for reliable facts and context on coronavirus? Stay informed with Pocket's curated guide to the global outbreak.
At Mozilla, we're all in this together
Tasty passwords, but not secure

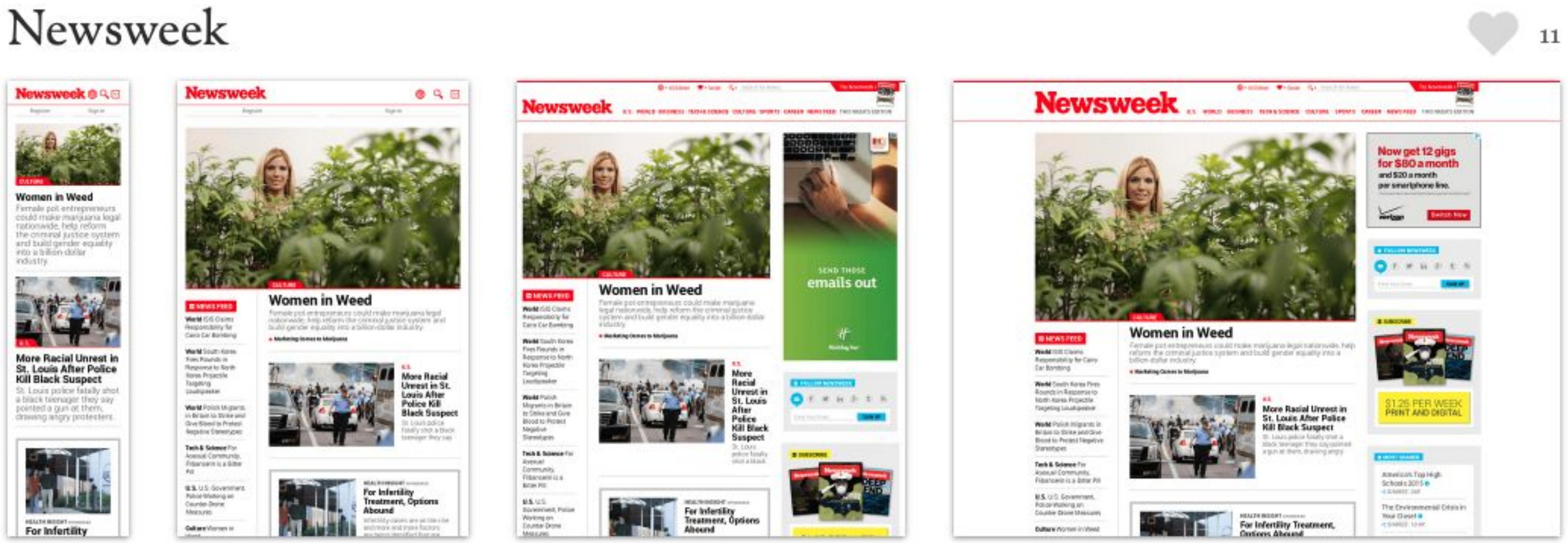
Real Cases:

Good Responsive Design: Postmates


Postmates
2
Anything, anytime, anywhere. We get it.
Food, drinks and groceries available for delivery or pickup.
Delivery
Pickup
Use Current Location
Download the Postmates App
Available for iOS and Android.
Get the app
Choose an area to explore
View All

Real Cases:

Good Responsive Design: Newsweek


Newsweek
11
Women in Weed
More Racial Unrest in St. Louis After Police Kill Black Suspect
For Infertility Treatment, Options Abound
Now get 12 gigs for $80 a month
$1.25 PER WEEK PRINT AND DIGITAL

Mobile First

Mobile First

Background

The mobile-first approach is exactly as it sounds: designing for the smallest screen and working your way up. The key to a mobile-first approach is a content centered one.

Since mobile has the most limitations (e.g. screen size and bandwidth), designing within these parameters in mind forces you to prioritize content ruthlessly.

Mobile first is additive.

Fluid Grids & Layout

Fluid Layout

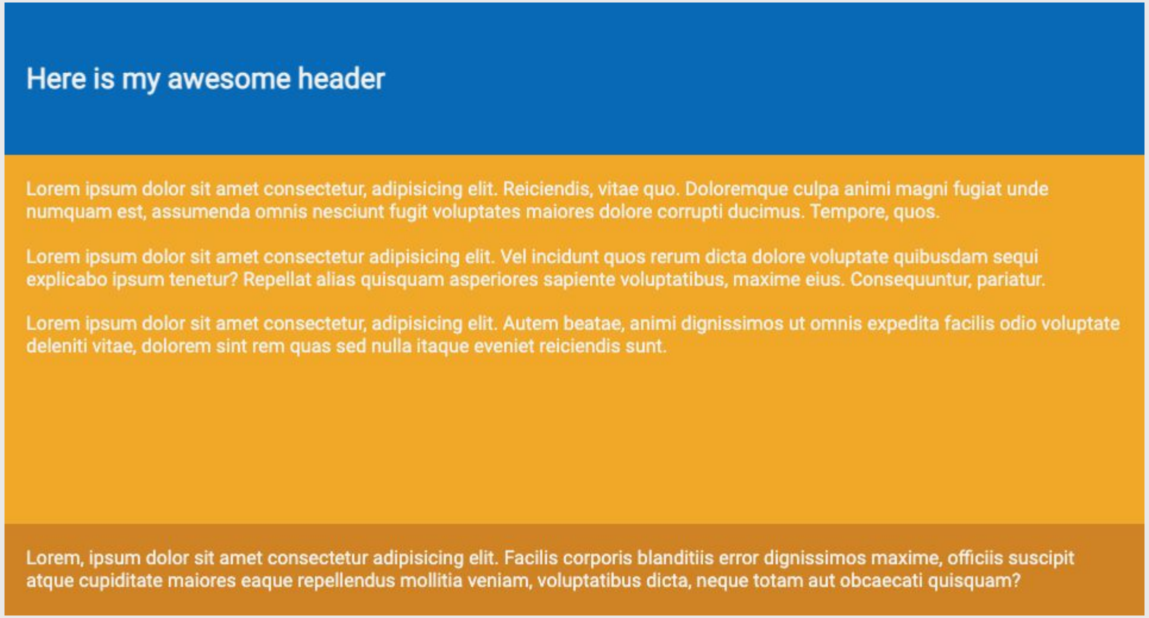
Fixed Width Layout

Relies on a container of a fixed width
(uses static units)

