Responsive Design

A Learning Spike

Ryan Morris

About us

Who am I?

Who are you?

- Name?
- What you do here?
- Technologies you’re into?
- Experience with CSS or programming?
- Experience in design?

Quick Poll

- How many of you are Product Managers or Designers?
- How many of you are Java developers? C#, .Net?
- How many of you are web developers? Full-stack? Back-end?
- What about CSS? Is it new to you? Familiar? Expert level?
- Any experience with Javascript? TypeScript?
- Any experience with front-end frameworks? FoundationUI?

How we're going to work

- Your questions are welcome, anytime!
- Being a web developer/designer requires *constant* learning
- My goal is to give you the tools to work efficiently with web technologies - We’re going to practice!
- As a result, we will be going through online docs very often - When in doubt, search!

Useful links

- Repository for all labs code + solutions: https://github.com/rm-training/responsive-design
- Link to these slides: <TBD>
- A server for our labs:
  - Chrome server: https://goo.gl/cUHJLY
  - Or Node & npm: https://nodejs.org/en/

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

Transitions & Animations

Going beyond

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

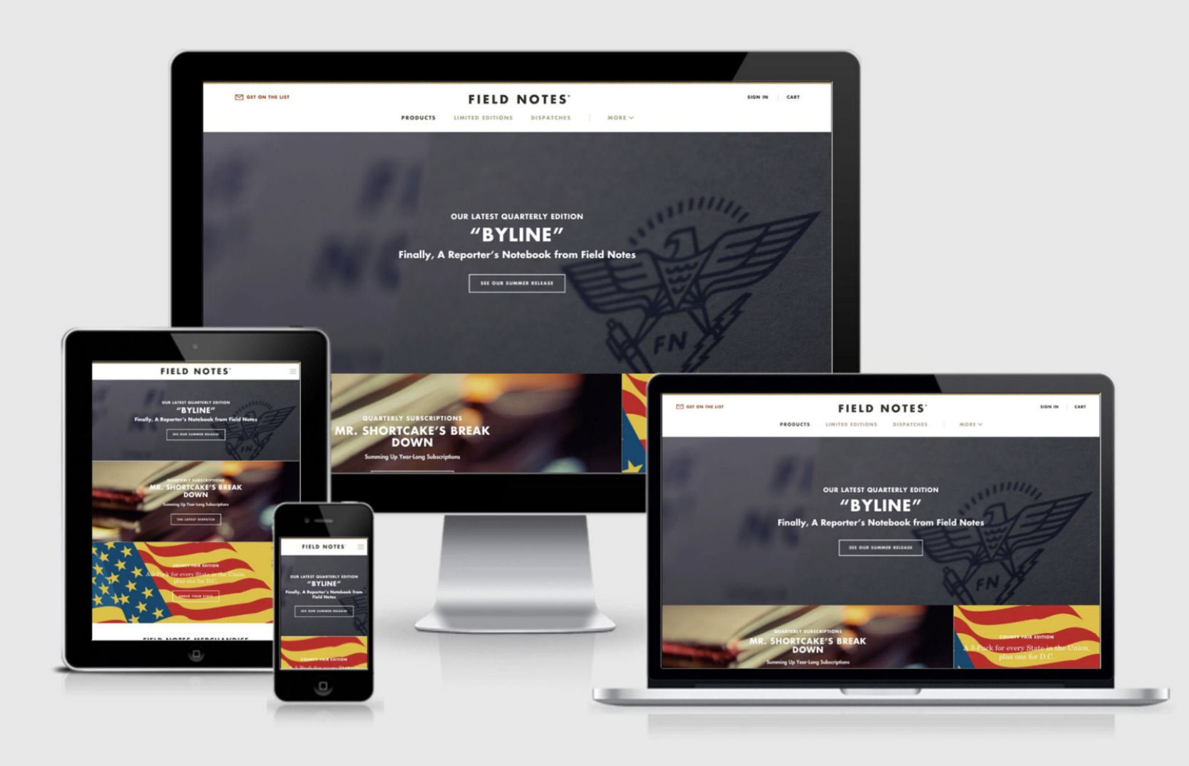
Transitions & Animations

Going beyond

Responsive Web Design

What is responsive web design?

It’s all about having web applications show up properly on all kinds of devices (laptop, PC, tablet, phones, watches)


GET ON THE LIST
FIELD NOTES
SIGN IN
CART
PRODUCTS
LIMITED EDITIONS
DISPATCHES
MORE
OUR LATEST QUARTERLY EDITION
“BYLINE”
Finally, A Reporter’s Notebook from Field Notes
SEE OUR SUMMER RELEASE
QUARTERLY SUBSCRIPTIONS
MR. SHORTCAKE’S BREAK DOWN
Summing Up Year-Long Subscriptions
COUNTY FAIR EDITION
A 3-Pack for every State in the Union, plus one for D.C.

Responsive design approach

- An **adaptive design** will adjust the layout of the page based on the device that is viewing it.

  This usually requires more CSS + code to handle the different layouts.

  A **mobile separate** approach can be seen as adaptive, but the alternative layout is determined before the page is served to the user.

- A **responsive design** will adjust to fit the device without requiring layout changes, scaling sizes as needed with set break-points.

  This usually requires jumping through more hoops.

Why does responsive design matter?

- Search Engine Optimization
- Improved mobile experience
- Visitors expect a seamless flow of access to content/services across many devices
- Future proofing for a world of devices
  Smart watches, car dashboards, etc

How to implement responsive web design?

- We’ll allow our layouts to adapt to the device and its functionalities using:
  - Fluid, proportional grids
  - Fluid Units
  - Flexible Media (Images, Videos)
  - Conditional Media Queries

Principles of Responsive Design

- Design **Mobile-first** or **Desktop-first**
- Consider **Progressive Enhancement** or **Graceful Degradation**
- Always write **Unobtrusive JavaScript**
- Be **Accessible**
- Always be **fast**
- **Content** is king (Content-first)
- **Don’t dumb down the experience**

Mobile Myths

- MOBILE ≠ RUSHED
- MOBILE ≠ LESS
- COMPLEX ≠ COMPLICATED
- TAP QUALITY > TAP QUANTITY
- NO SUCH THING AS MOBILE WEB
- FOCUS FOR ALL PLATFORMS
- DON’T THINK APP; THINK SERVICE
- METADATA IS THE NEW ART DIRECTION

Credit to Josh Clark

All this in practice

- Design for the **lowest common denominator**
- **Minimal HTML**. CSS and JavaScript separated from the HTML.
- **All content is "accessible"** - title, alt attributes. Form fields tab order is established.
- **Content is Semantic** - use of HTML to reinforce the meaning of the page
- Enhanced layout is in the **CSS** and the enhanced experience is in the **JS**
- **Avoid disruptive UI** (no landing page popups)
- End-user browser **preferences are respected**

navigation

A case study

Navigation

Show/Hide toggle

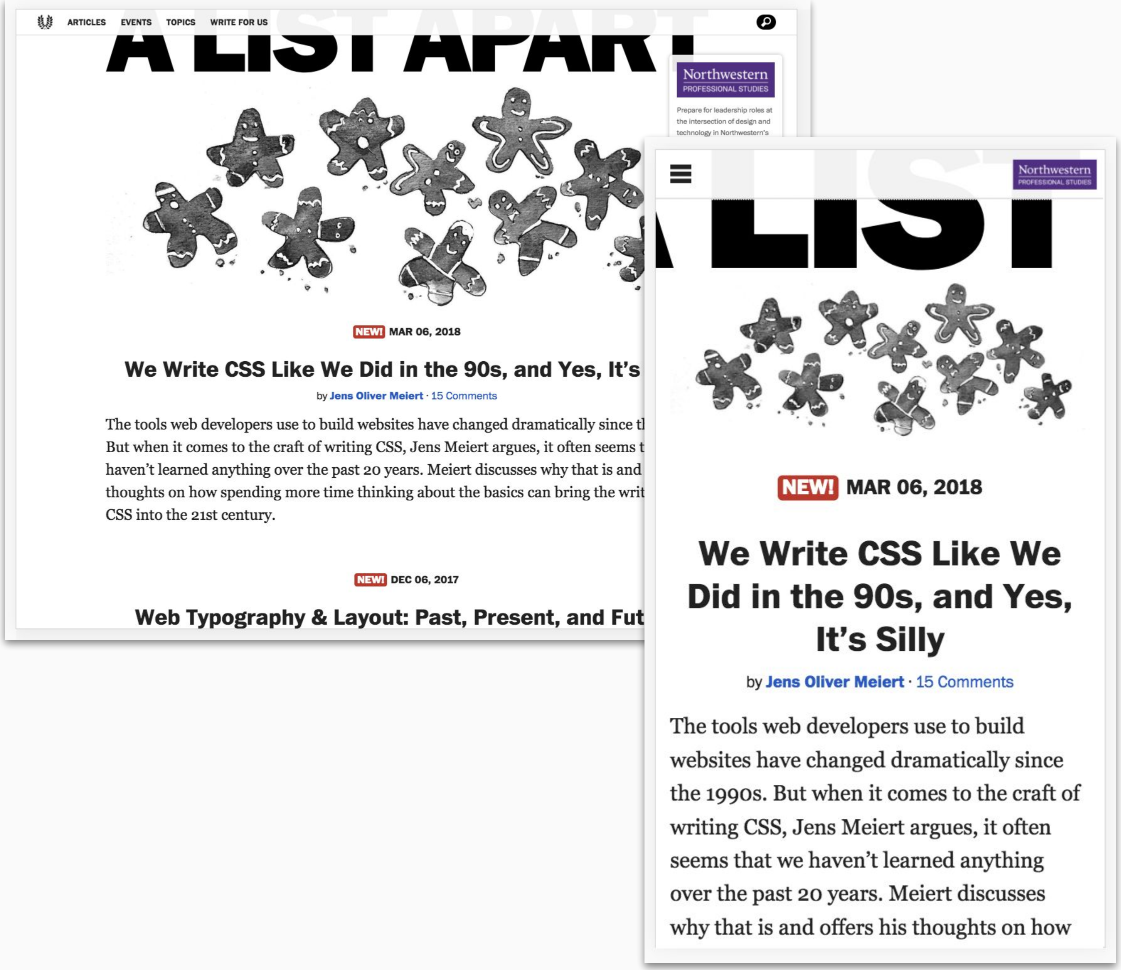
Off-canvas menu

Hamburgers

Conditionally loaded menus

Progressive Reveal

Adaptive Layout



Prepare for leadership roles at the intersection of design and technology in Northwestern's

NEW! MAR 06, 2018

We Write CSS Like We Did in the 90s, and Yes, It's

by Jens Oliver Meiert · 15 Comments

The tools web developers use to build websites have changed dramatically since t
But when it comes to the craft of writing CSS, Jens Meiert argues, it often seems t
haven't learned anything over the past 20 years. Meiert discusses why that is and
thoughts on how spending more time thinking about the basics can bring the writ
CSS into the 21st century.

