

# **IN4MATX 133: User Interface Software**

**Lecture 4:**  
**Responsive Design & Bootstrap**

Professor Daniel A. Epstein  
TA Jamshir Goorabian  
TA Simion Padurean

# Today's goals

By the end of today, you should be able to...

- Describe how responsive and adaptive design differ and when you might prefer one or the other
- Explain the advantages and disadvantages of a mobile-first design
- Utilize media queries to create responsive layouts
- Develop grid-based layouts using Bootstrap

# Recall the three waves of computing...

**The Computer  
for the 21st Century**

*Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence*

by Mark Weiser

**T**he most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

Consider writing, perhaps the first information technology. The ability to represent spoken language by symbols is far less abstract than memory from the limits of individual memory. Today this technology is ubiquitous in industrialized countries. Not only do books, magazines and newspapers convey written information, but so do street signs, billboards, shop signs and even graffiti. Candy wrappers are covered in writing. The constants are growing. The products of "literacy technology" does not require active attention, but the information to be transmitted is ready for use at a glance. It is difficult to imagine modern life without it.

Silicon-based information technology, in contrast, is far from having become part of the environment. More than 50 million personal computers have been sold, and the computer now exists primarily in a world of its own. It

MARK WEISER is head of the Computer Science Laboratory at the Xerox Palo Alto Research Center. He worked at PARC before working at PARC, he was a professor of computer science at the University of Michigan from 1979 to 1985, and at the University of Michigan in 1979. Weiser also helped found an electronic publishing company and a video arts company and claims to enjoy "writing programs for the fun of it." His most recent technical work involved the implementation of the first three-level computer memory reclamation, known in the field as garbage collection.

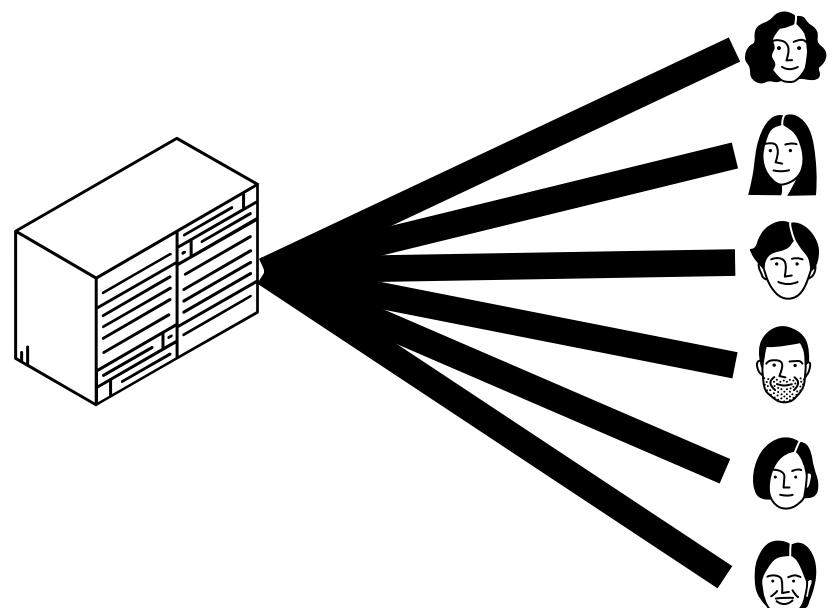
94 SCIENTIFIC AMERICAN September 1991



# Three waves of computing



Mainframe  
computing



“Many to one”



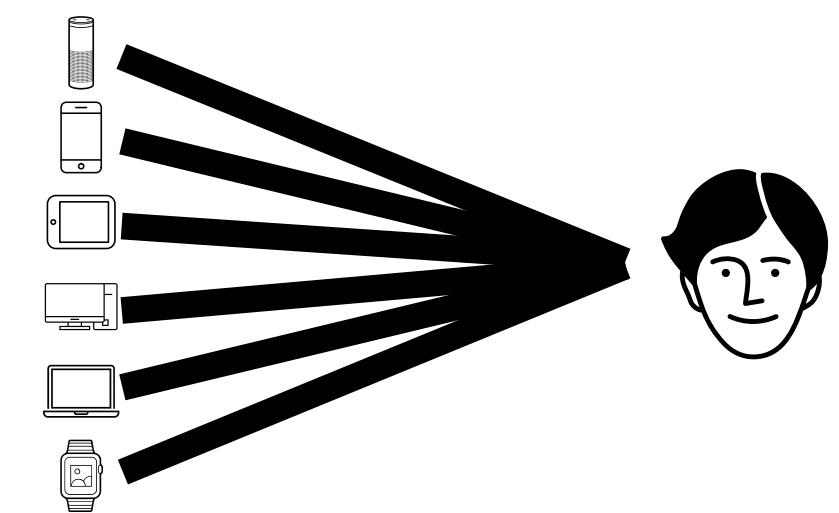
Personal  
computing



“One to one”



Ubiquitous  
computing



“One to many”

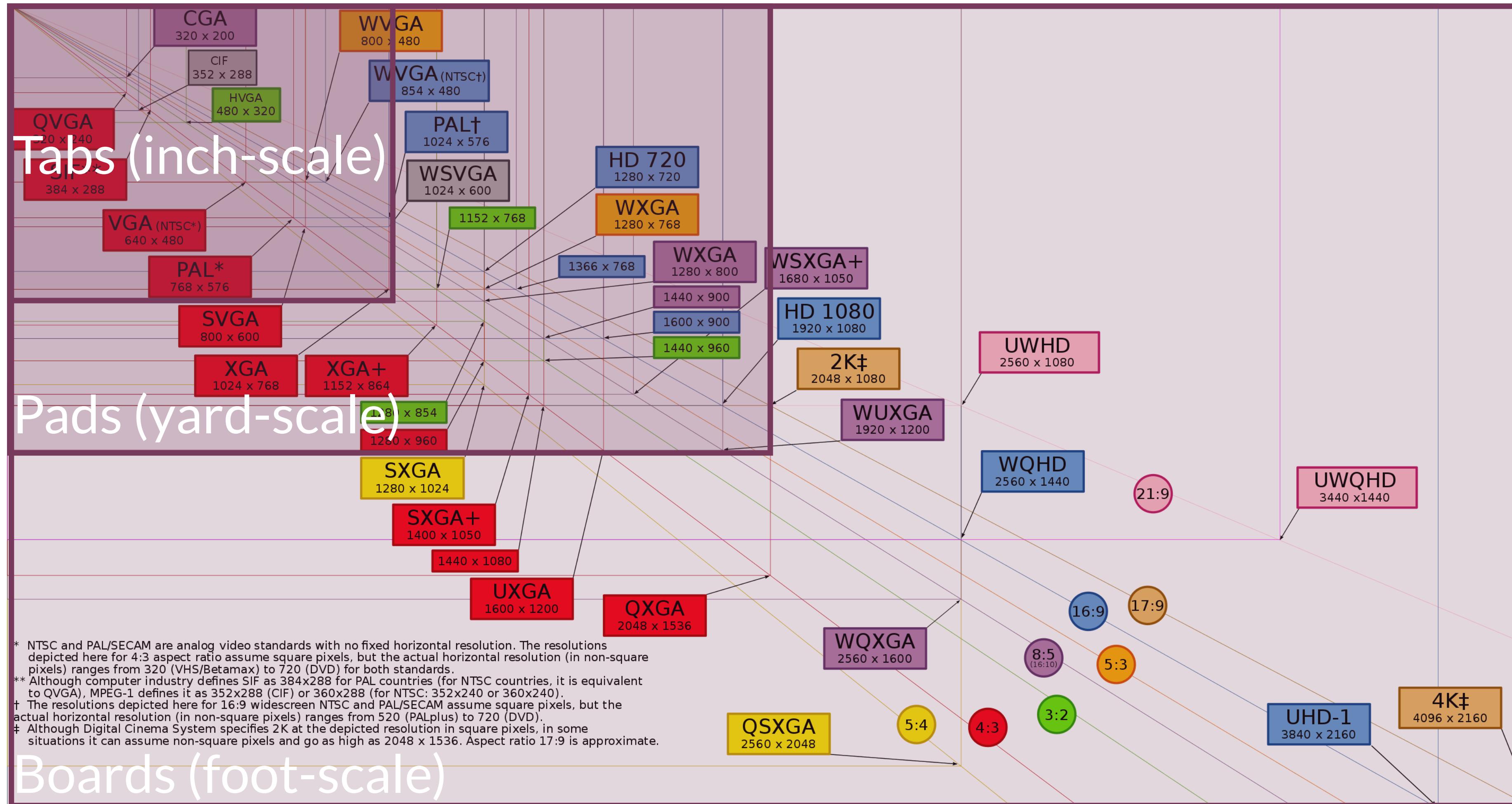
# Websites in the personal computing era

- 960 px wide was pretty common
  - Most screens were 1024x978, leave some room for vertical scrollbar
  - Nicely divisible, can create even columns