Resizing the browser/viewing it on a
different device won't have an effect on
the page

Fluid Layout

Fixed Width Layout



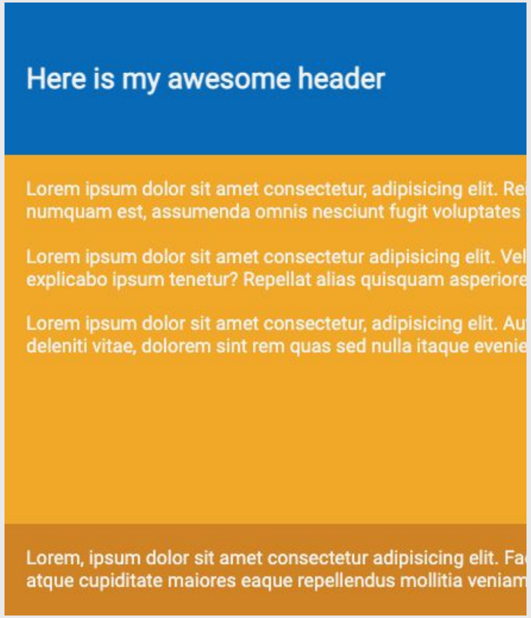
Here is my awesome header

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Reiciendis, vitae quo. Doloremque culpa animi magni fugiat unde numquam est, assumenda omnis nesciunt fugit voluptates maiores dolore corrupti ducimus. Tempore, quos.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel incidunt quos rerum dicta dolore voluptate quibusdam sequi explicabo ipsum tenetur? Repellat alias quisquam asperiores sapiente voluptatibus, maxime eius. Consequuntur, pariatur.

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Autem beatae, animi dignissimos ut omnis expedita facilis odio voluptate deleniti vitae, dolorem sint rem quas sed nulla itaque eveniet reiciendis sunt.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Facilis corporis blanditiis error dignissimos maxime, officiis suscipit atque cupiditate maiores eaque repellendus mollitia veniam, voluptatibus dicta, neque totam aut obcaecati quisquam?



Here is my awesome header

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Re
numquam est, assumenda omnis nesciunt fugit voluptates

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel
explicabo ipsum tenetur? Repellat alias quisquam asperiore

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Au
deleniti vitae, dolorem sint rem quas sed nulla itaque evenie

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Fa
atque cupiditate maiores eaque repellendus mollitia veniam

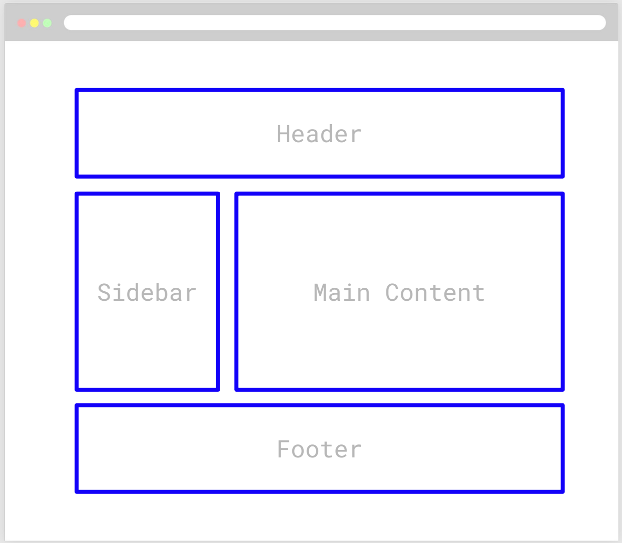
Fluid Layout

Fixed Width Layout

Fixed width layouts do not change size as the user increases/ decreases width of browser window

To create:

- Width of any main boxes is set in pixels
- Layout can be centered by setting the value of the left and right margins to auto