NEW! DEC 06, 2017

Web Typography & Layout: Past, Present, and Fut

NEW! MAR 06, 2018

We Write CSS Like We Did in the 90s, and Yes, It's Silly

by Jens Oliver Meiert · 15 Comments

The tools web developers use to build websites have changed dramatically since the 1990s. But when it comes to the craft of writing CSS, Jens Meiert argues, it often seems that we haven't learned anything over the past 20 years. Meiert discusses why that is and offers his thoughts on how

Navigation

Show/Hide toggle

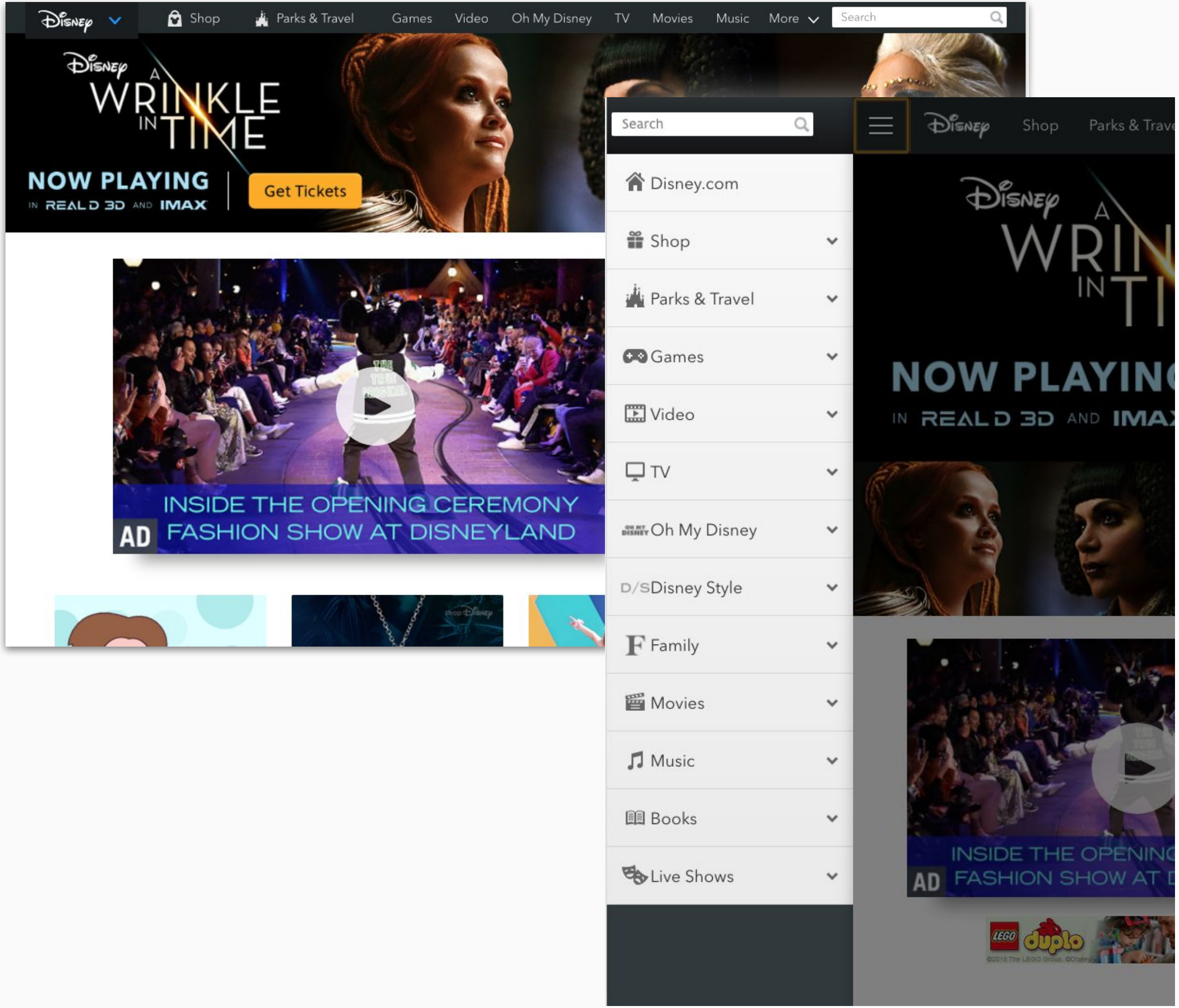
Off-canvas menu

Hamburgers

Conditionally loaded menus

Progressive Reveal

Adaptive Layout



Navigation

Show/Hide toggle

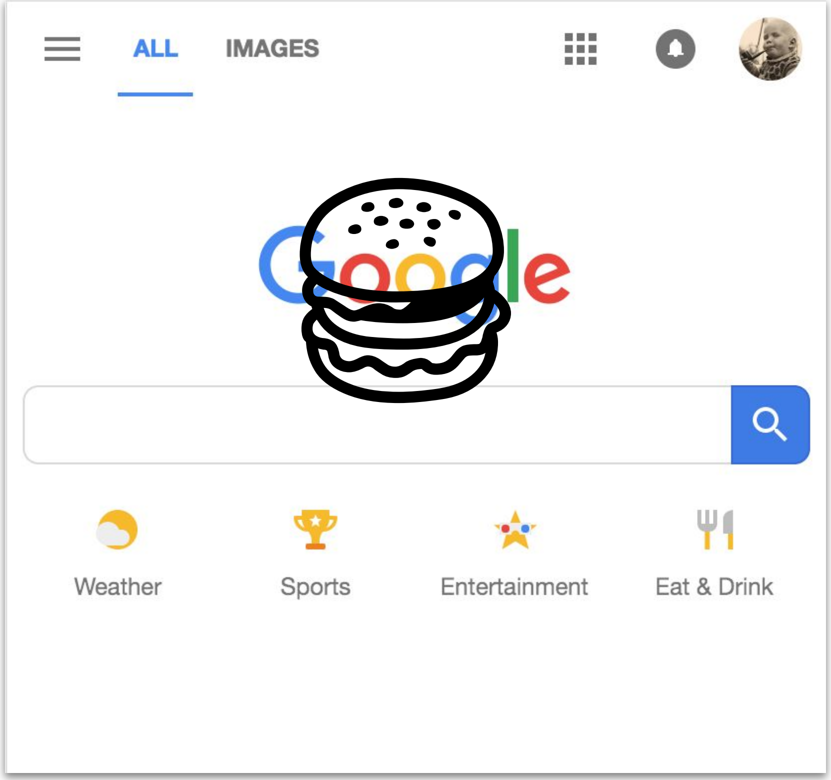
Off-canvas menu

Hamburgers

Conditionally loaded menus

Progressive Reveal

Adaptive Layout



Navigation

Show/Hide toggle

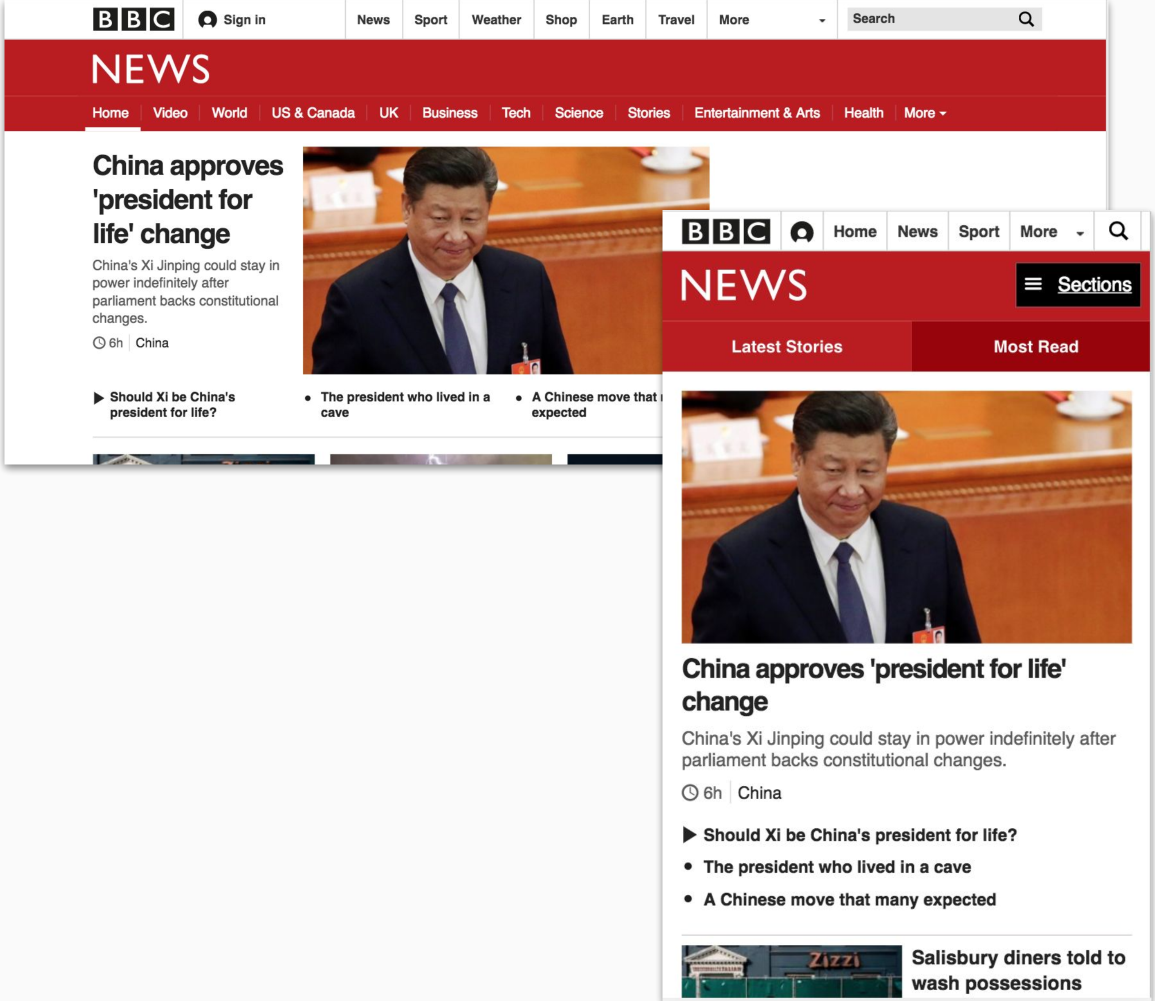
Off-canvas menu

Hamburgers

Conditionally loaded menus

Progressive Reveal

Adaptive Layout



China approves 'president for life' change

China's Xi Jinping could stay in power indefinitely after parliament backs constitutional changes.