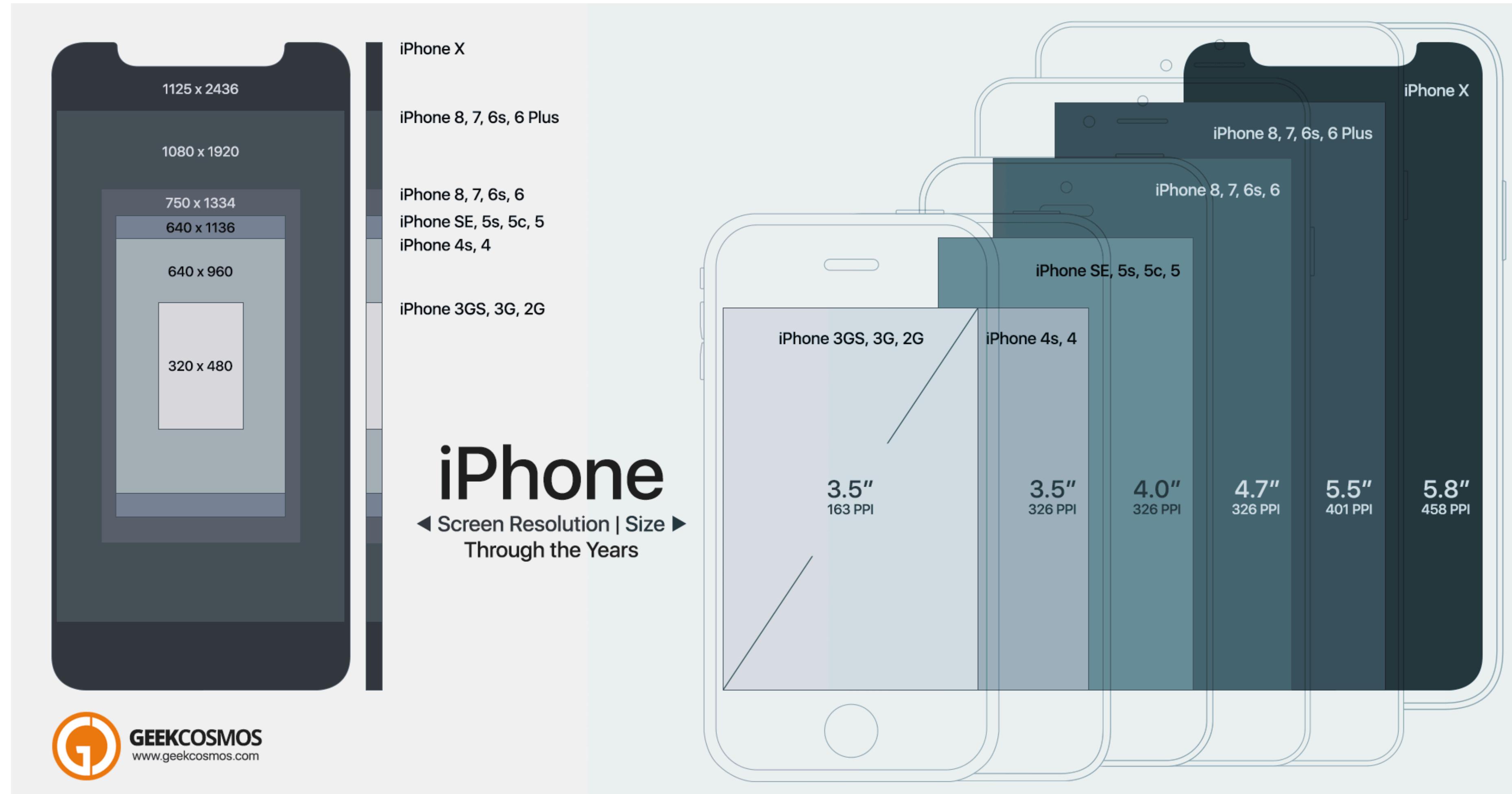
<https://960.gs/>

# Websites today: ubiquitous computing



[https://en.wikipedia.org/wiki/Display\\_resolution](https://en.wikipedia.org/wiki/Display_resolution)

# Websites today: just the iPhone!



**So... how do we account for this?**

**Responsive design or Adaptive design**

# **Responsive design**

- Develop one set of HTML and CSS which changes layout depending on screen sizes

# **Adaptive design**

- Develop and maintain multiple sets of code, change layout depending on device type and screen size

# Question



## Responsive or Adaptive?

- A Top is responsive, bottom is adaptive
- B Top is adaptive, bottom is responsive
- C Both are responsive
- D Both are adaptive
- E These are neither responsive nor adaptive



# Responsive design

- + Easier to maintain one code base, future-proof
- Worse performance; requires downloading entire stylesheet
- Emphasis on making it “look right” rather than creating an experience

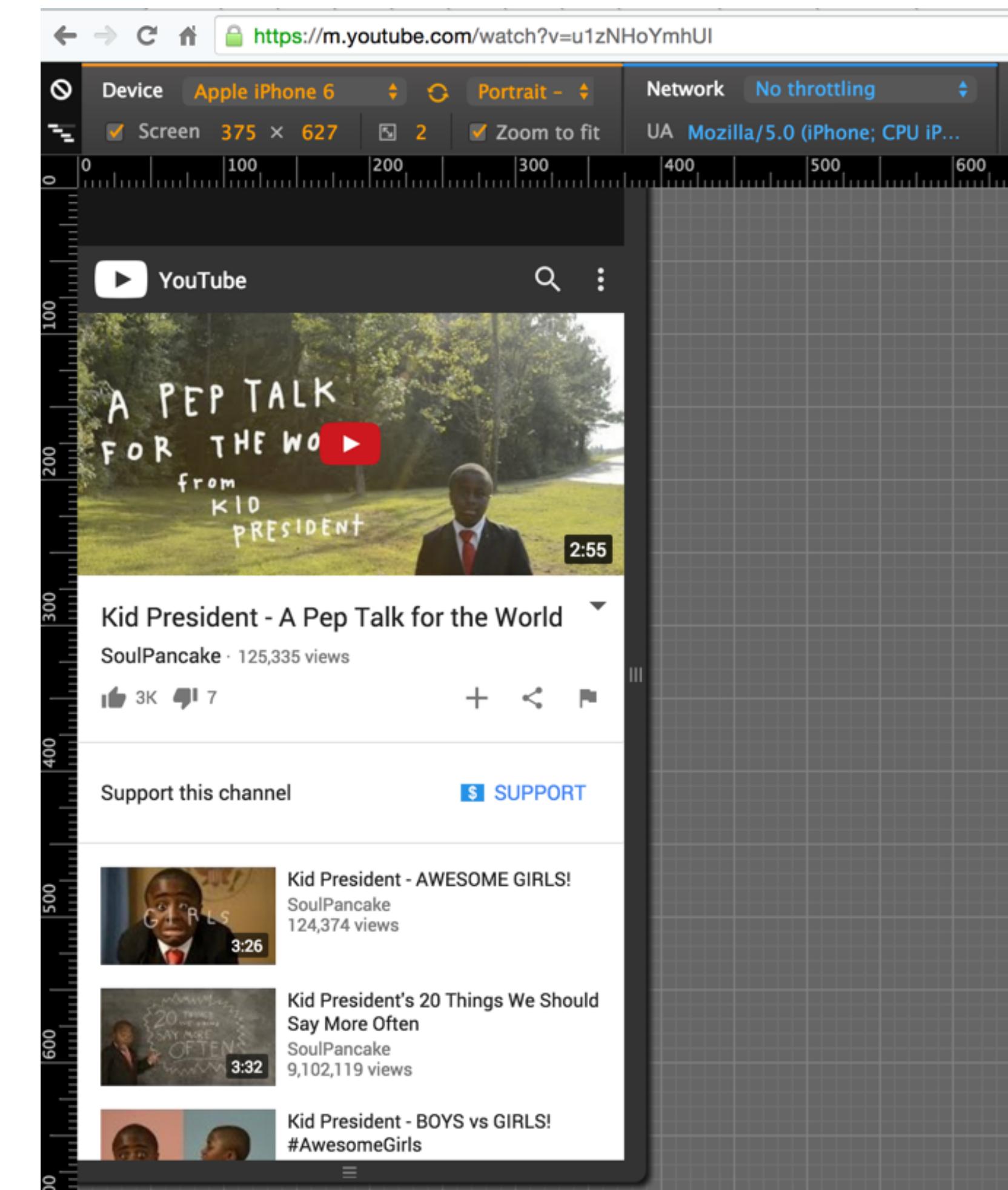
# Adaptive design

- + Can cater experience to a device’s capabilities and performance
- Much more difficult to maintain separate codebases
- Limits development to a few key capabilities because you have to implement for everything

**Most pages are responsive,  
but sometimes it's crucial  
to create the best experience**

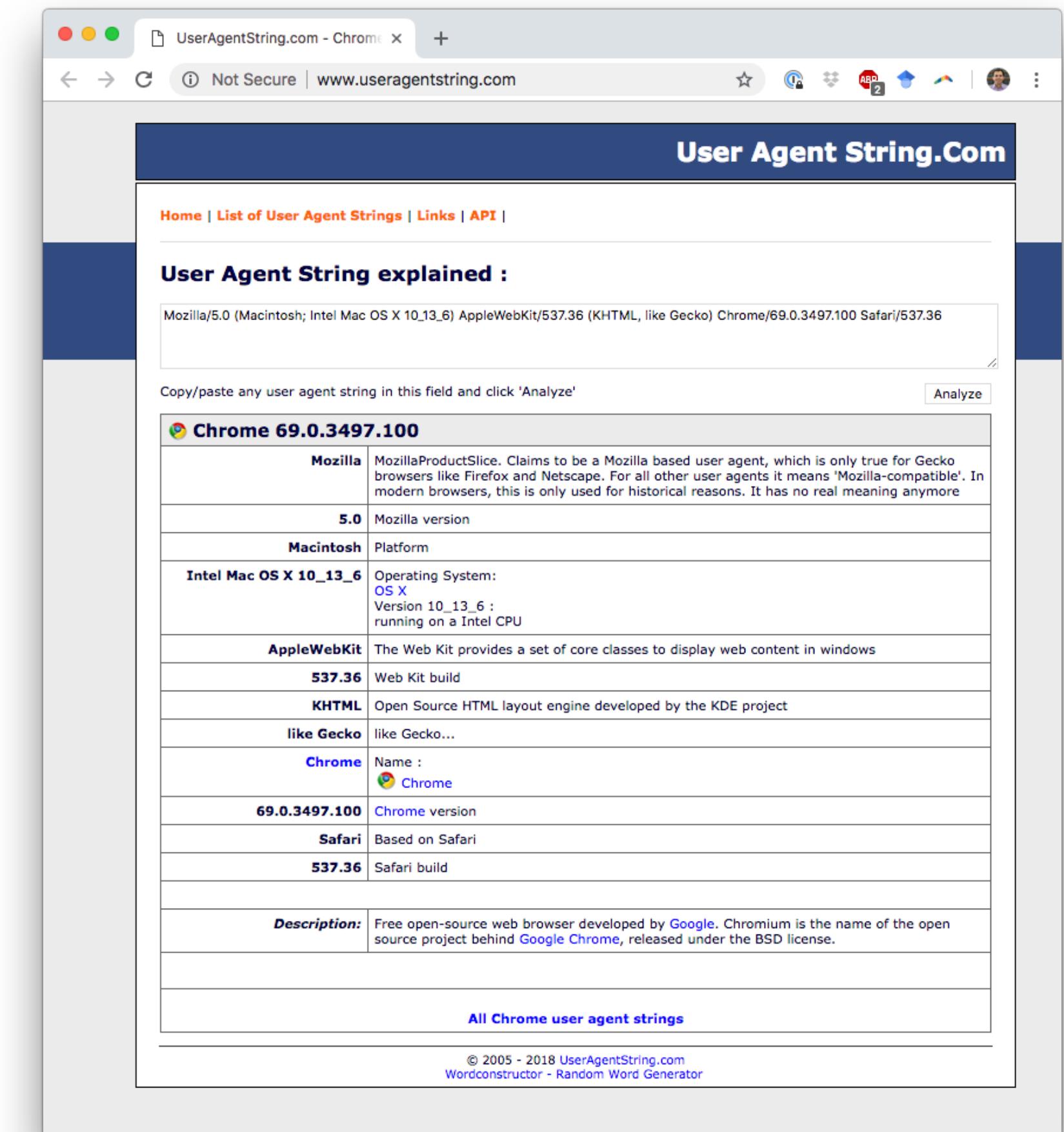
# Adaptive design

- Video = a lot to load
  - Why send a higher resolution than the screen can render?
  - Why use up your own bandwidth?
  - Laggy videos mean unhappy users
- Google can afford the development burden