Fluid Layout

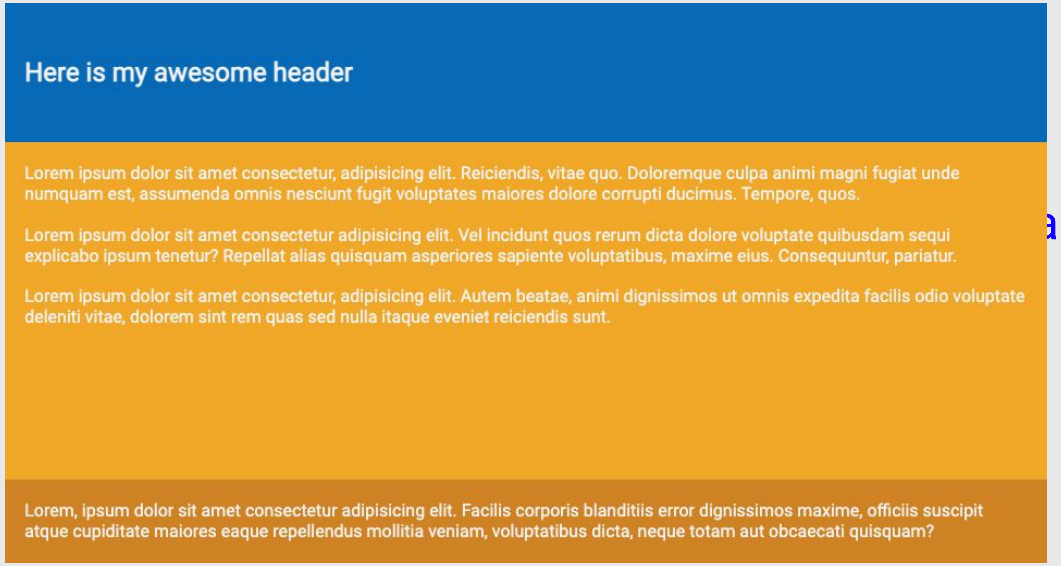
Fluid Layout

A fluid layout often has:

- A container element that fill the entire screen size
- The container element responds to adjustments in browser widths
- It does this by using relative widths (percentages)
- widths of the columns nested within the container adjust accordingly
- It does not use media queries

Fluid Layout

Fluid Layout



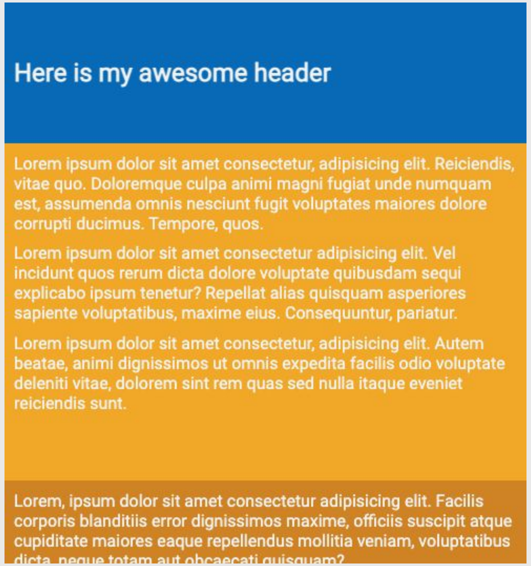
Here is my awesome header

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Reiciendis, vitae quo. Doloremque culpa animi magni fugiat unde numquam est, assumenda omnis nesciunt fugit voluptates maiores dolore corrupti ducimus. Tempore, quos.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel incidunt quos rerum dicta dolore voluptate quibusdam sequi explicabo ipsum tenetur? Repellat alias quisquam asperiores sapiente voluptatibus, maxime eius. Consequuntur, pariatur.

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Autem beatae, animi dignissimos ut omnis expedita facilis odio voluptate deleniti vitae, dolorem sint rem quas sed nulla itaque eveniet reiciendis sunt.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Facilis corporis blanditiis error dignissimos maxime, officiis suscipit atque cupiditate maiores eaque repellendus mollitia veniam, voluptatibus dicta, neque totam aut obcaecati quisquam?



Here is my awesome header

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Reiciendis, vitae quo. Doloremque culpa animi magni fugiat unde numquam est, assumenda omnis nesciunt fugit voluptates maiores dolore corrupti ducimus. Tempore, quos.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Vel incidunt quos rerum dicta dolore voluptate quibusdam sequi explicabo ipsum tenetur? Repellat alias quisquam asperiores sapiente voluptatibus, maxime eius. Consequuntur, pariatur.

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Autem beatae, animi dignissimos ut omnis expedita facilis odio voluptate deleniti vitae, dolorem sint rem quas sed nulla itaque eveniet reiciendis sunt.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Facilis corporis blanditiis error dignissimos maxime, officiis suscipit atque cupiditate maiores eaque repellendus mollitia veniam, voluptatibus dicta, neque totam aut obcaecati quisquam?

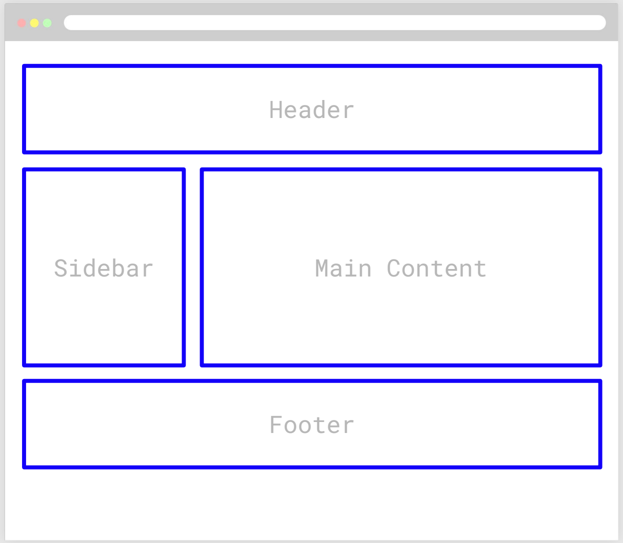
Fluid Layout

Fluid Layout

Fluid layouts stretch and contract as the user increases/ decreases the size of their browser window

To create:

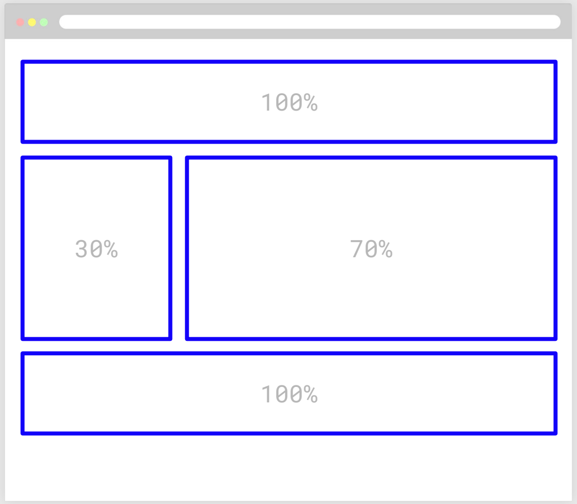
- Uses percentages to set the width of each box so that the design will stretch to fit the size of the screen