6h | China

- Should Xi be China's president for life?
- The president who lived in a cave
- A Chinese move that expected

China approves 'president for life' change

China's Xi Jinping could stay in power indefinitely after parliament backs constitutional changes.

6h | China

- Should Xi be China's president for life?
- The president who lived in a cave
- A Chinese move that many expected

Salisbury diners told to wash possessions

Navigation

Show/Hide toggle

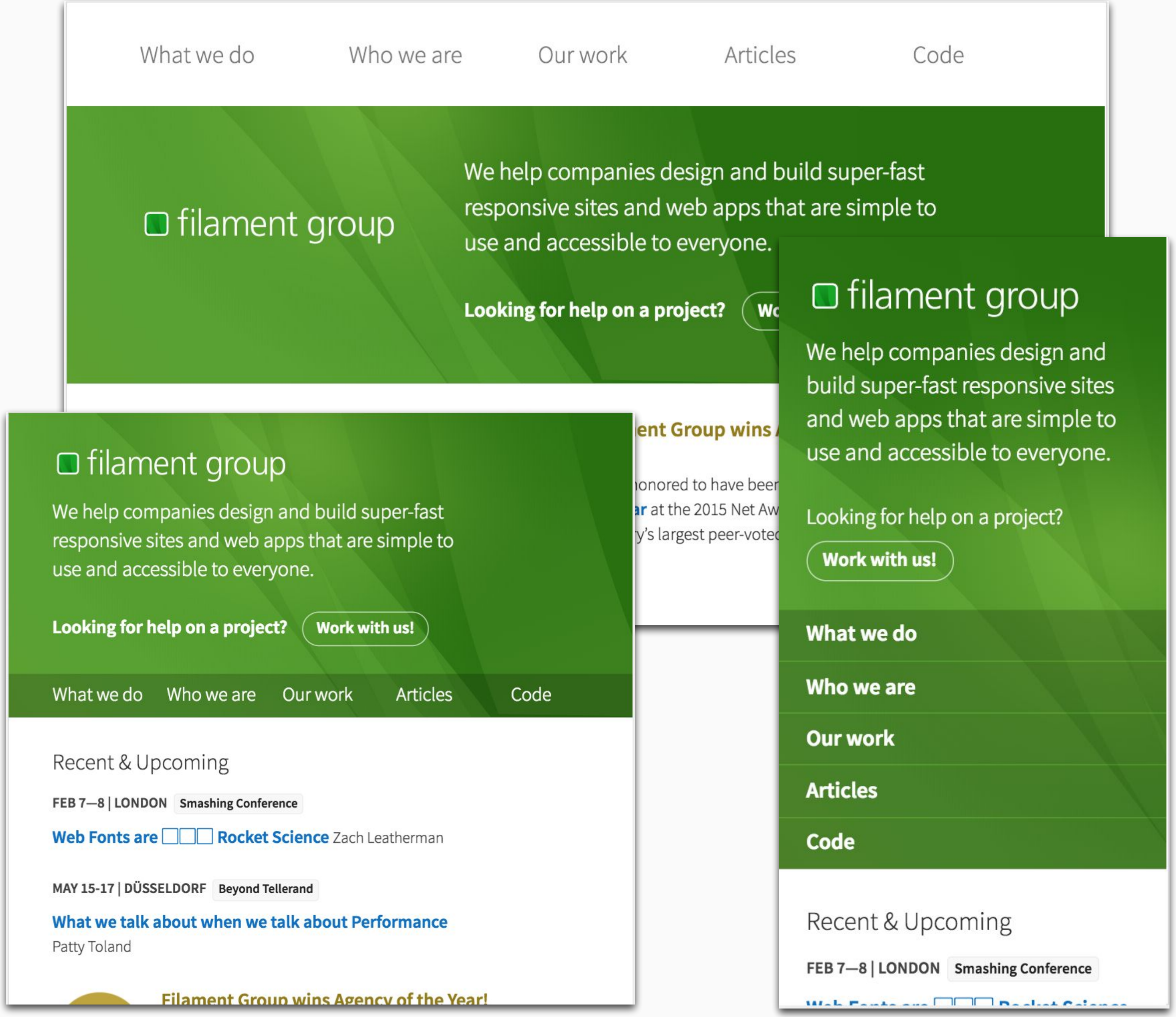
Off-canvas menu

Hamburgers

Conditionally loaded menus

Progressive Reveal

[Adaptive Layout]()



We help companies design and build super-fast responsive sites and web apps that are simple to use and accessible to everyone.

Looking for help on a project? Work with us!

Recent & Upcoming

FEB 7—8 | LONDON Smashing Conference

Web Fonts are □□□ Rocket Science Zach Leatherman

MAY 15-17 | DÜSSELDORF Beyond Tellerand

What we talk about when we talk about Performance Patty Toland

Filament Group wins Agency of the Year!

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

Transitions & Animations

Going beyond

Responsive Styling

Responsive Styling

- HTML Viewport meta tag
- CSS Responsive Units
- CSS Media Queries
- Build the page with a flexible structure or framework
- Use a framework that supports flexible layouts out of the box, such as FoundationUI or Bootstrap

Viewport Meta

- Meta tag placed in **`<head>`** of your HTML
- Introduced by Apple in iOS mobile
- **`width`** => width of the viewport at 100%
- **`initial-scale`** => controls the zoom level when loaded (1 is 100%)

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

See examples/viewport-meta.html

Responsive units

- CSS units that scale relative to the size of other elements or the viewport
  - Percentages (%)
  - Ems (em)
  - Viewport Units (vw, vh)
  - Pts & Px?
    - Not traditionally scalable though the browser will usually handle zooms
    - Won’t handle user’s browser preferences when resizing

See examples/responsive-units.html

Responsive units - Percentages

- Percentages are relative to outer container

```css
/* 10% of the parent container’s width */
width: 100%;

/* 50% of the parent container’s height */
height: 50%;

/* 50% of outside container width*/
width: 50%;
```

Responsive units - Ems

- **`em`** is relative to the **`font-size`** of the container element
    - Base font-size of a document is 16px, so 1em = 16px
    - Because of this they can be hard to follow in deeply nested elements
- Root ems (**`rem`**) are the top-level container's font-size

```
body {font-size: 16px} /* browser default */

div {font-size: 1.2em; /* 1.2 x 16px = 19.2px */ }

h1 {font-size: 1.2em; /* 1.2 x 19.2px = 23.04px */ }
h2 {font-size: 1.2rem; /* 1.2 * 16px = 19.2px */ }
```

Responsive units - Viewport Units

- Viewport units (**`vw`**, **`vh`**, **`vmin`**, **`vmax`**)
- It is a percentage of the full viewport (1=1%)
- Careful, it includes scrollbars in the width/height

```
width: 10vw; /* 10% of viewport width */
height: 100vh; /* 100% of viewport height */

veight: 10vmin; /* 10% of smallest viewport dimension*/
```

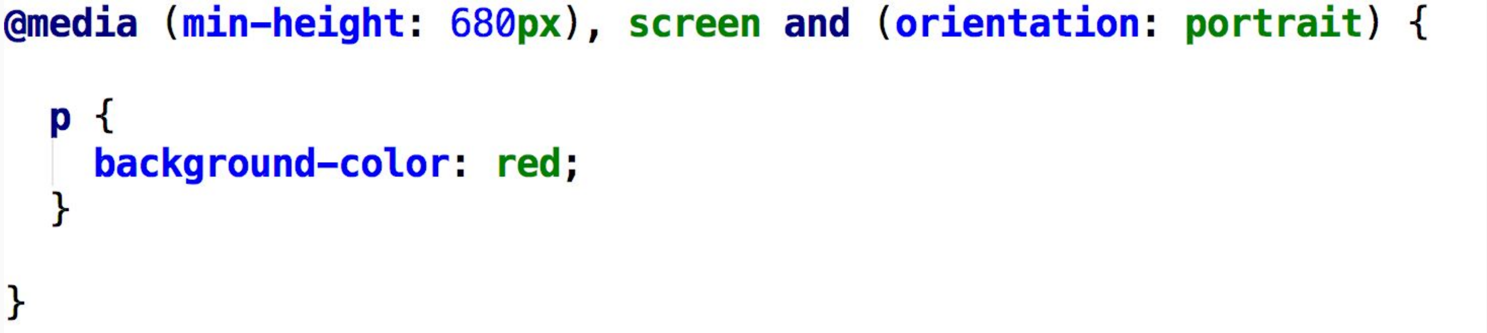
CSS calculations

- Natively perform simple calculations for css property values
- Can handle simple expressions with +, -, *, /
- Can be nested with parenthesis

```
/* responsive font sizing */
font-size: calc(16px + 0.5vw)

/* responsive width */
width: calc(100% - 1rem);
width: calc(100% / 6);
```