# Adaptive design

- User agent string accessible via JavaScript
  - `navigator.userAgent`
- There's usually a better way
  - Do you care about the browser or operating system?  
Or is resolution sufficient?
  - Can be spoofed or incorrect



[https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser\\_detection\\_using\\_the\\_user\\_agent](https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent)

# Adaptive design

- Media queries in CSS

```
/* CSS */  
@media screen and (device-width: 375px) and (device-height: 667px)  
and (-webkit-device-pixel-ratio: 2) {  
    /* iPhone 8-specific CSS */  
}
```

- Load appropriate external stylesheet

```
<!--HTML-->  
<head>  
    <link rel="stylesheet" media="screen and (device-width: 375px)  
        and (device-height: 667px) and (-webkit-device-pixel-ratio: 2)" href="iPhone8.css">  
</head>
```

# Media query syntax

- @media
- screen, print, speech, all
- min-width, max-width
- orientation, -webkit-min-device-pixel-ratio
- Many, many more

[https://www.w3schools.com/cssref/css3\\_pr\\_mediaquery.asp](https://www.w3schools.com/cssref/css3_pr_mediaquery.asp)

# **Transitioning to responsive design**

# Breakpoints

- The point at which your design “breaks” and is no longer visually appealing or usable
- Designs vary, but most have 3-5 breakpoints
  - extra small (old mobile), small (mobile), medium (tablet), large (laptop or desktop), extra large (wide desktop or wall display)
  - Again, somewhat similar to Weiser’s three types of computers

# Breakpoints

```
@media screen and (max-width: 640px) {  
    /* small screens */  
}
```

```
@media screen and (min-width: 640px and max-width:  
1024px) {  
    /* medium screens */  
}
```

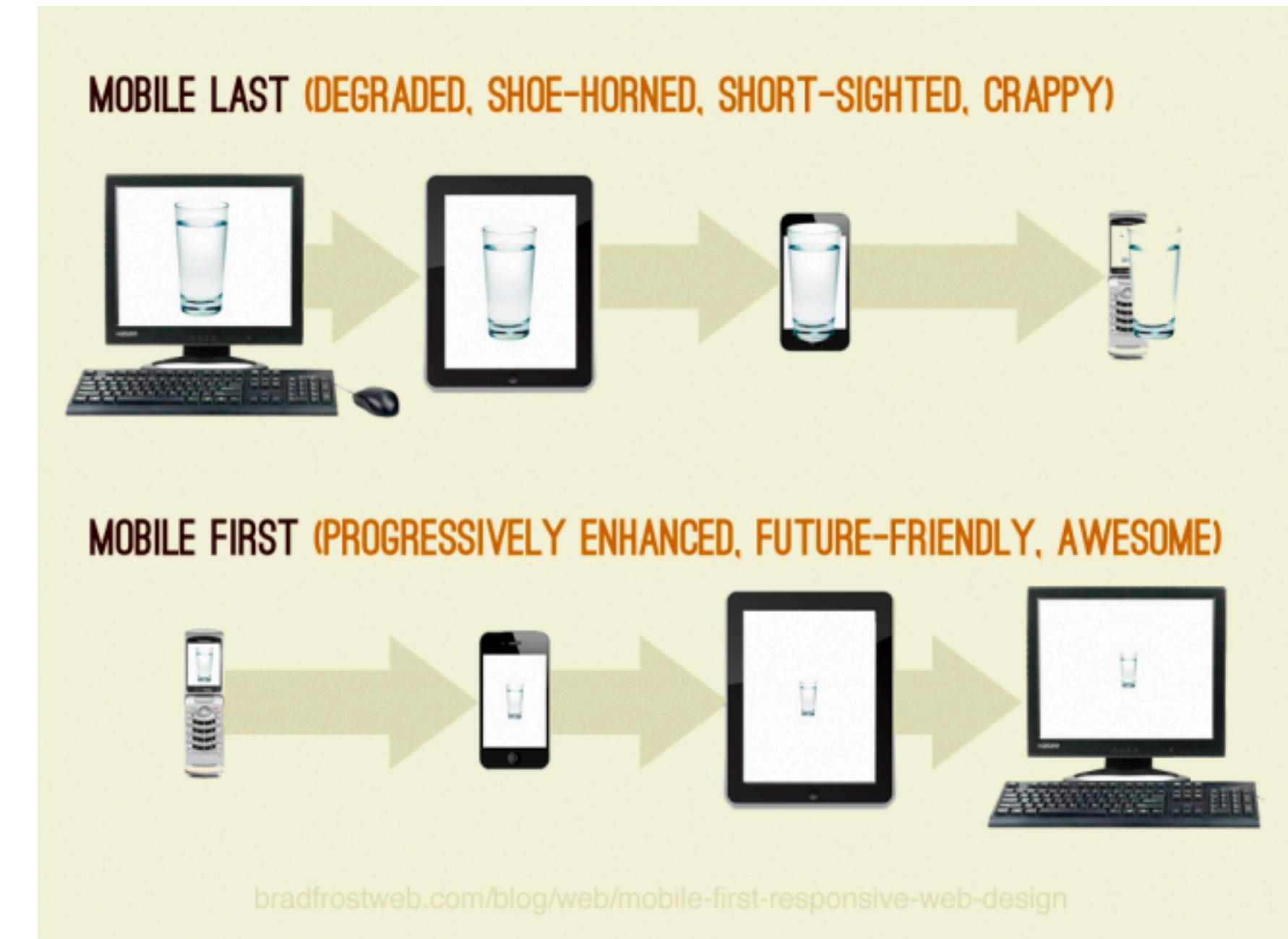
```
@media screen and (min-width: 1024px) {  
    /* large screens */  
}
```

# Responsive design

- Fluid grids
  - Lay out content in columns whose widths can vary
  - Bootstrap helps with this; more on that in a bit
- Flexible images
  - Let image size change based on screen layout
  - Put images in containers which will scale appropriately
  - Set `width: 100%, max-width: 100%, height: auto`

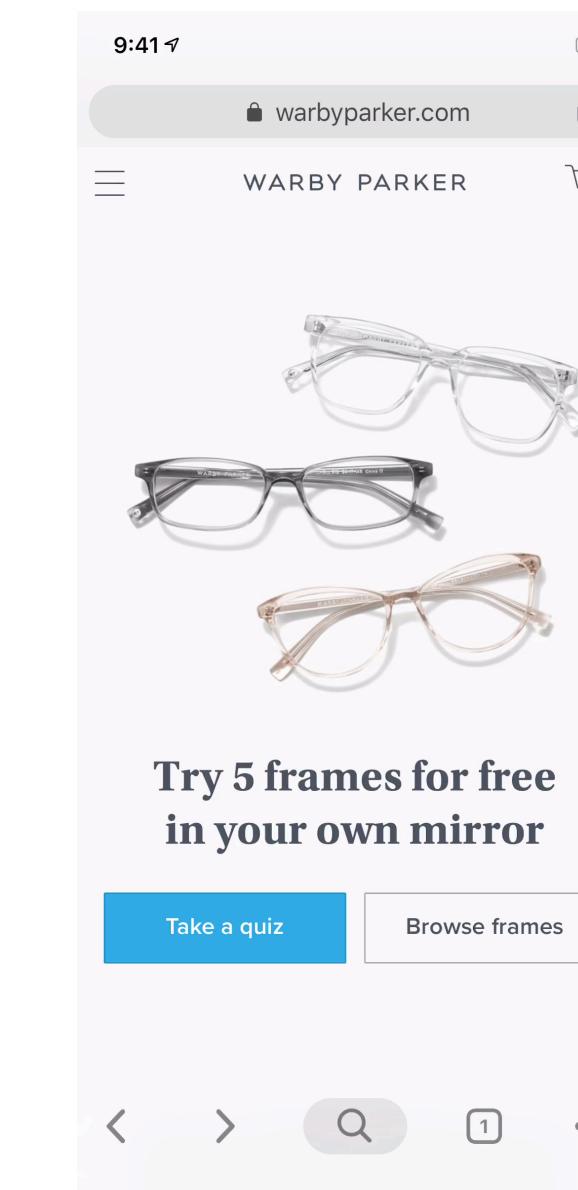
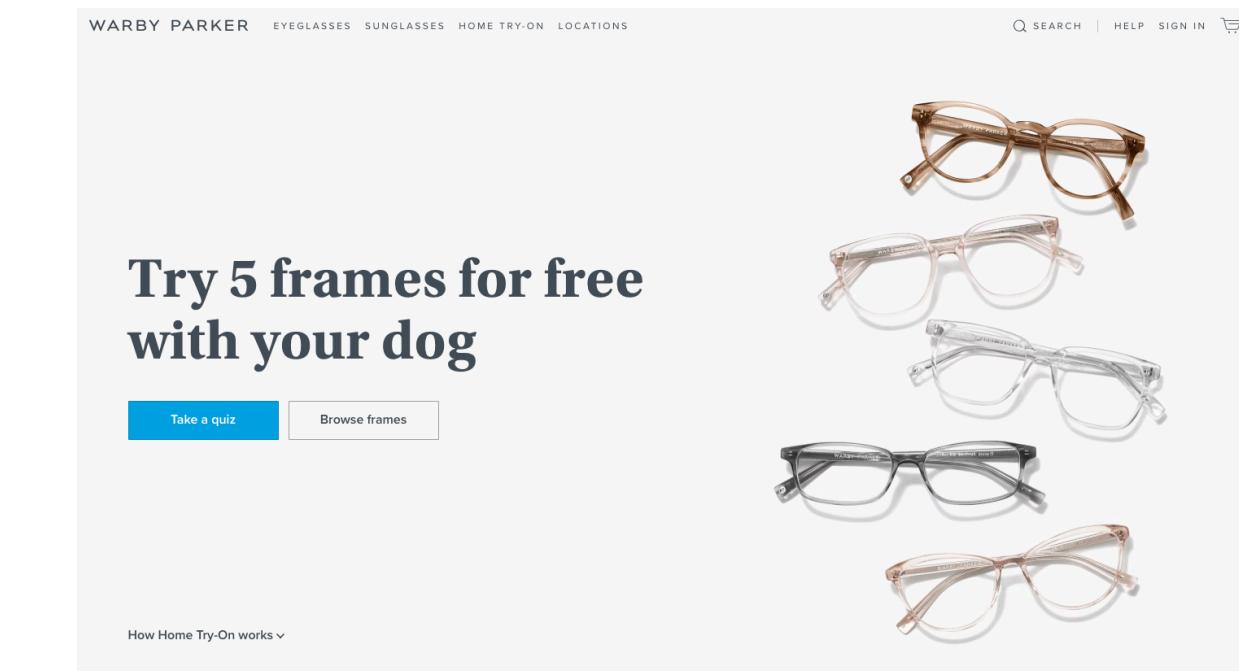
# Mobile-first design