Fluid Layout

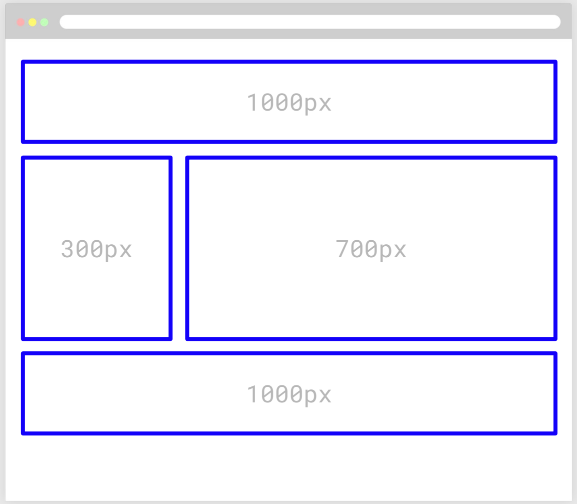
Fluid vs Fixed Layout

Relative Units



Fluid Layout

Static Units



Fixed Layout

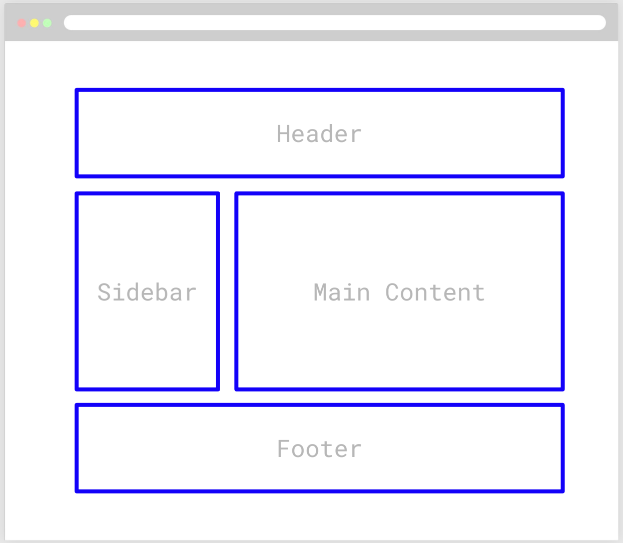
Guided Walk-Through:

Fluid Layout with Flexbox

20 minutes

In this exercise, we'll build a sidebar layout using flex properties. In desktop, the sidebar sits to the left of the main content. However, in mobile the sidebar is sandwiched between the header and the main content with the footer on the bottom of the stack.

Go to the **starter_code_week_08** folder and complete the **fluid_layout_flexbox** exercise.



Flexible Images

Responsive Background Images

background-image

```
.image-style {
  background-image: url(img/image.jpg);
  background-position: center center;
  background-size: cover;
}
```

Background images are somewhat natively responsive; they won't spill out of their containers. But containing elements will crop the image.

To adjust which part of the image you see, use **background-position** and **background-size**. In particular, **background-size: cover;** is pretty close to magic: It uses an algorithm to make the image match its container.

Images Sized to their Containers

```
.image-style {
  display: block;
  max-width: 100%;
  height: auto;
}
```

Images want to display at their dimensions (say, 800 x 600px), but they’ll often be in containers and will spill over if they’re bigger. This code forces the image to be a block element whose width matches its container.

Images Sized to their Containers

object-fit

```css
.image-style {
  object-fit: cover;
  width: 100%;
  height: 180px;
}
```

The **object-fit** property does for images what background-size does for background images: it provides options for how an image is displayed within a designated area, hiding some of it if necessary.

HTML srcset + sizes Attributes

Figuring out which image should go to a particular device is cumbersome and nearly impossible. Now, you can tell the browser how much of the screen an image takes up (with the sizes attribute) and give it the image at different sizes (srcset). The browser does the rest.

Both sizes and srcset are attributes you can optionally add to any <img> tag you want to enable.

Code Sample: HTML srcset + sizes

```
<div class=”image-container”>
  <img src=”img/image-large.jpg” alt=”Forest Lands”
       srcset=”img/image-large.jpg 1200px,
               img/image-medium.jpg 800px,
               img/image-small.jpg 400px”
       sizes=”(min-width: 1200px) 33vw,
              (max-width: 1199px) and (min-width: 768px) 50vw,
              100vw”>
</div>
```

The code is a doozy! We’re telling the browser we have three of the same image with various pixel widths (1200px, 800px, 400px), and then what size the image displays at a given screen width (33vw, 50vw, 100vw).

Guided Walk-Through:

Flexible Image Investigation

5 minutes

Let’s see how a browser handles responsive images.