Media Queries

- CSS supports media queries, a way to apply CSS rules based on screen sizes and constraints:

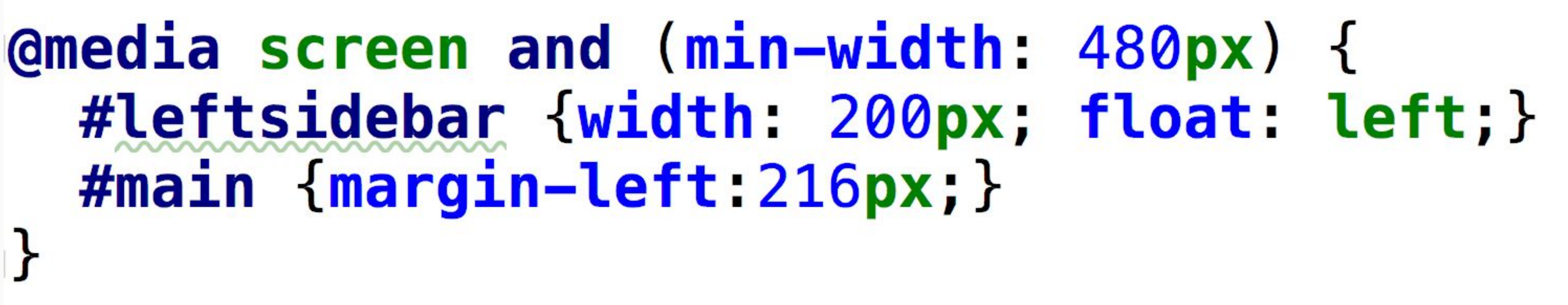


```
@media (min-height: 680px), screen and (orientation: portrait) {

  p {
    background-color: red;
  }

}
```

- The above code would turn all paragraphs to a red background color for devices that have a *height of a at least 680 pixels* and a *portrait orientation*

Query by range

- Media queries can also be used to customize any kind of CSS rules based on the screen size. We can specify a **`min-width`**, **`max-width`**, **`min-height`**, **`max-height`**:

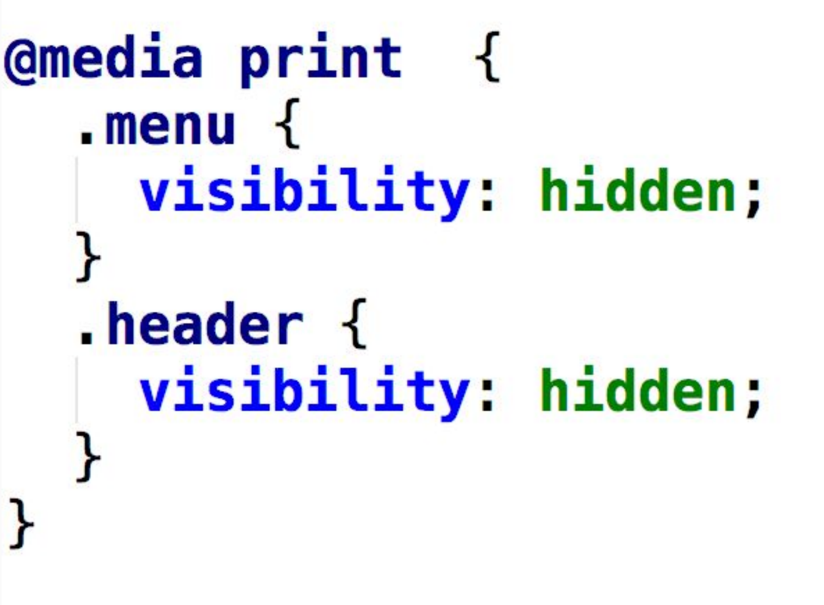


```
@media screen and (min-width: 480px) {
  #leftsidebar {width: 200px; float: left;}
  #main {margin-left:216px;}
}
```

- **`min-width`** => “when width is greater than or equal to this”
- **`max-width`** => “when width is less than or equal to this”

Query by device

- Media queries can also be used to customized the way a page looks like when it gets printed:



```
@media print  {
  .menu {
    visibility: hidden;
  }
  .header {
    visibility: hidden;
  }
}
```

Query by feature

- You can query by features supported by a device, ie: does it support hovers or is it color?

```
/* does device support hover events? */
@media (hover: hover) {}

/* does device support color? */
@media (color) {}
```

Combine queries

- Combine conditions with *commas*, **`not`**, **`and`** and **`only`**
  - **`not`** will negate the entire combined query
  - **`only`** will be ignored by browsers that don’t support media queries
  - **`not`** & **`only`** require the “media type”, ie: **`all`**

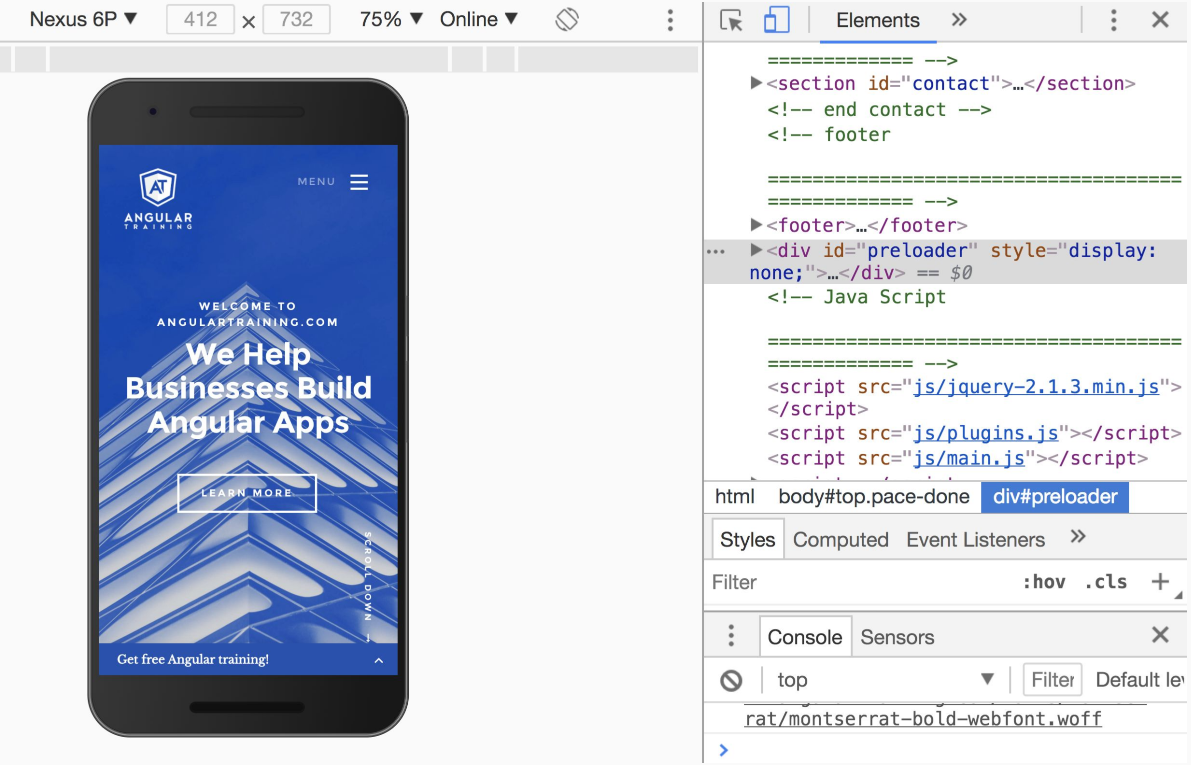
```
/* all queries must match */
@media (min-width: 30em) and (orientation: landscape) { }

/* any of these must match -- like an OR */
@media screen, print, (min-width: 30em) { }

/* not a color screen, or a color print */
@media not screen and (color), print and (color) { ... }
```

How to test responsive web design?

Modern browsers have developer tools to simulate the rendering of a webpage on any kind of device

You may also want to force "touch" event sensors.

Settings > Sensors > Force Touch > Enabled



```
  ============= -->
▶<section id="contact">…</section>
  <!-- end contact -->
  <!-- footer

  ====================================
  ============= -->
▶<footer>…</footer>
▶<div id="preloader" style="display:
none;">…</div> == $0
  <!-- Java Script

  ====================================
  ============= -->
  <script src="js/jquery-2.1.3.min.js">
  </script>
  <script src="js/plugins.js"></script>
  <script src="js/main.js"></script>
```

```
rat/montserrat-bold-webfont.woff
```

Lab 1 - Media queries

- In this lab, we’re going to make a simple web page responsive using media queries

- The webpage is in your lab folder: **`1-not-responsive.html`**

- When you resize your screen, you’ll see the current layout looks pretty bad

- **Your mission:** Use a media query to change the layout when the screen is less than *1024 pixels wide*. In that case, the three boxes should get displayed as a stack instead of side-by-side.

- What happens when you remove the viewport meta?