- “Graceful degradation” vs. “progressive enhancement”
- Plan your design for mobile
- Then make your app *better* with more real estate
  - Add more features
  - Make existing features easier to navigate



# A few tips for mobile design

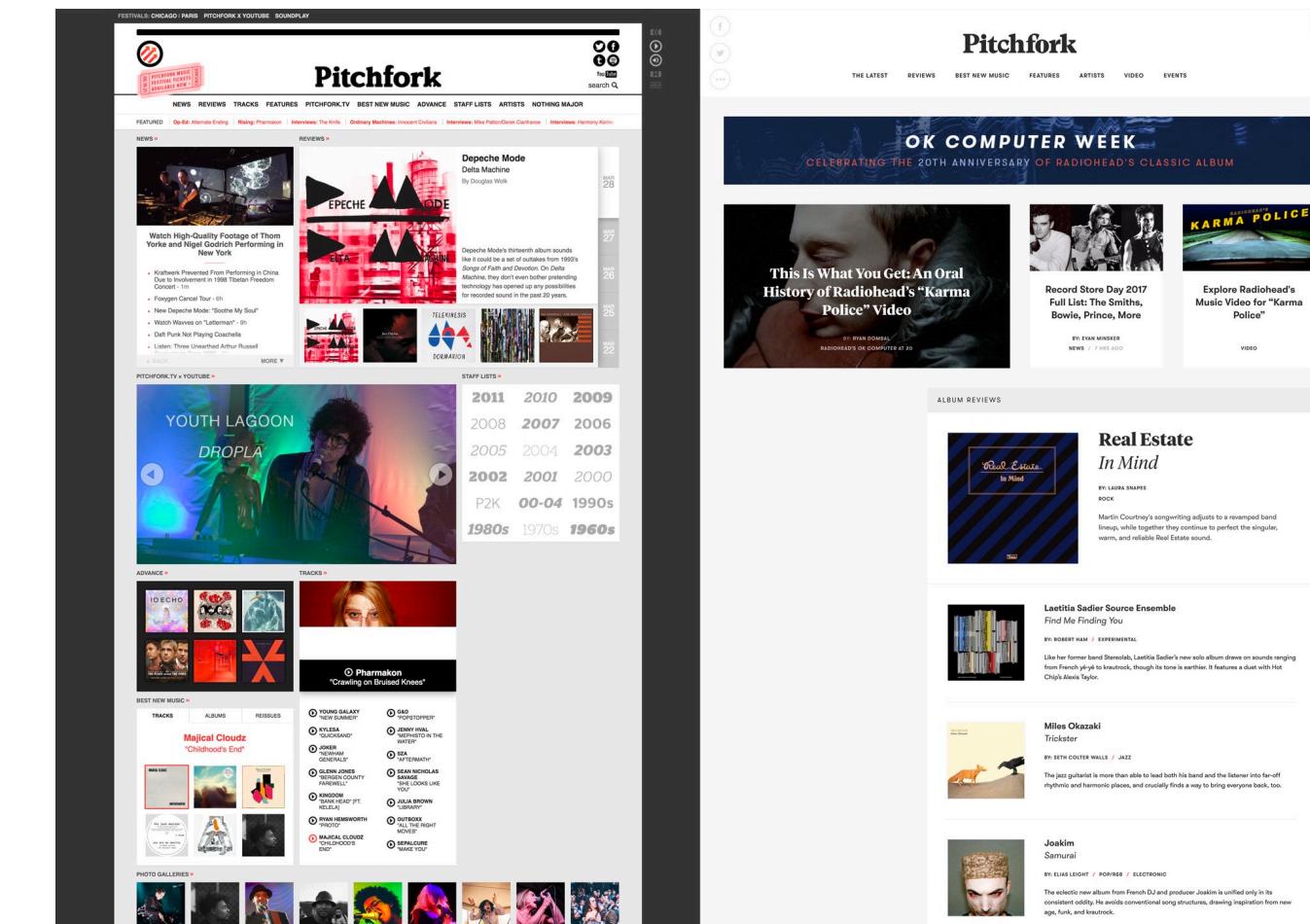
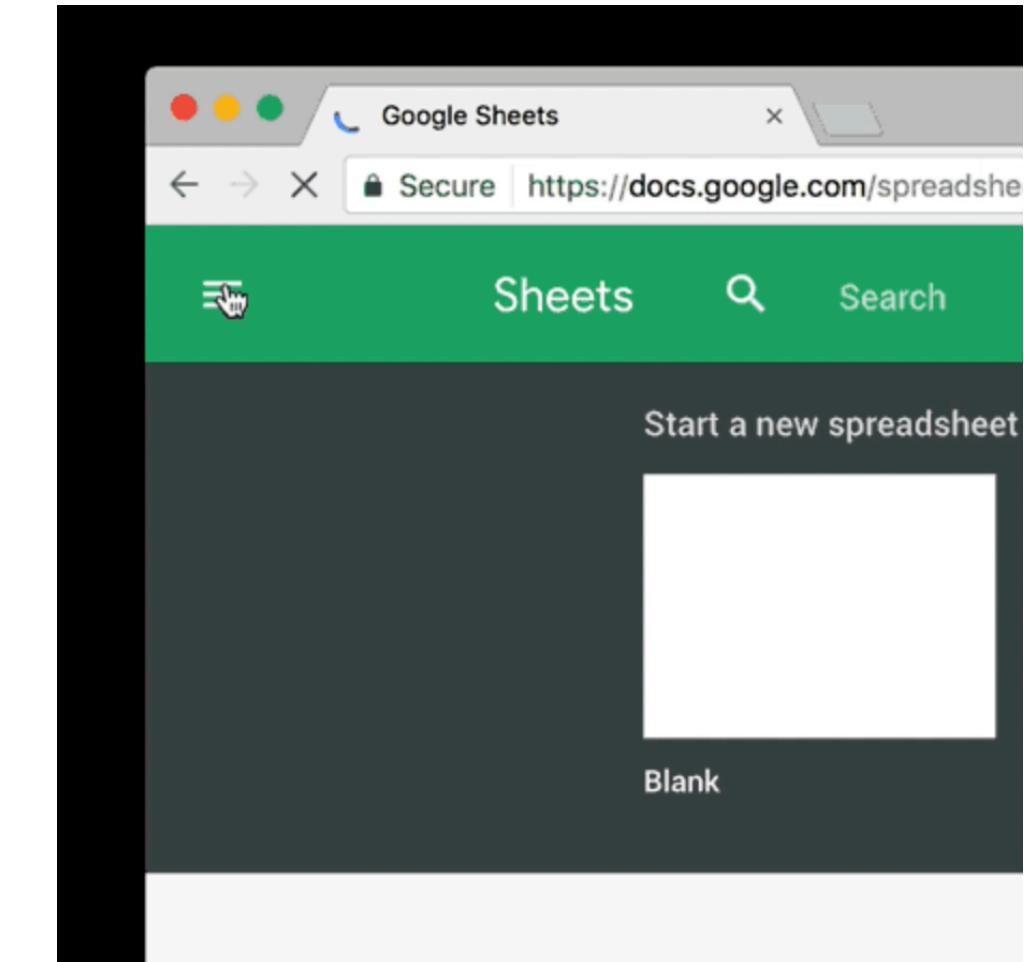
- Show the same content, organize it appropriately
- Stack content vertically
- Show navigation on demand
- Larger touch targets



<https://www.bluefountainmedia.com/blog/desktop-vs-mobile-three-key-website-design-differences>

# Mobile-first, not mobile-only

- Copying mobile UI to desktop creates inefficiencies
  - Extra clicks to navigate
  - Underutilized real estate



<https://blog.prototyp.io/mobile-first-desktop-worst-f900909ae9e2>

# Mobile-first, not mobile-only

- Plan your design for mobile
- But consider how the experience should change on desktop, etc.
- Go beyond making everything bigger
  - *Enhance* your design