Sir Alfred Joseph Hitchcock KBE
Director, Screenwriter, Producer and Editor

Breakpoints

Breakpoints

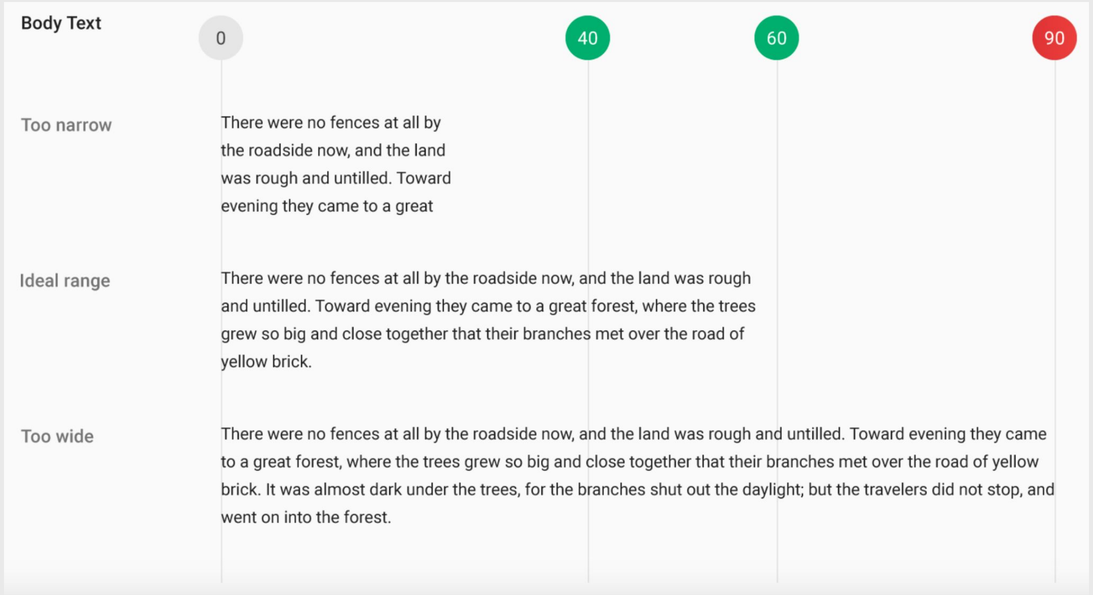
Add breakpoints where your design breaks. If you start with your design for the smallest device (think, mobile-first) and start to drag your browser window wider, when do the lines become too long to read comfortably? When could you make better use of the screen? **That is the point at which to add a media query** and write some additional CSS.

Breakpoints

Working in this way means that any device coming in under that breakpoint gets the narrow layout, and anything coming in over that breakpoint gets the layout making more use of horizontal screen space. It doesn’t matter if the device is an iPhone, Samsung or whatever.

Breakpoints

Line lengths becoming too long is a good indicator of the design “breaking” and requiring a breakpoint. The ideal way to indicate that is to use the **em** unit, as that will give a consistent result if the user has larger text that you expected.



There were no fences at all by the roadside now, and the land was rough and untilled. Toward evening they came to a great

There were no fences at all by the roadside now, and the land was rough and untilled. Toward evening they came to a great forest, where the trees grew so big and close together that their branches met over the road of yellow brick.

There were no fences at all by the roadside now, and the land was rough and untilled. Toward evening they came to a great forest, where the trees grew so big and close together that their branches met over the road of yellow brick. It was almost dark under the trees, for the branches shut out the daylight; but the travelers did not stop, and went on into the forest.

Media Queries

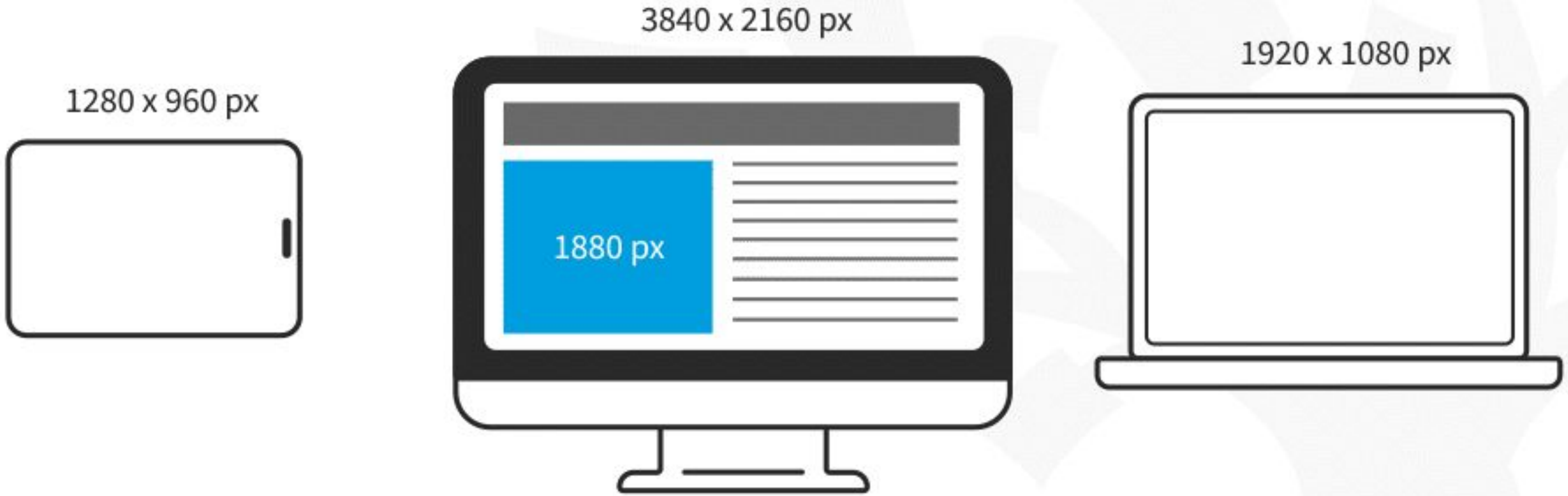
Media Queries

- We can use media queries to allow certain rules to apply for an iPad or iPhone, to add styles for a printer, or to create a responsive site.
- With media queries, we can allow most of our styles to remain the same, while we make small tweaks for specific formats.