What we just learnt

Responsive web design is a way to make our web pages display differently on different kinds of devices

We can use media queries to have custom CSS for different screen sizes and orientations

Browsers provide developer tools to emulate the rendering of our website on different devices

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

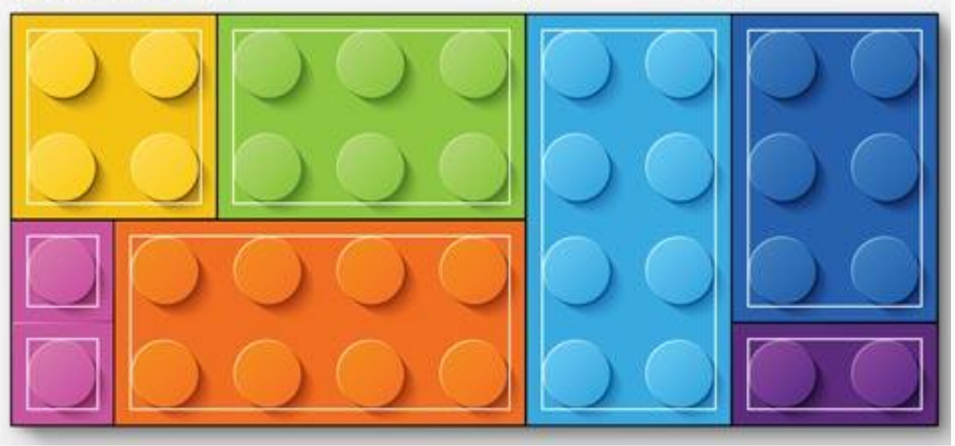
Transitions & Animations

Going beyond

Grids & FlexBox

Page Layout

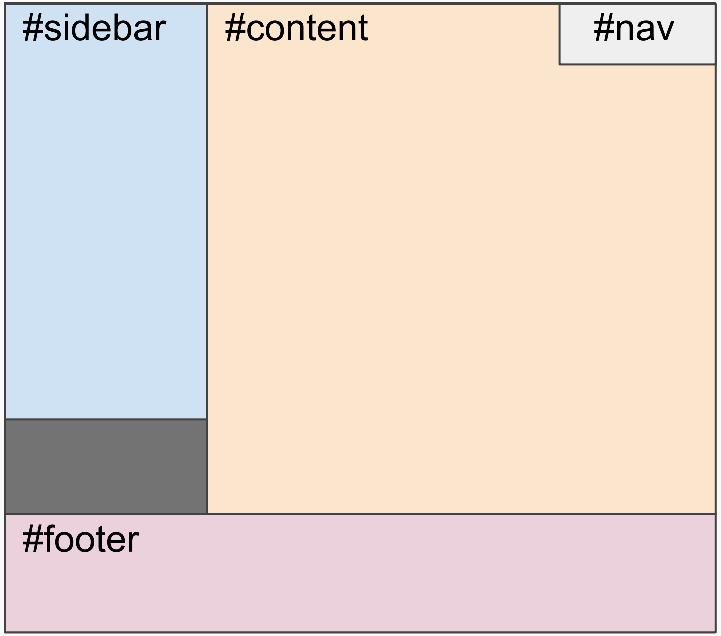
- Traditionally page would be designed as a whole - a **one-page design** - but a more **modular design** approach lends itself towards responsive results
- “A network of small layout systems”, **micro-layouts**, can be put together like legos to build up the page differently based on the device
- Resulting in **pattern libraries** & **style guides**



CSS for Layout

- Layouts on the web were traditionally limited to complex workings of **float** and **position**

```
#nav{
  position: absolute;
  top: 0; right: 0;
}
#content {
  float: right;
  width: 70%;
}
#sidebar {
  float: left;
  width: 30%;
}
#footer {clear: both;}
```



See examples/layouts-float.html

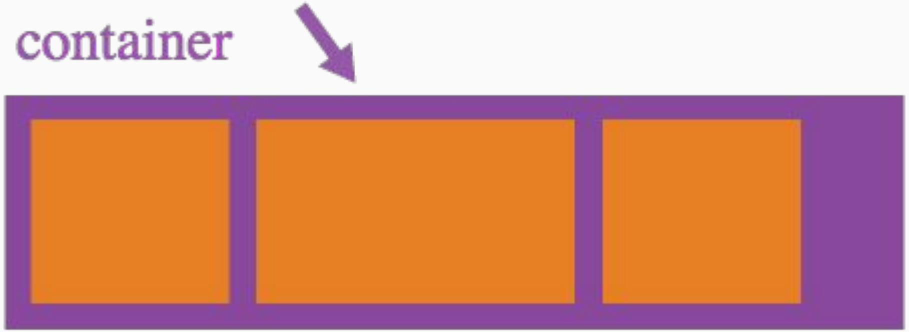
Modern CSS Layouts

- CSS has new layout options for better, native control over how a page lays out including **multi-columns**, built-in **grids** and the new **flexible box** layout

Flexible Box Layout

- Ability to control how elements fill height and width within a container

```
.container {
   display: flex;
}
```

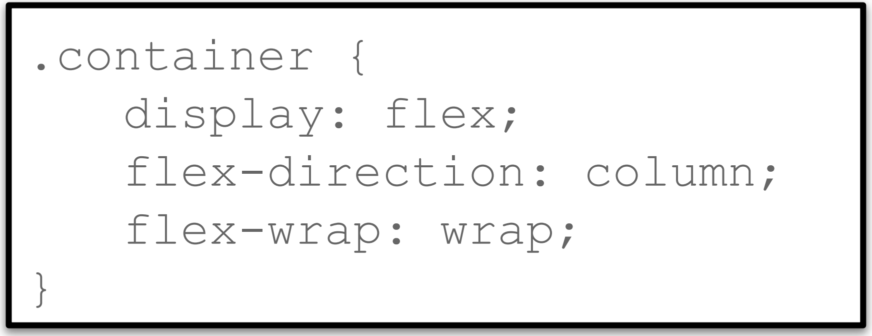


- We define a container to use the flex display mode and then specify how it’s child elements should flow (direction), align, and expand (grow)

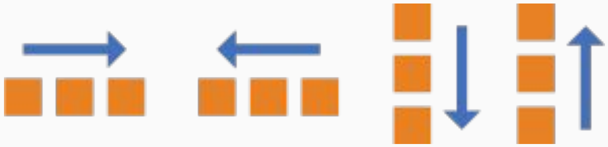
See examples/layouts-flex.html

Flex box container

- It is is a one-dimensional layout, meaning it is limited to one direction at a time. Either horizontal (default) or vertical.
- **flex-direction** controls the *axis* of flow
- **flex-wrap** controls whether items should wrap to the next line



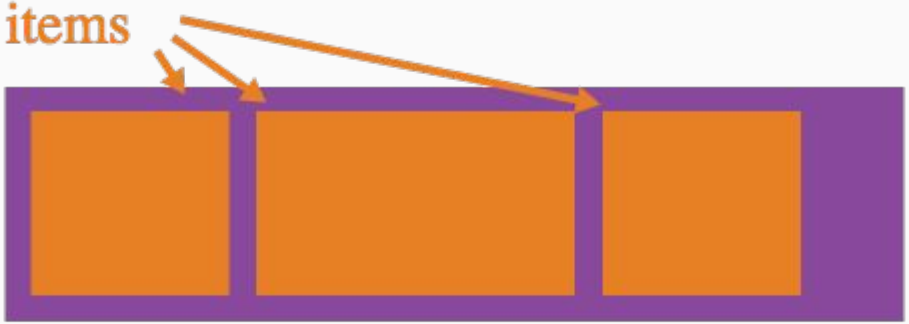
```
.container {
   display: flex;
   flex-direction: column;
   flex-wrap: wrap;
}
```



Flex box children

- You also control how the child items flow, align and resize to fill the container

```
.item {
    flex-grow: 1;
    flex-shrink: 0;
    flex-basis: 50px;
}
```



- **flex-grow** defines how much the item can expand if there is remaining space

- **flex-basis** can be used to set a minimum width

Flex FTW

- We can get pretty nice layouts from just flex (no more floats!)
    - A nicely spaced out menu
    - A one-dimensional grid of cards
    - Centered content (finally)

See examples/layouts-flex-full.html

Lab 2 - Flex Box

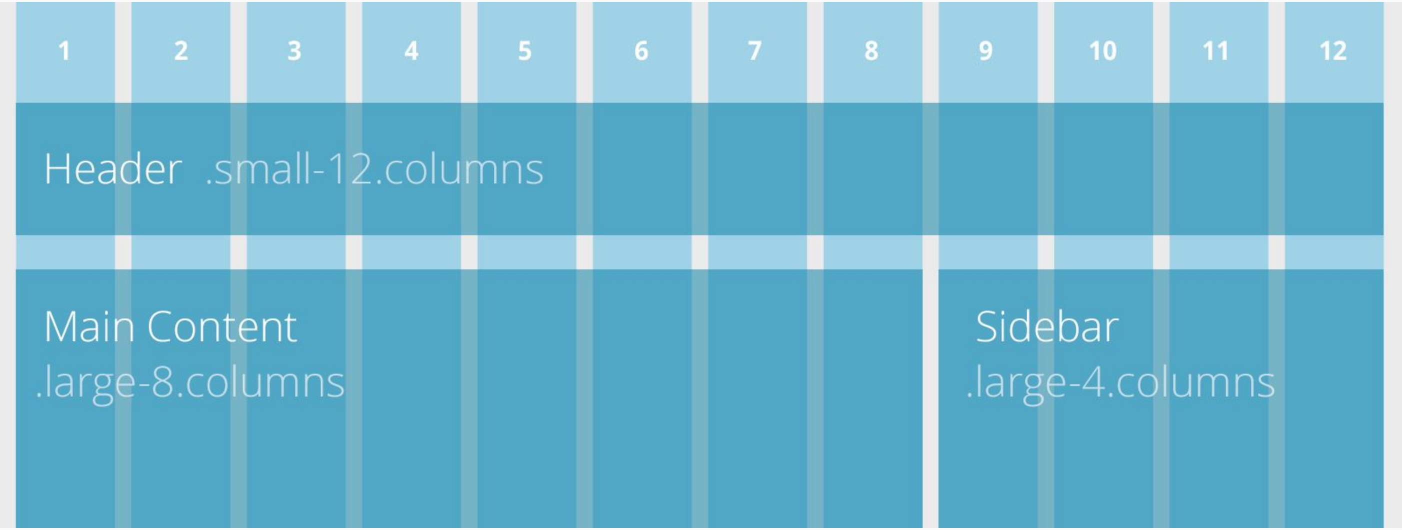
- Build two simple micro layouts using just flex box
- The webpage is in your lab folder: **`2-flex-items.html`**
- **Your mission:**
  A: Use flex box on the <nav> <ul> to make it so the <li> elements flow horizontally. **Bonus:** Make the ‘Sign in’ button aligned to the right.

  B: *Then* use flex box on the items container to get them to flow horizontally and wrap to the next line like a gallery of photos. Each “column” should be ¼ of the page width.