# **Grid-based layouts**

# Grid-based layouts

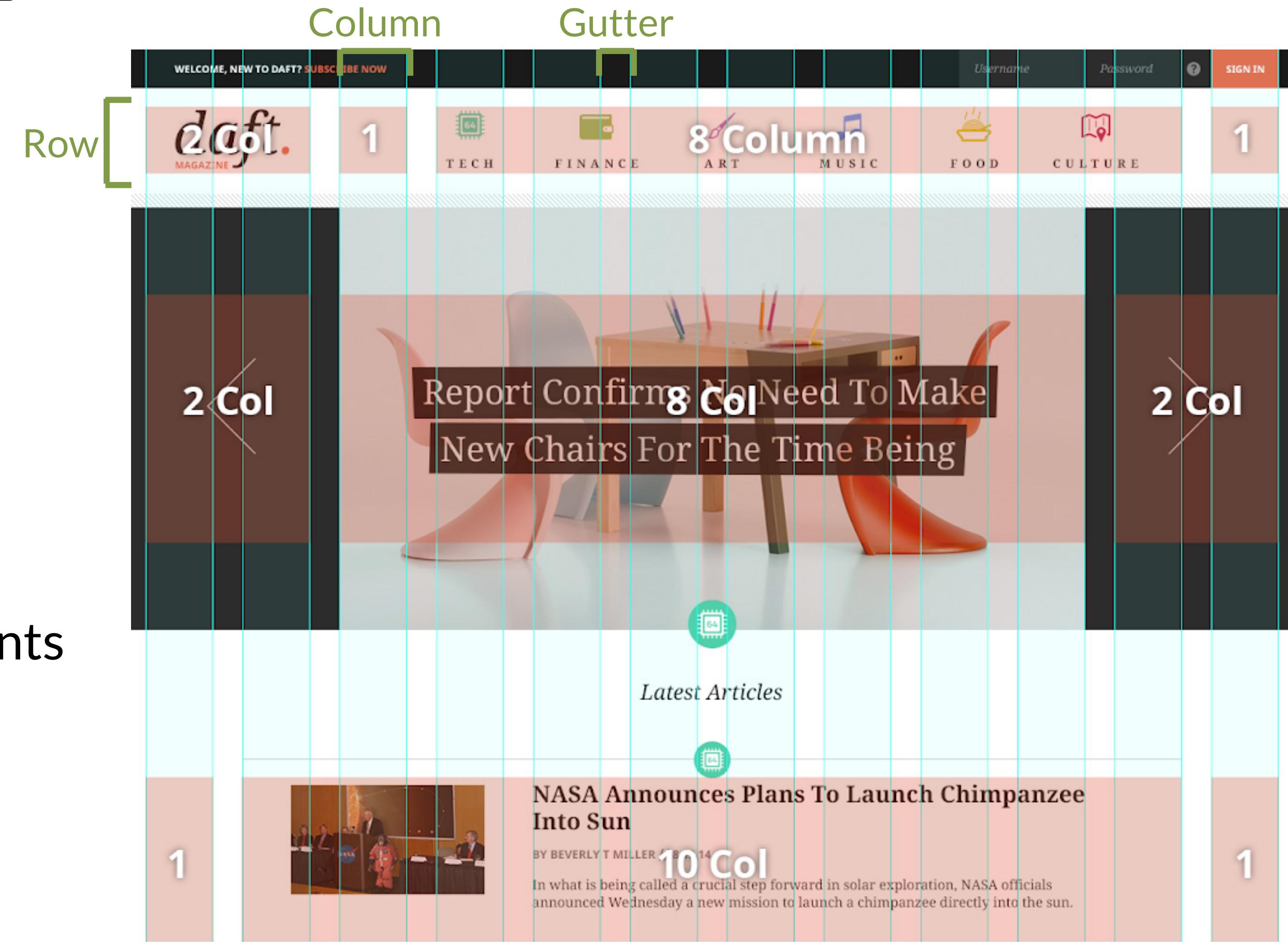
- Established tool for content arrangement
- Gridded content is familiar and easy to follow
- In general, it's good to target fewer lines
- But breaking that rule is important for creativity and attention-grabbing



<http://printingcode.runemadsen.com/lecture-grid/>

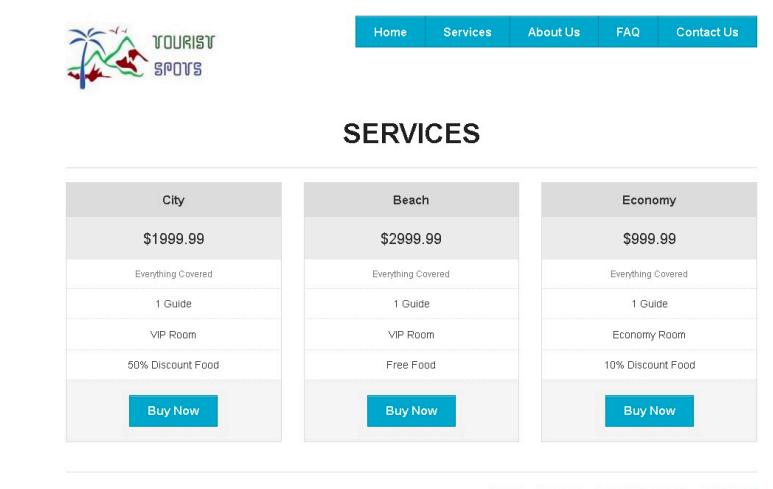
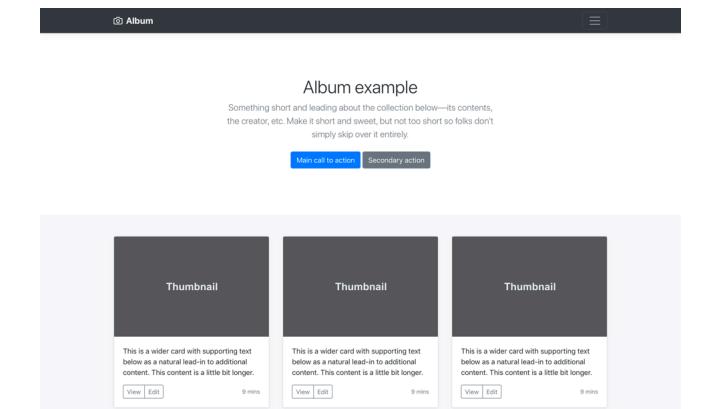
# Grid-based layouts

- Rows
- Columns
- Gutters
- Padding/spacing
  - Defined by specific elements



# Grid-based frameworks

- Bootstrap (<https://getbootstrap.com/>)
  - Most popular, most extensions
- Foundation (<https://foundation.zurb.com/>)
  - Includes icons, drag&drop editor
- Pure.css (<https://purecss.io/>)
  - Small file size, 3.8KB



# Digging into Bootstrap



**Bootstrap**

# Bootstrap

- Direct download
  - <http://getbootstrap.com/docs/4.1/getting-started/download/>
- CSS and JavaScript files
- Minified files are compressed, will load faster
- .map files support editing preprocessed files
  - We won't really touch on those in this class
- We'll use bootstrap.min.css for now

Name	Date Modified	Size	Kind
css	Jul 23, 2018 at 5:49 PM	--	Folder
bootstrap-grid.css	Jul 23, 2018 at 6:37 PM	38 KB	CSS
bootstrap-grid.css.map	Jul 23, 2018 at 6:37 PM	99 KB	Document
bootstrap-grid.min.css	Jul 23, 2018 at 6:37 PM	29 KB	CSS
bootstrap-grid.min.css.map	Jul 23, 2018 at 6:37 PM	68 KB	Document
bootstrap-reboot.css	Jul 23, 2018 at 6:37 PM	5 KB	CSS
bootstrap-reboot.css.map	Jul 23, 2018 at 6:37 PM	61 KB	Document
bootstrap-reboot.min.css	Jul 23, 2018 at 6:37 PM	4 KB	CSS
bootstrap-reboot.min.css.map	Jul 23, 2018 at 6:37 PM	26 KB	Document
bootstrap.css	Jul 23, 2018 at 6:37 PM	174 KB	CSS
bootstrap.css.map	Jul 23, 2018 at 6:37 PM	430 KB	Document
bootstrap.min.css	Jul 23, 2018 at 6:37 PM	141 KB	CSS
bootstrap.min.css.map	Jul 23, 2018 at 6:37 PM	562 KB	Document
js	Jul 23, 2018 at 5:49 PM	--	Folder
bootstrap.bundle.js	Jul 23, 2018 at 6:37 PM	212 KB	JavaScript
bootstrap.bundle.js.map	Jul 23, 2018 at 6:37 PM	359 KB	Document
bootstrap.bundle.min.js	Jul 23, 2018 at 6:37 PM	71 KB	JavaScript
bootstrap.bundle.min.js.map	Jul 23, 2018 at 6:37 PM	294 KB	Document
bootstrap.js	Jul 23, 2018 at 6:37 PM	124 KB	JavaScript
bootstrap.js.map	Jul 23, 2018 at 6:37 PM	212 KB	Document
bootstrap.min.js	Jul 23, 2018 at 6:37 PM	51 KB	JavaScript
bootstrap.min.js.map	Jul 23, 2018 at 6:37 PM	176 KB	Document

# Bootstrap

- Load bootstrap

```
<link rel="stylesheet" href="css/bootstrap.min.css">
```

```
<link rel="stylesheet" href="css/override.css">
```

# Bootstrap

- Content Delivery Networks (CDN)
  - Browser-side caching reduces burdens of loading files
  - Integrity: hashes to ensure the downloaded file matches what's expected
    - Protects against server being compromised
  - Crossorigin: some imports require credentials, anonymous requires none
- ```
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
      integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
      crossorigin="anonymous">
```

# Bootstrap

## Specifying a viewport

- In page's head
- Sets device width and scale level (for zooming)

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

# Bootstrap

## Designating a container

- All bootstrap content lives in a container

```
<div class="container">  
  <!--Bootstrap content-->  
</div>
```

- Just a class; anything can be a container

```
<main class="container">  
  <!--Bootstrap content-->  
</main>
```

# Bootstrap

## Grid System

- Grid system has 12 columns
  - 12 has a lot of factors (1, 2, 3, 4, 6)
- Content over 12 columns will wrap
  - (3+6+4=13, the 4 will wrap)
- 15px gutter for each
- Classes for `row` and `col-[size]-[number]`