Media Queries

Media queries are the crux of responsive design. They define the breakpoints at which different CSS rules are applied. Any CSS rule can be adjusted within the media queries.

Media Queries



Media Queries

Viewport

The first step in enabling media queries is to set the viewport so that pixels are rendered based on your CSS rules regardless of device orientation or screen size.

This is because some devices render pixels differently based on their resolution.

Media Queries

Viewport

You have to tell the browser you want the page to be responsive. Include this code above the head. This tells the browser to scale the page to the device that’s viewing it.

```
<head>
  <meta name=”viewport” content=”width=device-width, initial-scale=1”>
</head>
```

Media Queries

Viewport

- The viewport meta tag controls how a webpage is displayed on a mobile device.
- Without the tag, mobile devices will assume you want the full desktop experience and will set the viewport width at 980px (iOS)

device-width – This tells the browser "My Website adapts to your width"
initial-scale – Sets the initial zoom level and prevents default zooming

Media Queries

Syntax

```
@media screen and (max-width: 600px) {
  section {
    width: 50%;
  }
}
```

Media Queries

Syntax

Place your media queries at the bottom of your stylesheet so they can override any code that comes before them.

If you have multiple media queries, they should go from largest to smallest device.

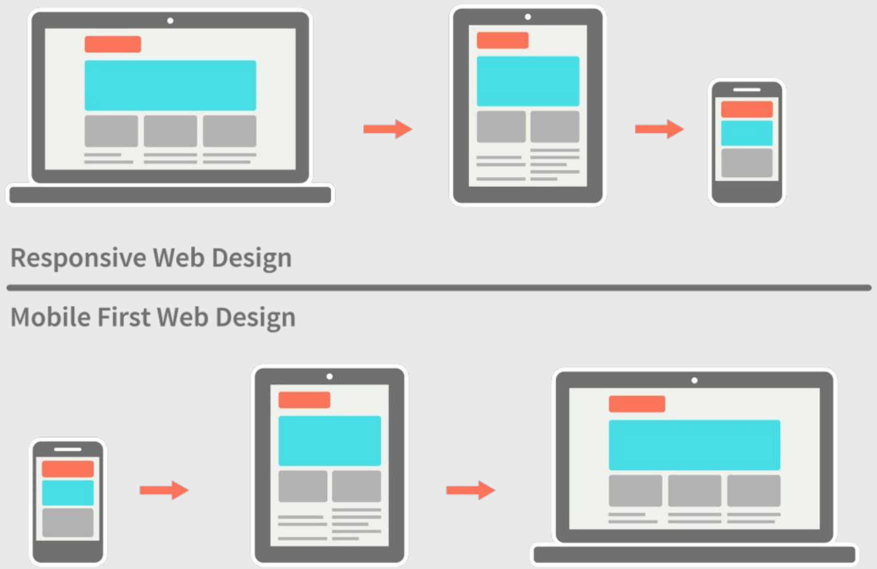
```
@media screen and (max-width: 600px){
  section {
     width: 50%;
  }
}

@media screen and (max-width: 400px){
  section {
     width: 100%;
  }
}
```

Media Queries

You can approach media queries in two ways:

1. Desktop first, defining max-width breakpoints to go smaller.
2. Mobile first, defining min-width breakpoints to go larger.
   Sometimes you need to use min and max.



Media Queries

Desktop-First Media Queries

Anything within the media query applies to the **mobile layout**.

All CSS rules above the media query apply to the desktop layout.

```
@media screen and (max-width: 600px) {
  section,
  aside {
    width: 100%;
  }
}
```

Media Queries

Mobile-First Media Queries

Anything within the media query applies to the desktop layout.

All CSS rules above the media query apply to the mobile layout.

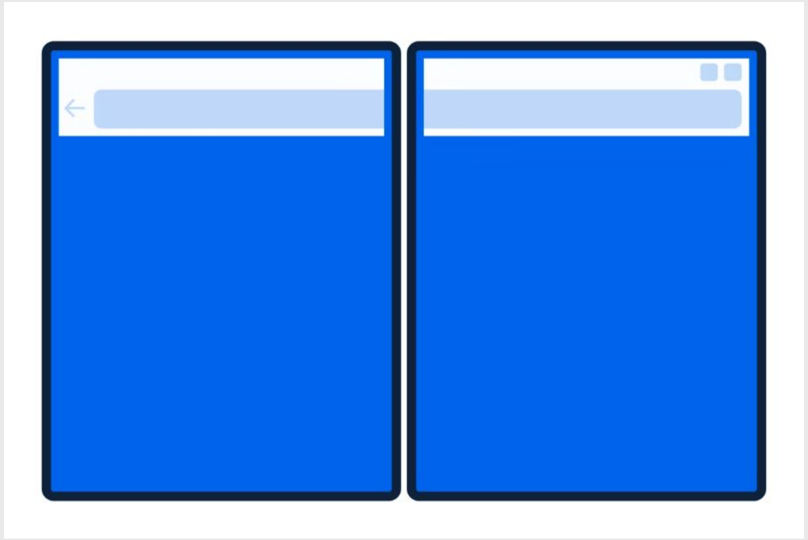
```
@media screen and (min-width: 600px) {
  section {
    width: 70%;
  }
  aside {
    width: 30%;
  }
}
```

Media Queries

Dual Screen & Foldables

Dual screen and foldable devices are the next step in responsive design, so they’re viewed as another responsive design target that we can style for with media features.

```
@media (horizontal-viewport-segments: 2) {}
```