  Hints: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

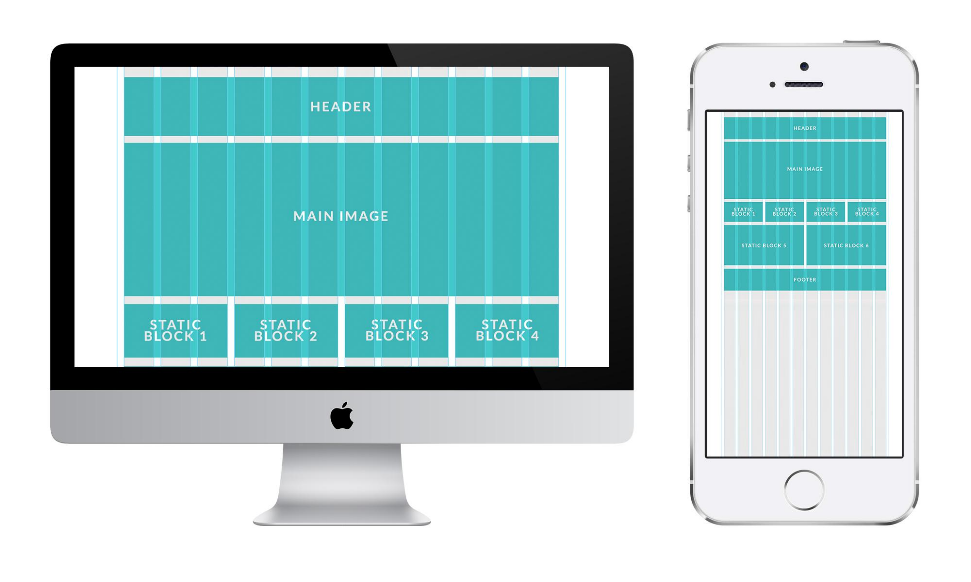
The grid

- **Grids** consist of a set number of **columns** and any number of **rows**
- **Gutters** can be specified to indicate spacing between columns
- Responsive grids will **stack** rows on a smaller device



Bad grid

- **Grids** are a way to help you lay out an app or site but shouldn’t be considered rigid structure
  - You can have sub-grids
  - You can design around combined columns (12-col at full size, down to 4-col at a smaller device)
- Don’t just resize your content



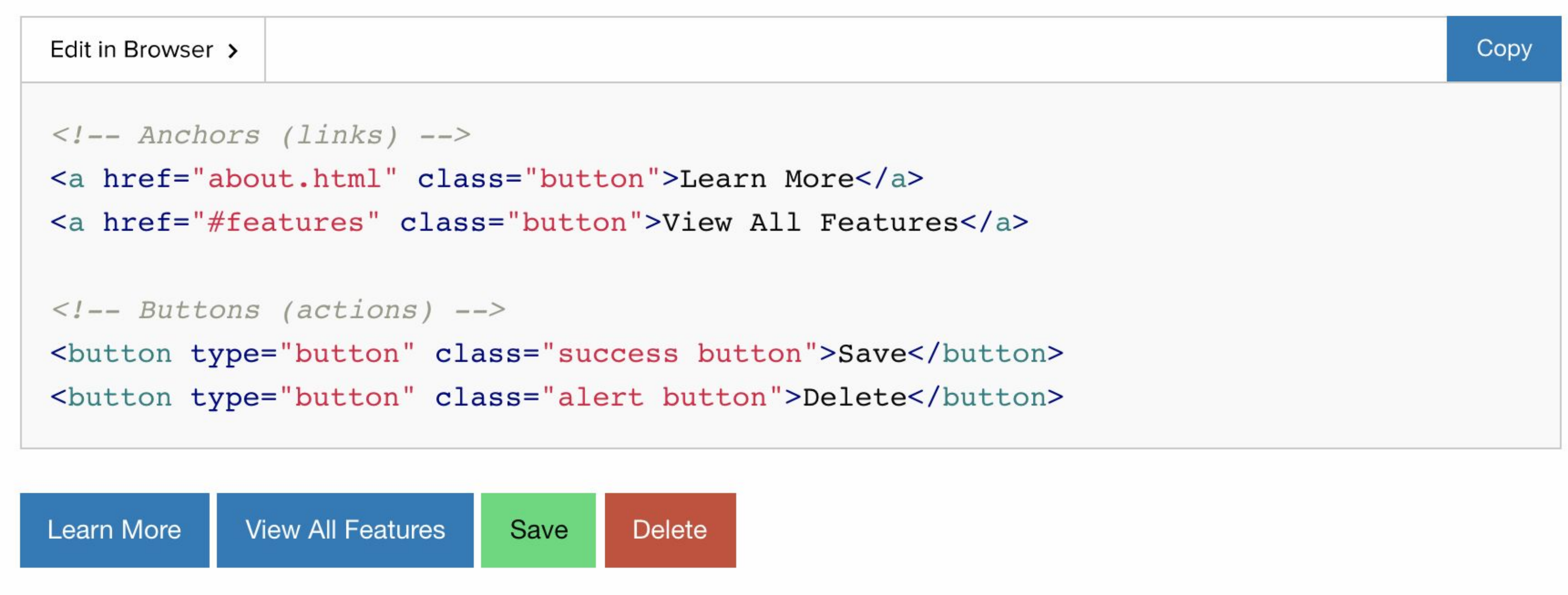
The Foundation UI framework

- Several UI frameworks embrace responsive design and give us predefined CSS classes and Javascript code to make our web applications responsive
- Foundation (https://foundation.zurb.com) is one of them
- Other examples include Bootstrap, Material Design, etc.

Let’s explore their documentation

The Foundation UI framework

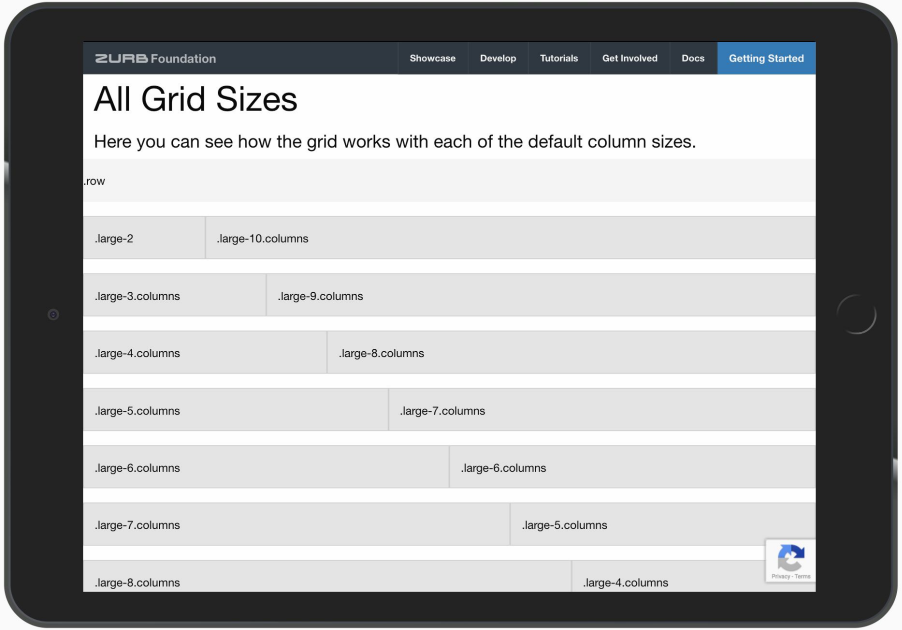
Here’s an example of Foundation buttons. All we need is use CSS classes from Foundation to get a nice look and feel:



```html
<!-- Anchors (links) -->
<a href="about.html" class="button">Learn More</a>
<a href="#features" class="button">View All Features</a>

<!-- Buttons (actions) -->
<button type="button" class="success button">Save</button>
<button type="button" class="alert button">Delete</button>
```

Responsive web design with the Foundation UI framework

- Foundation UI supports a 12 responsive column grid
- We can head to https://foundation.zurb.com/grid-1.html:

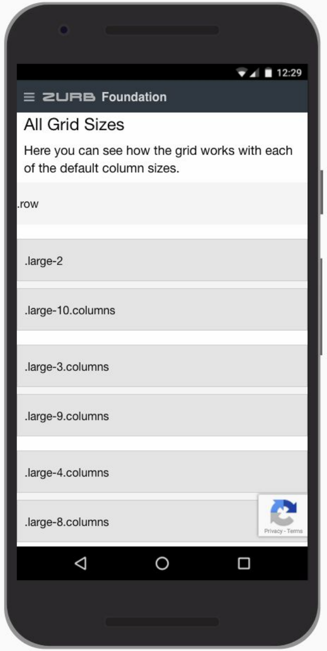


All Grid Sizes

Here you can see how the grid works with each of the default column sizes.

.row

.large-2	.large-10.columns
.large-3.columns	.large-9.columns
.large-4.columns	.large-8.columns
.large-5.columns	.large-7.columns
.large-6.columns	.large-6.columns
.large-7.columns	.large-5.columns
.large-8.columns	.large-4.columns



All Grid Sizes

Here you can see how the grid works with each of the default column sizes.

.row

.large-2

.large-10.columns

.large-3.columns

.large-9.columns

.large-4.columns

.large-8.columns

Foundation Grid (Basic)

- Three breakpoint sizes via media queries:
  small, medium (640+), large (1024)

```
<div class="row">
    <div class="small-6 medium-4 columns"></div>
    <div class="small-6 medium-8 columns"></div>
</div>

<div class="row">
    <div class="columns small-12 large-2"></div>
</div>
```

- There are actually three grids in Foundation:
  Float (v5), Flex (v6), XY-grid (v6.4)

Lab 3 - Responsive UI with foundation

- In this lab, we’re going to make our previous web page from **lab #1** responsive using grids from the Foundation framework

- **Your mission:** Use Foundation grids to have our different blocks get stacked on screens that are less than 640 pixels wide.