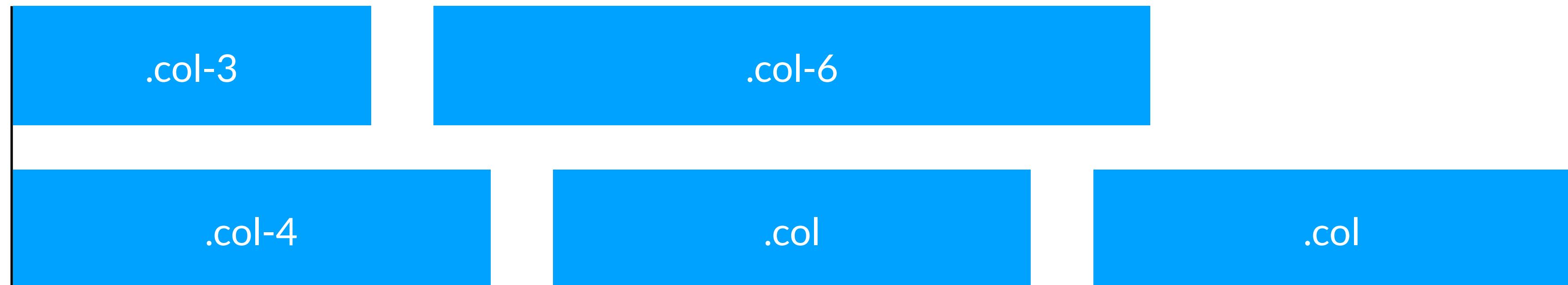
	Extra small devices Phones (<768px)	Small devices Tablets ( $\geq 768px$ )	Medium devices Desktops ( $\geq 992px$ )	Large devices Desktops ( $\geq 1200px$ )
<b>Grid behavior</b>	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
<b>Container width</b>	None (auto)	750px	970px	1170px
<b>Class prefix</b>	<code>.col-xs-</code>	<code>.col-sm-</code>	<code>.col-md-</code>	<code>.col-lg-</code>
<b># of columns</b>	12			
<b>Column width</b>	Auto	~62px	~81px	~97px
<b>Gutter width</b>	30px (15px on each side of a column)			
<b>Nestable</b>	Yes			
<b>Offsets</b>	Yes			
<b>Column ordering</b>	Yes			

# Bootstrap

## Grid System

- Within the same row, content will wrap once it goes over 12 columns
  - Size parameter is optional; will divide space proportionally

```
<main class="container">
  <div class="row">
    <div class="col-3">A</div>
    <div class="col-6">B</div>
    <div class="col-4">C</div>
    <div class="col">D</div>
    <div class="col">E</div>
  </div>
</main>
```

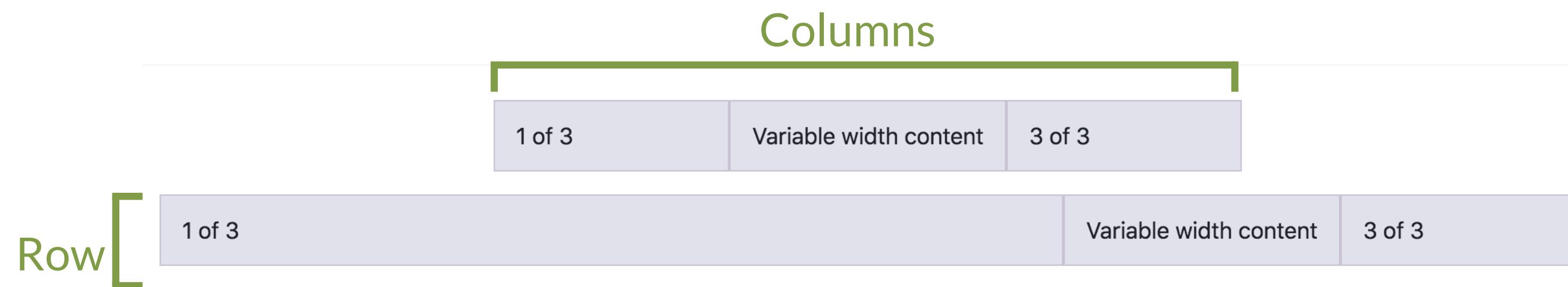


# Bootstrap

## Grid System

- Rows are block elements, while columns are inline

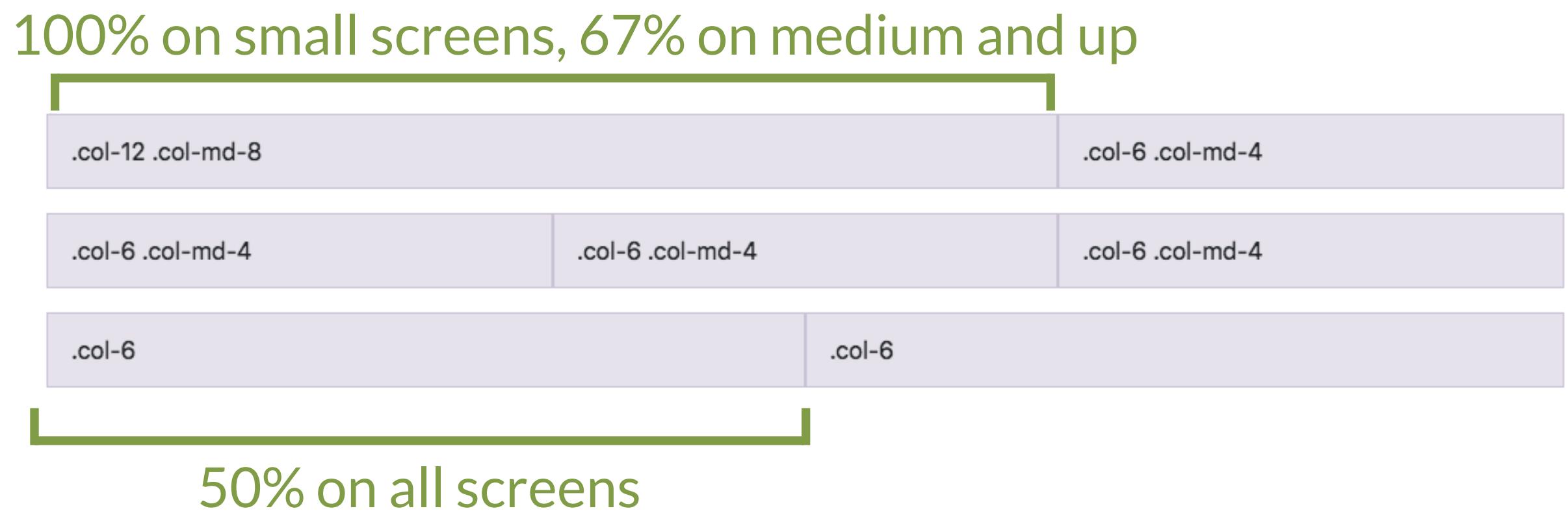
<https://getbootstrap.com/docs/4.1/layout/grid/>



# Bootstrap

## Grid System

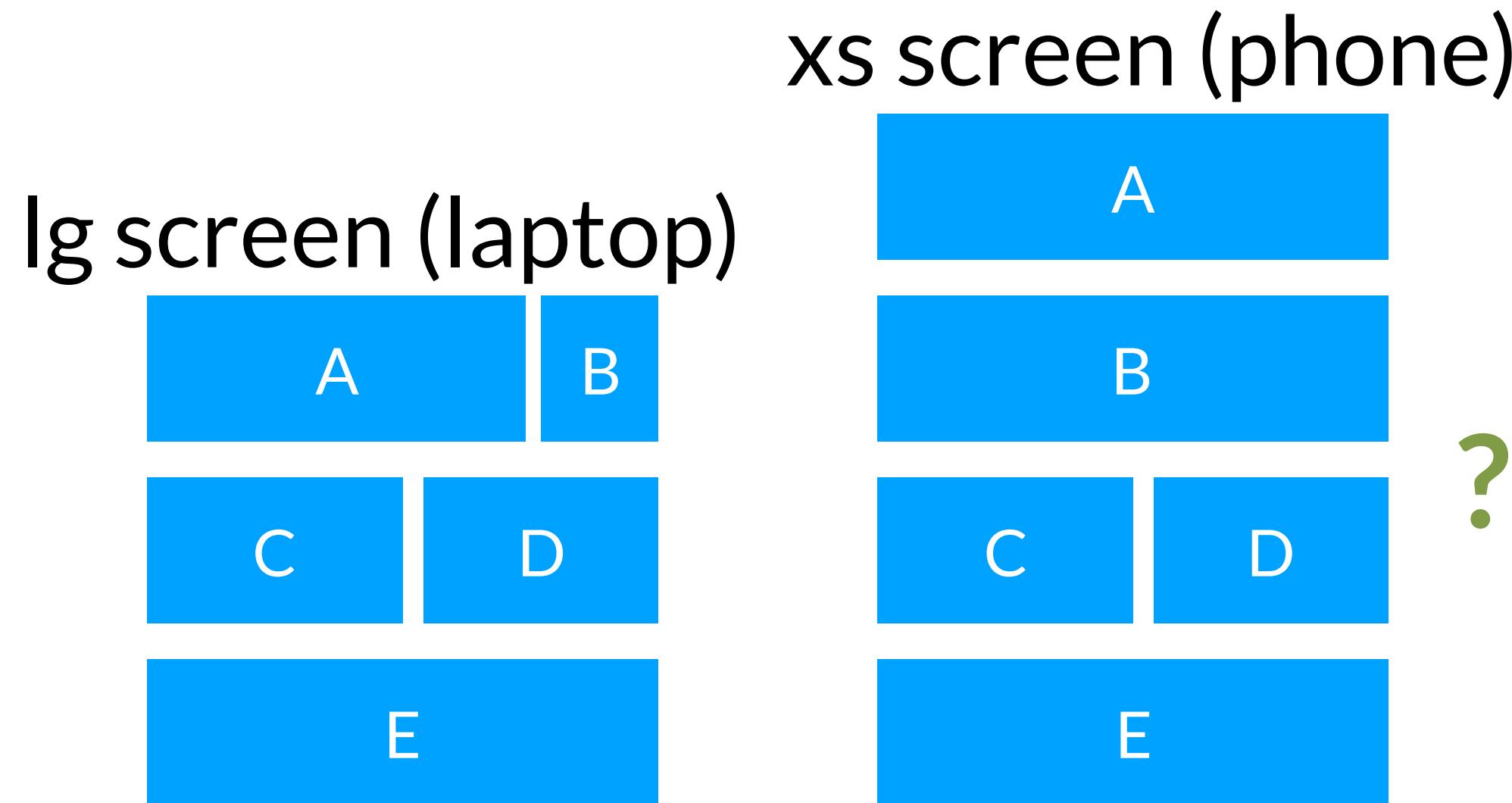
- `.col` with no size defaults to the smallest (`xs`)
- The largest size listed will cover any larger sizes which are not-listed
- Will default to width 12 when no size is specified