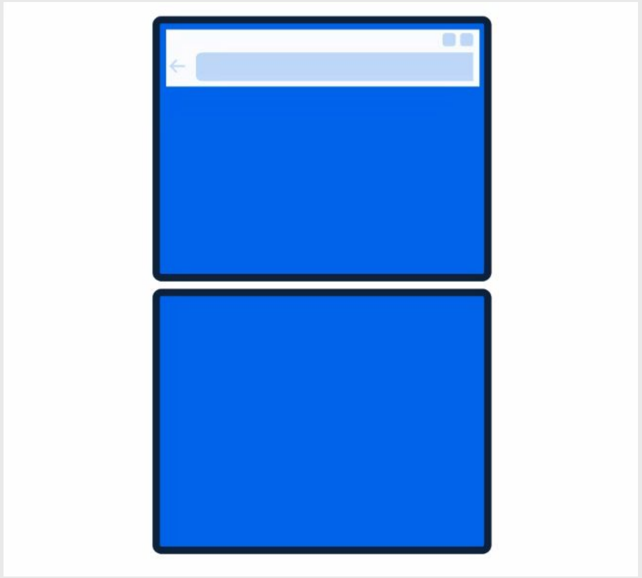


Media Queries

Dual Screen & Foldables

Dual screen and foldable devices are the next step in responsive design, so they're viewed as another responsive design target that we can style for with media features.

```
@media (vertical-viewport-segments: 2) {}
```



Guided Walk-Through:

Responsive Layout with media-queries

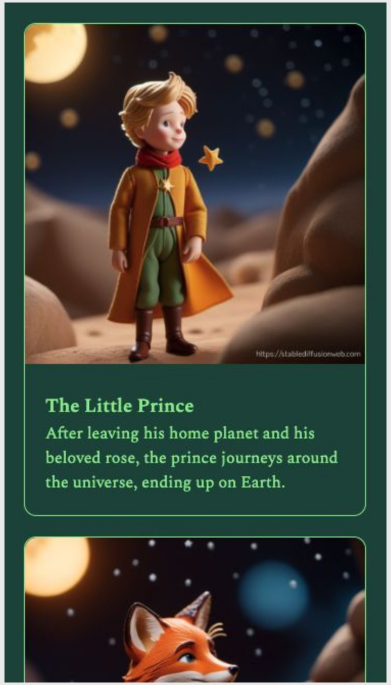
20 minutes

Apply **media queries** to achieve a responsive layout on the Little Prince webpage.

Instructions:

Go to the **starter_code_week_08** folder and complete the **responsive_with_media_queries** exercise.

1. Add two media queries having a min-width of 768px and 1024px to the bottom of the **styles.css** file.
2. The goal is to achieve a one-, two- and three-column layout.



The Little Prince

After leaving his home planet and his beloved rose, the prince journeys around the universe, ending up on Earth.

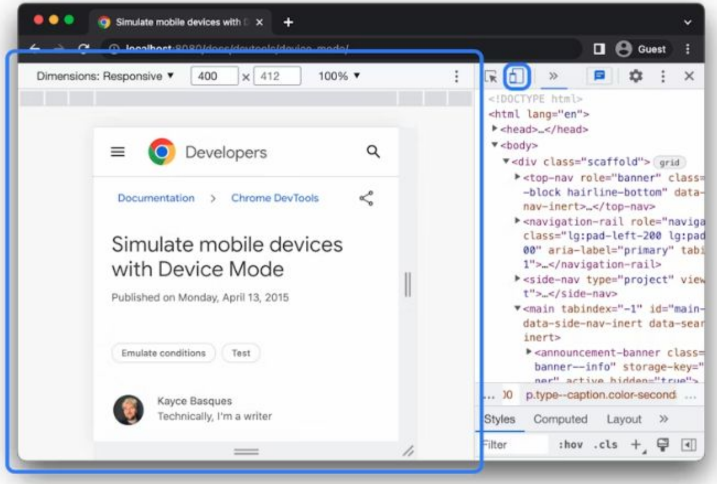
Simulating Mobile Devices

Device Mode

Simulate a mobile viewport

Use Device Mode to approximate how your page looks and performs on a mobile device.

Click Toggle Device toolbar to open the interface that enables you to simulate the mobile viewport.



Simulate mobile devices with Device Mode

Published on Monday, April 13, 2015

Kayce Basques
Technically, I'm a writer

```
<!DOCTYPE html>
<html lang="en">
 ▸<head>…</head>
 ▾<body>
   ▾<div class="scaffold"> grid
     ▸<top-nav role="banner" class=
      -block hairline-bottom" data-
      nav-inert>…</top-nav>
     ▸<navigation-rail role="naviga
      class="lg:pad-left-200 lg:pad
      00" aria-label="primary" tabi
      1">…</navigation-rail>
     ▸<side-nav type="project" view
      t">…</side-nav>
     ▾<main tabindex="-1" id="main-
      data-side-nav-inert data-sear
      inert>
       ▸<announcement-banner class=
        banner--info" storage-key="
```

p.type--caption.color-second

Mobile Friendly Navigation

Guided Walk-Through:

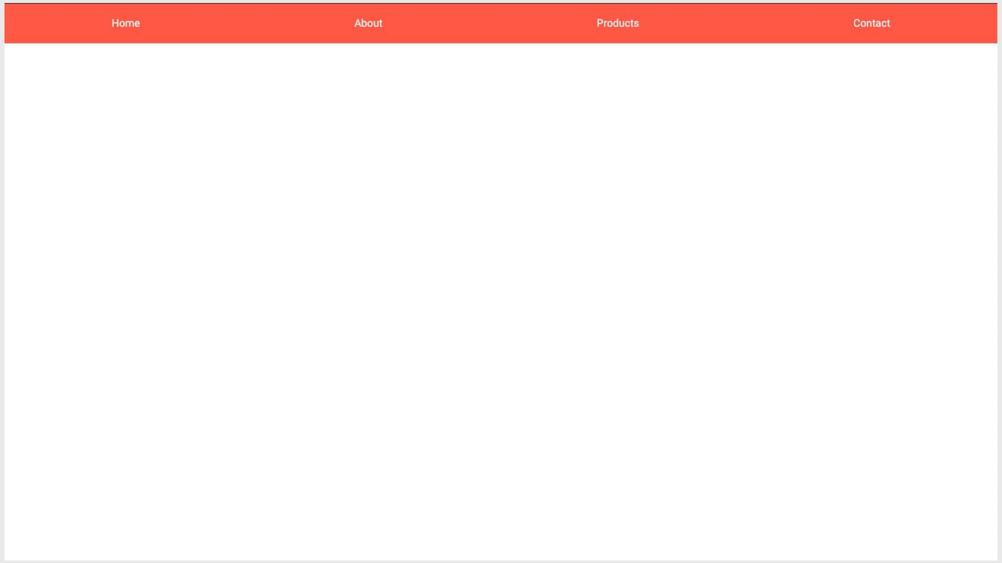
Navigation Bar with Flexbox

20 minutes

In this exercise, we'll build a navigation bar using flex properties. Look closely at the example image for spacing details.

Instructions:

Go to the **starter_code_week_06** folder and complete the **nav_bar_flexbox** exercise.



Responsive Typography

Responsive Typography

Percentage Height / Width

You don’t have to use pixels to size everything. Other units are available and have advantages. Percentages in particular pair very well with responsive layout elements.

Anywhere you can use pixels, you can use percentages.

```
.container {
  border: 1% solid black;
  margin: 5%;
  width: 80%;
}
```

Responsive Typography

Example: Percentage Width

Using percentage width containers is very common. Just remember, top-level containers are sized relative to the viewport, whereas child containers are sized relative to their parent container.

```
<section class=”container”>
  <div class=”child”>Stuff</div>
</section>

.container {
  width: 80%; /* 80% of the viewport
*/
}
.child {
  width: 50% /* 50% of its parent */
}
```

Responsive Typography

vh and vw units

You can also use percentages of the viewport called vh and vw. There are 100 viewport height and 100 viewport width units per any screen you view (which is very similar to percentages!).

```
.container {
  height: 100vh; /* full height of
screen */
  width: 50vh; /* half of width of
screen */
}
```

Responsive Typography

Percentages vs vh/vw Units

The biggest difference?

Nesting an element with percentage means it is bound by its parent. Viewport units will still be the same no matter how nested they are.

```
<section class=”container”>
  <div class=”child-1”>Percent</div>
  <div class=”child-2”>vw Units</div>
</section>

.child-1 {
  width: 50% /* 50% of its parent */
}
.child-2 {
  width: 50vw; /* 50% of the entire
screen*/
}
```

Responsive Typography

em Typography

- **em** is a unit of measurement derived from print design (based on the size of the letter “M,” which is the largest letter).
- It’s a sizing unit based on a parent element's size.
- If **.basic-class** is inside a container with **font-size: 16px;**, what size is the font in pixels?

```
.basic-class {
  font-size: 2em;
}
```

Responsive Typography

Sensible Usage

1. Set a specific value in px on body:

```
body {
  line-height: 12px;
}
```

2. Base other styles on it:

```
h1 { line-height: 2.5em; }  // 30px
h2 { line-height: 2em; }    // 24px
```

This can help establish a vertical rhythm to make reading and scrolling easier on your users' eyes.

Responsive Typography

Thinking Through It

Some people use em for layout properties, although ideally they're typographic. You can see why – if you just adjust the body, everything downstream adjusts.

```
body {
    font-size: 10px;
}

div {
    height: 5em; // 50px
    width: 3em; // 30px
    margin: 0.5em; // 5px
    border: 0.1em; // 1px
}
```

Responsive Typography

Avoid ems within ems

ems can quickly become a tangled mess of fractions on fractions when nested elements are concerned.

The example to the right shows just how tricky they can get!

```
<div class="container">
  <h2>How big is this title?</h2>
</div>

body {
  font-size: 10px;
}
div {
  font-size: 1.6em; // 10 x 1.6 = 16px
}
h2 {
  font-size: 1.1em; // 16 x 1.1 = 17.6px
  padding: 0.55em; // 16 x 0.55 = 8.8px
  margin: 0.2em; // 16 x 0.2 = 3.2px
}
```

Responsive Typography

The holy grail usage of **em** is book smart: Set a font-size on the body and set all layout sizes relative to it, then you can just reset body's size for responsive viewports and everything else follows accordingly.

This follows good code sense. Think about how many places you'd have to adjust your code to certain situations and minimize the amount of specifically "hard-coded" values that you might later have to change.

Benefits of Relative Units

```css
html { font-size: 16px; }
h1 { font-size: 33px; }
h2 { font-size: 28px; }
h3 { font-size: 23px; }
h4 { font-size: 19px; }
small { font-size: 13px; }
.box { padding: 20px; }

@media screen and (min-width: 1400px) {
  html { font-size: 20px; }
  h1 { font-size: 41px; }
  h2 { font-size: 35px; }
  h3 { font-size: 29px; }
  h4 { font-size: 24px; }
  small { font-size: 17px; }
  .box { padding: 25px; }
}
```

```css
html { font-size: 1em; }
h1 { font-size: 2.074em; }
h2 { font-size: 1.728em; }
h3 { font-size: 1.44em; }
h4 { font-size: 1.2em; }
small { font-size: 0.833em; }
.box { padding: 1.25em; }


@media screen and (min-width: 1400px) {
  html { font-size: 1.25em; }
}
```

Exercise