- **Hint:** In order to use Foundation CSS, import it using the following URL:

```
<link rel="stylesheet"
href="http://dhbhdrzi4tiry.cloudfront.net/cdn/sites/foundation.min.css">
```

Native CSS Grids

- CSS has released its own approach to grids with the **CSS Grid Layout**!
- **Support** is not 100% (caniuse.com)
- Follows similar patterns and syntax as flex-box but supports **two-dimensional** flow and has more controls
- Design guide here: https://css-tricks.com/snippets/css/complete-guide-grid/
- Learn CSS Grid in 7 Minutes: https://www.youtube.com/watch?v=ojKbYz0iKQE

CSS Grid Terminology

Container

Item

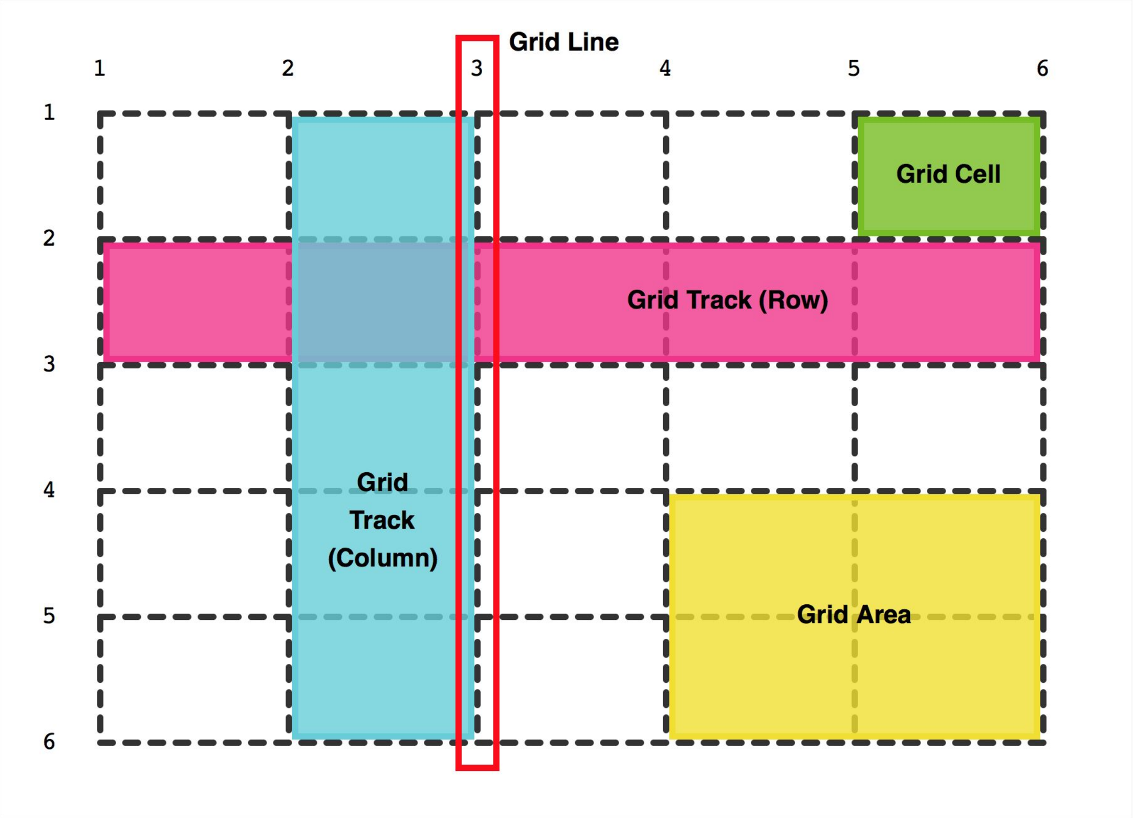
Line

Cell

Area

Track

Gap



Native CSS Grids

- Set any container to be a **grid**

```
display: grid;
```

fr is a new unit indicating a “fraction” of available space

- All first-level descendant elements will be the **items**

- Then define columns and/or rows

```
grid-template-columns: 1fr 1fr 1fr; /* three columns */
grid-template-rows: 2fr 1fr 1fr; /* three rows */
```

- Cells will automatically be laid out left-to-right

Native CSS Grids

- For each **item** you can declare where it sits in the **grid**

```
div.element {
    /* from column line 2 to column line 4 */
    grid-column: 2/4;

    /* from row line 2 to row line 3 */
    grid-row: 2/3;
}
```

- You can place elements anywhere - *no more spacers*
- But… always consider the order of content

Native CSS Grids

- You can even define **template areas** and assign elements to those areas

```
#container {
    grid-template-areas:
        “header header header”
        “menu main main”
        “menu footer footer”;
}

#header {grid-area: header;}
#menu {grid-area: menu;}
#footer {grid-area: footer;}
```

This makes media queries super simple - and the CSS ***visibly*** *declares layout*

See examples/layouts-css-grid.html

Going back to our first grid today

- How would we handle grids in 1-not-responsive.html?

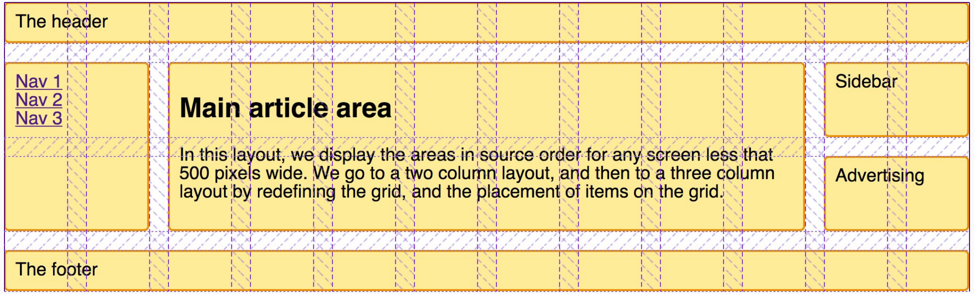
```
body {
 display: grid;
 grid-template-columns: 1fr 1fr 1fr;
 }
```

Additional grid functionality

- grid-gap
- repeat(12, 1fr);
- minmax()
- span

Lab 4 - CSS Grids

- We’ll be working with **4-css-grids.html**
- **Your mission:** Use CSS Grids to create a 12 column grid, then use grid-template-areas to lay out each element according to this template.



The header

Nav 1
Nav 2
Nav 3

Main article area

In this layout, we display the areas in source order for any screen less that 500 pixels wide. We go to a two column layout, and then to a three column layout by redefining the grid, and the placement of items on the grid.

Sidebar

Advertising

The footer

- **Hint 1**: You don’t need to define *each* 12 columns
- **Hint 2:** Use a *wrapper* container as the grid
- **Bonus:** Make it responsive using media queries so that the elements stack at a smaller width viewport

What we just learnt

We learnt about our options when defining layouts within CSS - with the right tools we can achieve just about anything

Floats are a thing of the past for layout.

Flexible Box layout is perfect for one dimensional flows.

CSS Grid is extremely powerful and is a good drop in replacement for any grid system.

You don’t “need” 12 columns ;)

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

Transitions & Animations

Going beyond

Icons & Web Fonts

Font family

- We can control the font a user sees with the css property **font-family**

```
font-family: "Times New Roman", Times, serif;
```

- But our choices are typically limited to web-safe fonts

Arial	sans-serif
Courier New	monospace
Georgia	serif
Times New Roman	serif
Trebuchet MS	sans-serif
Verdana	sans-serif

Web Fonts

- **Web Fonts** offer a way for us to use just about any font (supported since ie4)
- We can load and define a font from within our CSS

```
@font-face {
    font-family: "myFont"; /* we name it */
    src: url("myFont.ttf"); /* indicate the file source */
}
```

```
html {
    /* then we can reference it */
    font-family: "myFont", "Bitstream Vera Serif", serif;
}
```

Web Fonts

- Not all browsers support the same font formats - WOFF and TTF/OTF are best

  https://www.w3schools.com/css/css3_fonts.asp

- Alas, not all fonts are free

  Either we find a **license-free** font, we **create** one, pay for the **license** or use a **font service** such as Google Fonts or TypeKit

Icons - Images, Fonts or SVG

- Originally icons were an **<img>** tag and then possibly updated to using a **sprite**
  But... images do not scale nicely

- **Icon Fonts** allow us to display vector-based, colorable icons to the user

```
<i class="wi wi-umbrella" aria-label=”umbrella”>
    <span class="sr-only">"umbrella"</span>
</i>
```

- But they aren’t easily read by **screen readers** so we include text & aria-label

Scalable Vector Graphics

- SVG is a **markup representation** of a vector image

  Scalable, programmable, colorable & transformable elements

- An example: https://codepen.io/massimo-cassandro/pen/waOJWr

  Or see /images/businessman.svg

- We can either generate our own or use a service like FontAwesome or IcoMoon

Icon Fonts vs SVG

- Font Icons

  + Faster & light-weight

  + Wider browser support

  - Less customizable

- SVG Icons

  + Super Flexible - colorable, transformable

  + More semantic, more accessible

  - Not simple to use

Lab 5 - FONTS

- In this lab, we’re going to use Google Fonts to add fonts to our previous labs.

- **Your mission:** Create a font pack of ***at least two fonts*** from Google Fonts. Link to the provided style sheet and apply the fonts to your content so that headers (h1-h6) use one font while the rest of the content uses the other. Don’t forget to set fallback fonts in case those fonts do not load!

- **Bonus step:** Check your network panel to see the size of the downloaded css & font file(s)

- **Double bonus:** Install an icon pack from **FontAwesome** and toss some icons in the page

What we just learnt

We learnt how to include and customize font selections for our web pages.

Web Fonts allow us to reference and use just about any open-source or licensed font on the web.

Icons are typically handled as a Font for browser support, otherwise SVG is the way of the future.