# Question



Which code creates



A

```
<div class="row">
  <div class="col-md-8 col-xs-12">A</div>
  <div class="col-md-4 col-xs-12">B</div>
  <div class="col-md-6">C</div>
  <div class="col-md-6">D</div>
  <div class="col-12">E</div>
</div>
```

B

```
<div class="row">
  <div class="col-md-8 col-xs-12">A</div>
  <div class="col-md-4 col-xs-12">B</div>
</div>
<div class="row">
  <div class="col-6">C</div>
  <div class="col-6">D</div>
  <div class="col-12">E</div>
</div>
```

C

```
<div class="row">
  <div class="col-8">A</div>
  <div class="col-4">B</div>
  <div class="col-xs-6">C</div>
  <div class="col-xs-6">D</div>
</div>
<div class="row">
  <div class="col-md-6 col-xl-4">E</div>
</div>
```

D A and B

E B and C

# Breakpoints

```
@media screen and (max-width: 640px) {  
    /* small screens */  
}
```

```
@media screen and (min-width: 640px and max-width:  
1024px) {  
    /* medium screens */  
}
```

```
@media screen and (min-width: 1024px) {  
    /* large screens */  
}
```

# Bootstrap

## Media queries

```
/* Extra small devices (phones, less than 768px) */  
@include media-breakpoint-up(xs) { ... }  
  
/* Small devices (tablets, 768px and up) */  
@include media-breakpoint-up(sm) { ... }  
  
/* Medium devices (desktops, 992px and up) */  
@include media-breakpoint-up(md) { ... }  
  
/* Large devices (large desktops, 1200px and up) */  
@include media-breakpoint-up(lg) { ... }
```

- Variables are Sass mixins, we'll discuss those later in the quarter

# Bootstrap

## Media queries

```
// Example usage:  
@include media-breakpoint-up(sm) {  
  .some-class {  
    display: block;  
  }  
}
```

# Bootstrap

## Hiding and showing

- There are some helpful classes for showing and hiding content across breakpoints

Use a single or combination of the available classes for toggling content across viewport breakpoints.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
.visible-xs-*	Visible	Hidden	Hidden	Hidden
.visible-sm-*	Hidden	Visible	Hidden	Hidden
.visible-md-*	Hidden	Hidden	Visible	Hidden
.visible-lg-*	Hidden	Hidden	Hidden	Visible
.hidden-xs	Hidden	Visible	Visible	Visible
.hidden-sm	Visible	Hidden	Visible	Visible
.hidden-md	Visible	Visible	Hidden	Visible
.hidden-lg	Visible	Visible	Visible	Hidden

<http://getbootstrap.com/css>

# Bootstrap

## Default styling

- Bootstrap will change a lot of styles for you
- There are other custom styles involving various suffixes

**h1. Bootstrap heading**

Semibold 36px

**h2. Bootstrap heading**

Semibold 30px

**h3. Bootstrap heading**

Semibold 24px

Email address

Email

Password

Password

EXAMPLE

Default Primary Success Info Warning Danger Link

<!-- Standard button -->  
<button type="button" class="btn btn-default">Default</button>

Copy

<!-- Provides extra visual weight and identifies the primary action in a set of buttons -->  
<button type="button" class="btn btn-primary">Primary</button>

<!-- Indicates a successful or positive action -->  
<button type="button" class="btn btn-success">Success</button>

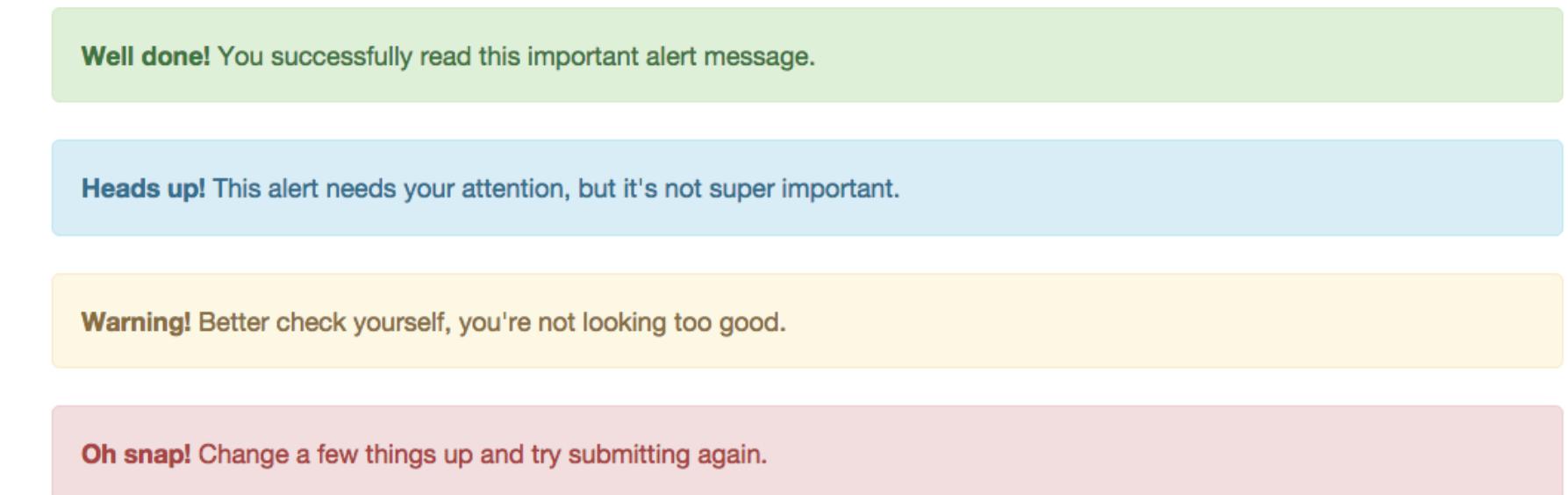
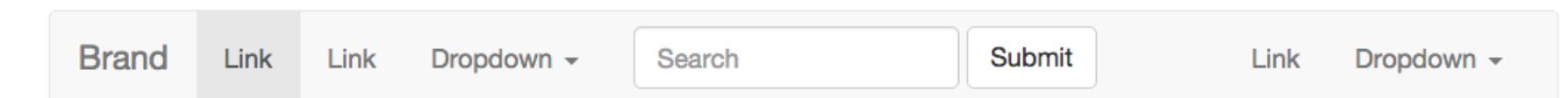
<http://getbootstrap.com/css>

# Bootstrap

## Components

- Components are elements pre-arranged into common patterns
- Makes making navigation bars, dropdowns, alerts, etc. simpler
- Some require JavaScript

<http://getbootstrap.com/css>



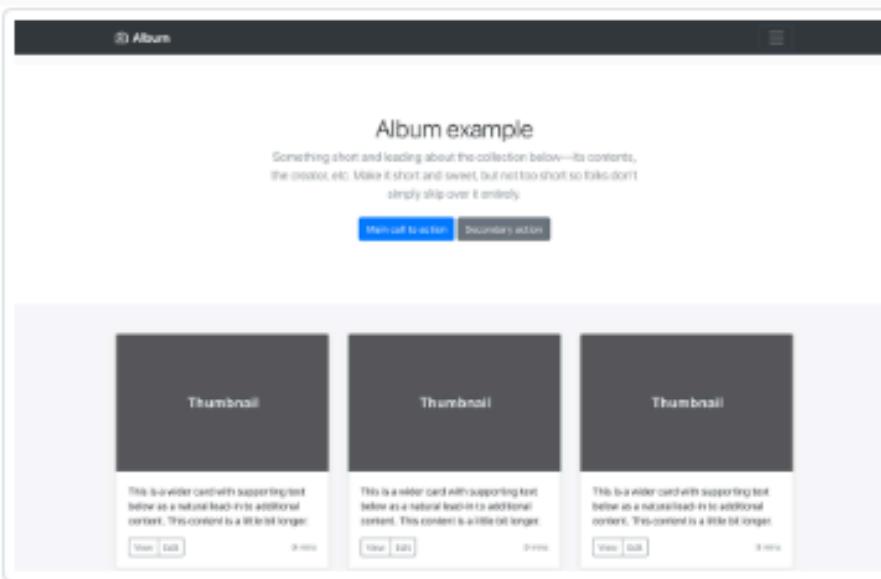
Panel heading
Some default panel content here. Nulla vitae elit libero, a pharetra augue. Aenean lacinia bibendum nulla sed consectetur. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nullam id dolor id nibh ultricies vehicula ut id elit.
Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