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

Transitions & Animations

Going beyond

Transitions & Animations

Animating in the web

- Animations should be seen as a progressive enhancement
- Traditionally handled through css **pseudo selectors** like **hover**

```
#my-button:hover {
  background-color: red;
}
```

- Or by **JavaScript**

```
document.getElementById(‘my-button’).addEventListener(function(){
   this.style.backgroundColor = ‘red’;
});
```

Animating in the web

- CSS now supports a handful of animation options, allowing us to build up complex animations (in particular when used with SVG images)
- **Transitions** allow us to define properties that will animate when changed
- **Transforms** allow us to modify an element within 2d and 3d space
- And the new animate & **keyframe** functionality allows us to build more complex animations
- Examples: https://www.creativebloq.com/inspiration/css-animation-examples

CSS Transitions

- To use a **transition,** specify which css property will be animated when its value changes
- Any change to the property, either through a state change, class change or JavaScript manipulation, will appear animated

```
transition-property: background-color; /* or ‘all’ for all */
transition-duration: 0.3s;
transition-timing-function: ease;

/* short-hand*/
transition: background 0.3s ease 1s; /* prop dur ease delay */
```

See examples/animations-transitions.html

Alas...

Not all properties are animatable - see reference

CSS Transformations

- **Transformations** allow us to adjust the element in 2D or 3D space, changing its shape or position
- We’ll use a new property, **transform**, along with transformation functions to set the value(s) of our transformations.

```
transform: scale(1.5); /* scale to 1.5* size */
transform-origin: bottom left; /*bottom, center or percentage*/

transform: rotate(90deg);
transform-origin: bottom left;
```

- The full list of functions

You can *transition* a *transform*

```
img {
  transition: transform 1s ease-in;
}

img:hover {
  transform: rotate(360deg); /* spin it! */
}
```

See examples/animations-transforms.html

Animating in the web

- CSS Animations are the third way to animate without JavaScript

  Performs well but still experimental

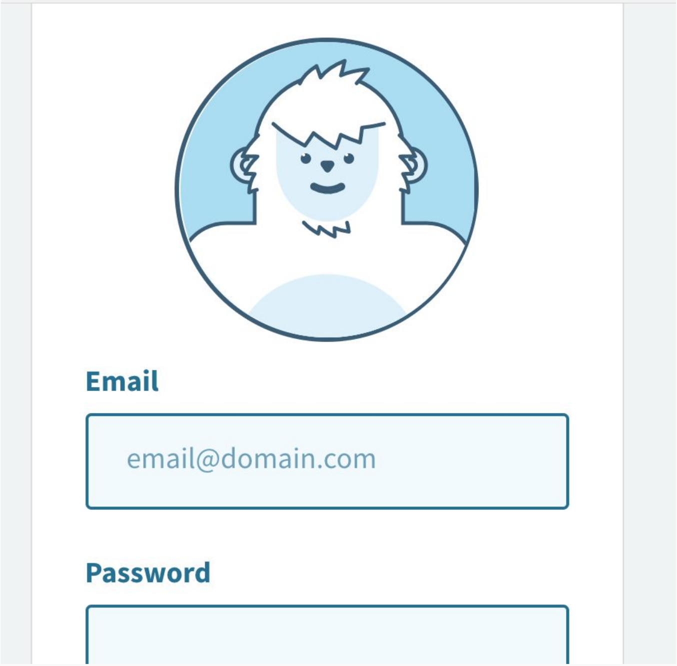
- Set animation properties on the element, defining **easing**, **timing**, **iterations**, etc...
- Then define **@keyframes**, specifying the values the properties will *step* through

```
p {
  animation-duration: 3s;
  animation-name: slidein;
}

@keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%;
  }
  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

See examples/animations-with-keyframes.html

Animations and SVG

- Putting it all together in context https://codepen.io/dsenneff/pen/QajVxO Or **examples/animations-with-svg.html**



Credit for this animation goes to @dsenneff

Lab 6 - Animating

- Let’s set up some basic transitions and animations.
- Starting from **6-animating.html**
- **Your mission:** Create a simple **transition** so that when the user hovers over the <nav> buttons, the background color changes. Experiment with also adding a **transform** to the animation (scale? rotate?).
- **Bonus**: Set up **animation** @keyframes to use on the div.items. See if you can get them to “slide in” from either the right or left side when the page loads -- but just the one time (we don’t want the slidein animation to repeat).

What we just learnt

What was once only possible with JavaScript is now achievable with pure CSS animations.

Transitions define simple, linear property changes.

Transforms affect the element, with or without animations.

Animate & @Keyframes give more fine grained control over the animation flow.

Outline

Responsive web design

Media Queries & the Viewport

Grids & FlexBox

Icons & Web Fonts

Transitions & Animations

Going beyond

Going beyond

Treat devices as first-class users

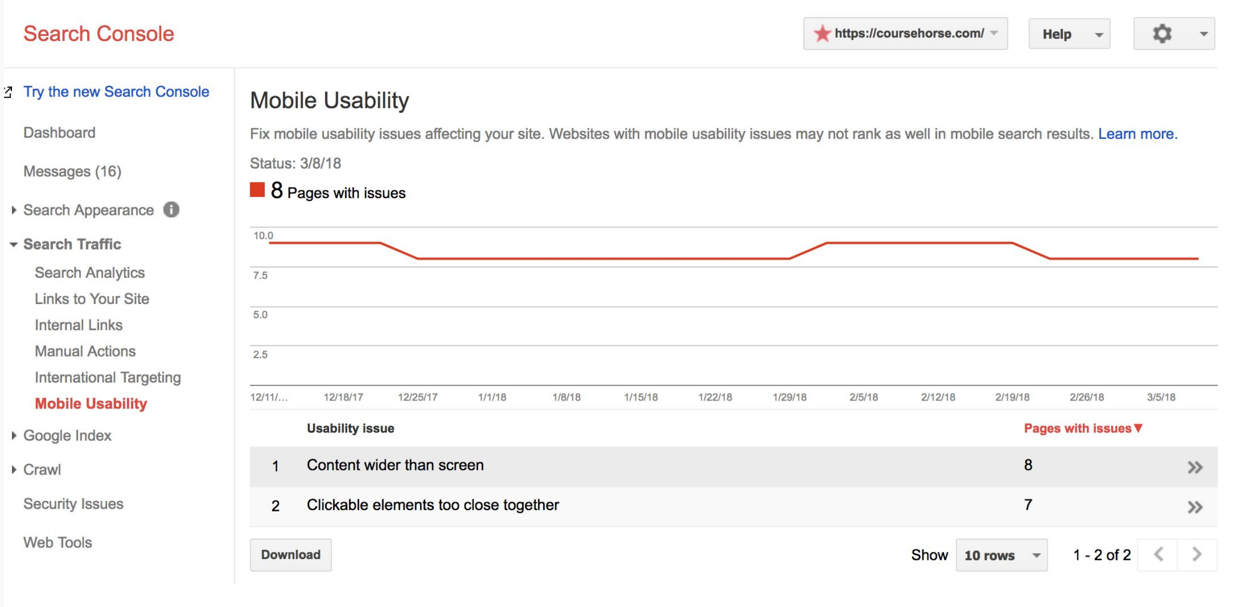
- **Speed matters;** connection speed matters
- **Not just a mouse**, lots of interaction
- **Consider the finger**; make tap areas large enough
- **Consider the location**; Geolocation
- **Device capabilities** beyond the web
- Be **frictionless** between web and mobile
- **Don’t overuse icons** (text + icon wins out, always)
- **Be responsive with feedback** (a tap should immediately feel like a tap)
- **Let the user control** their layout preferences (zoom, font-size, sr)

Use a framework?

- There are a lot of front-end frameworks for styles out of the box
  - FoundationUI
  - Bootstrap
  - SemanticUI
- Determine if you need a grid
  - Don’t follow grids rigidly, they are there to help you lay out your page.

Testing your designs

- https://testmysite.thinkwithgoogle.com/
- https://search.google.com/test/mobile-friendly
- Browserstack.com
- Google Search Console



Mobile Usability

Fix mobile usability issues affecting your site. Websites with mobile usability issues may not rank as well in mobile search results. Learn more.

Status: 3/8/18

🟥 8 Pages with issues

	Usability issue	Pages with issues▼
1	Content wider than screen	8
2	Clickable elements too close together	7

Going beyond

Responsive Media (Retina devices, scaling)

More native APIS (See: https://whatwebcando.today)

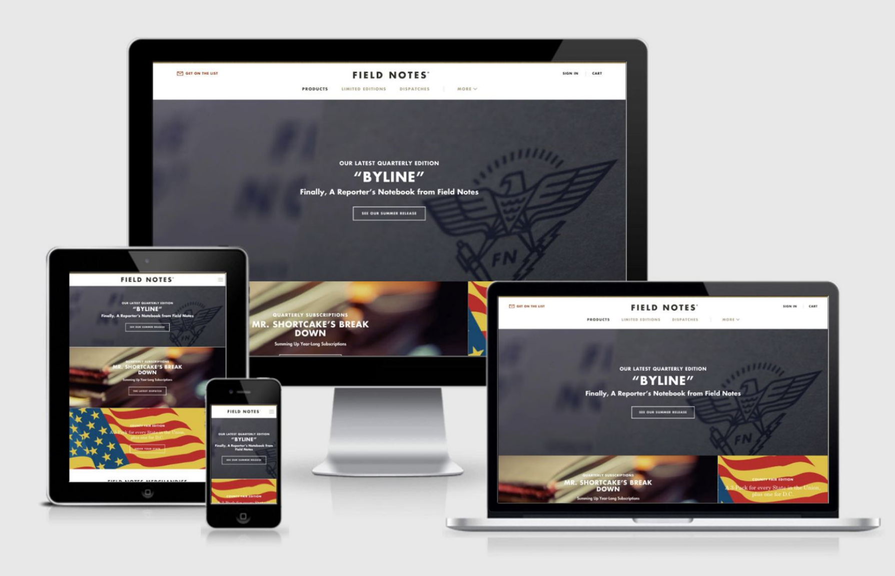
Web sites as apps (PWAs)

Accelerated Mobile Pages (AMP)

Thanks for your attention

I need your feedback before you leave:

<TBD>


FIELD NOTES
OUR LATEST QUARTERLY EDITION
"BYLINE"
Finally, A Reporter's Notebook from Field Notes
QUARTERLY SUBSCRIPTIONS
MR. SHORTCAKE'S BREAK DOWN
Summing Up Year-Long Subscriptions