**Grid frameworks  
make development easier.**

**What are the downsides?**

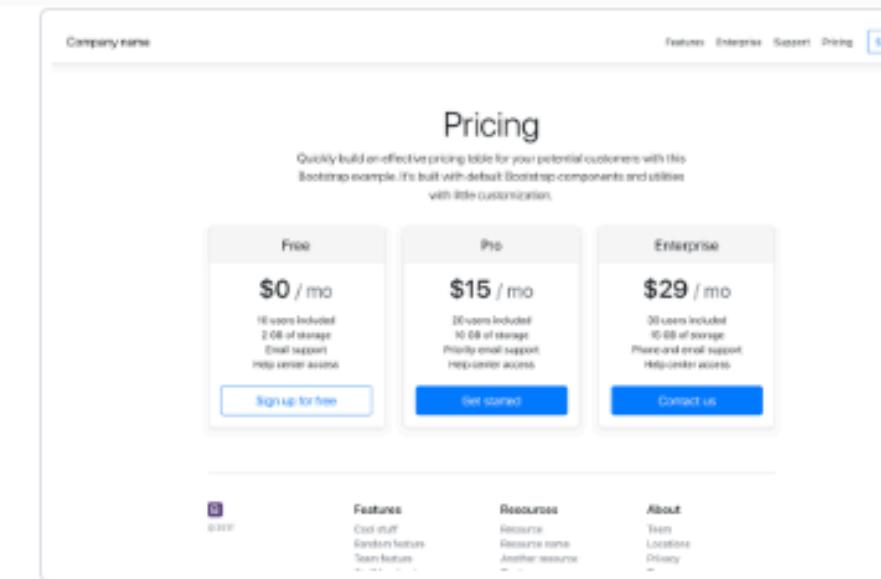
# Opposition to Grid-based frameworks

Can lead to similar-looking webpages



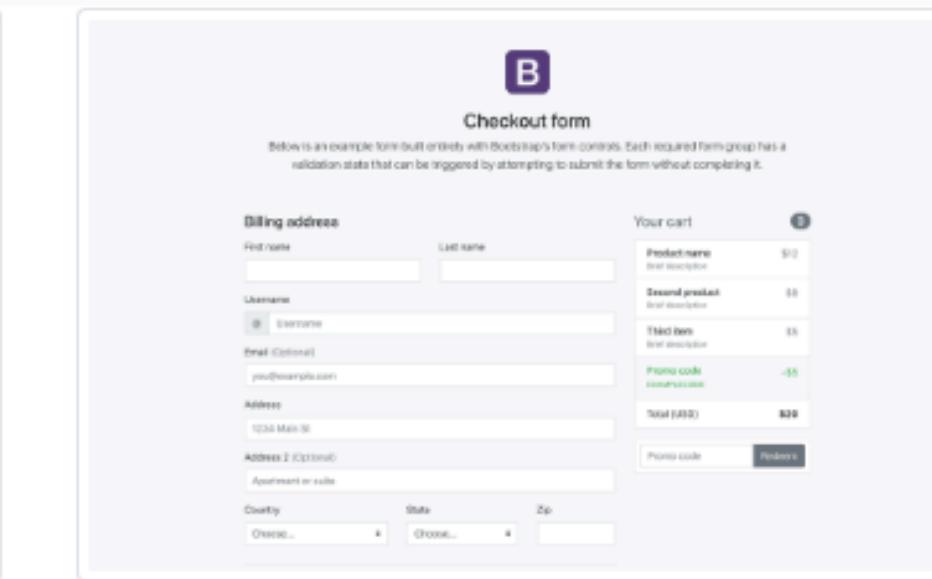
## Album

Simple one-page template for photo galleries, portfolios, and more.



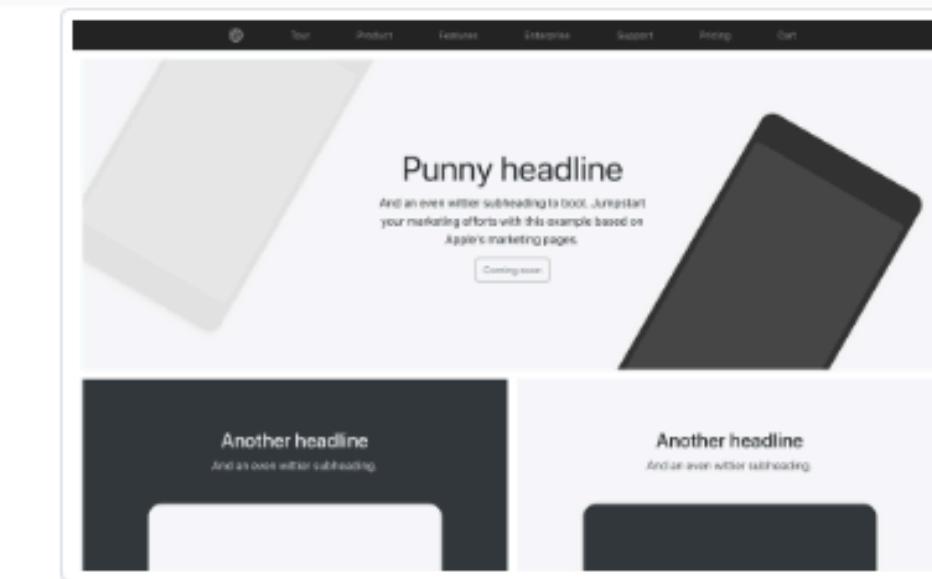
## Pricing

Example pricing page built with Cards and featuring a custom header and footer.



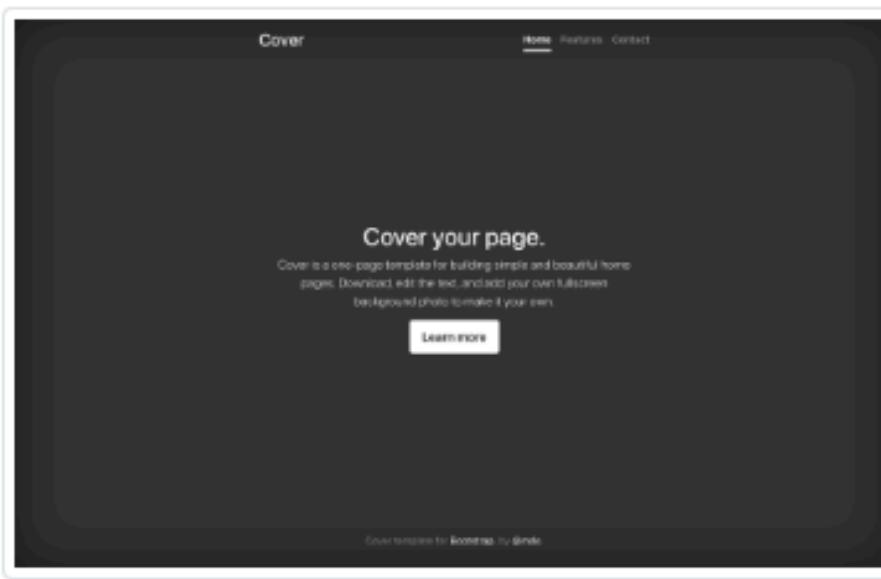
## Checkout

Custom checkout form showing our form components and their validation features.



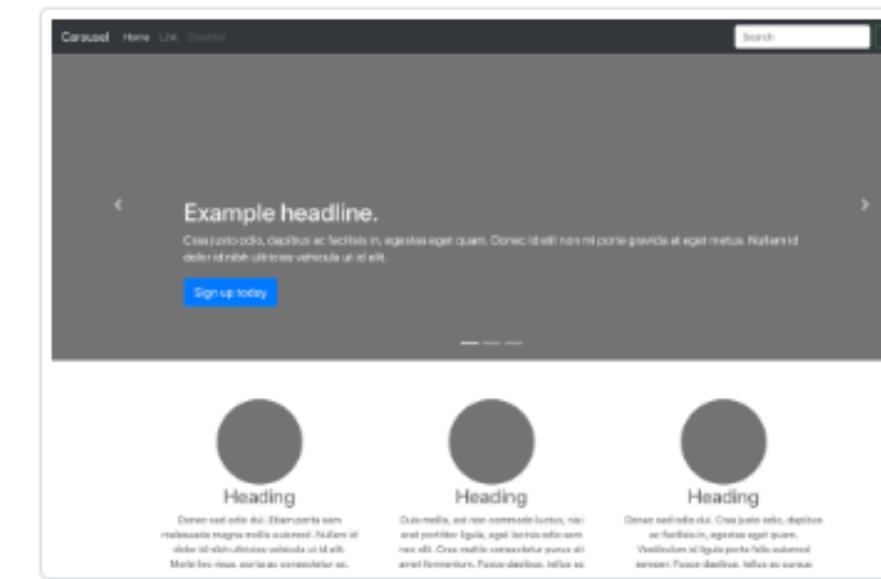
## Product

Lean product-focused marketing page with extensive grid and image work.



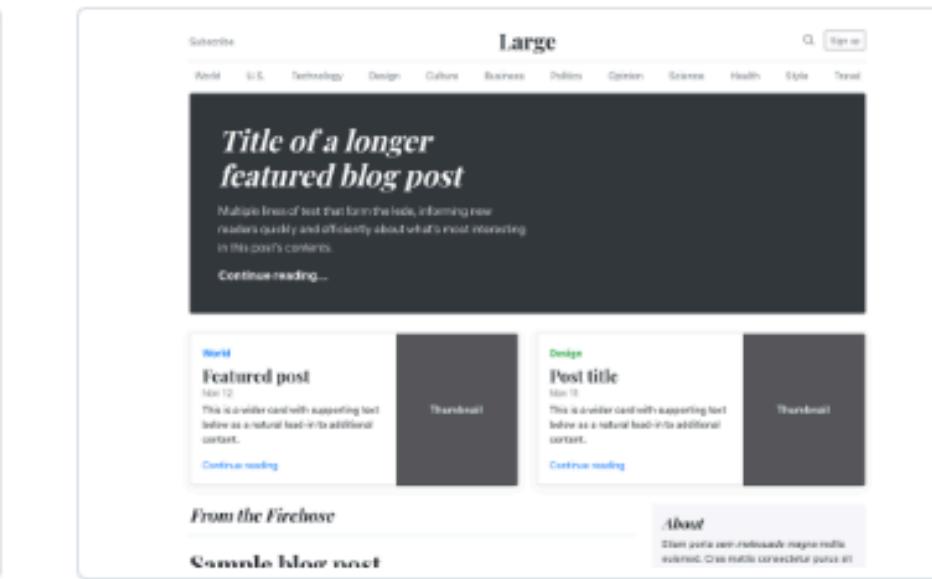
## Cover

A one-page template for building simple and beautiful home pages.



## Carousel

Customize the navbar and carousel, then add some new components.



## Blog

Magazine like blog template with header, navigation, featured content.

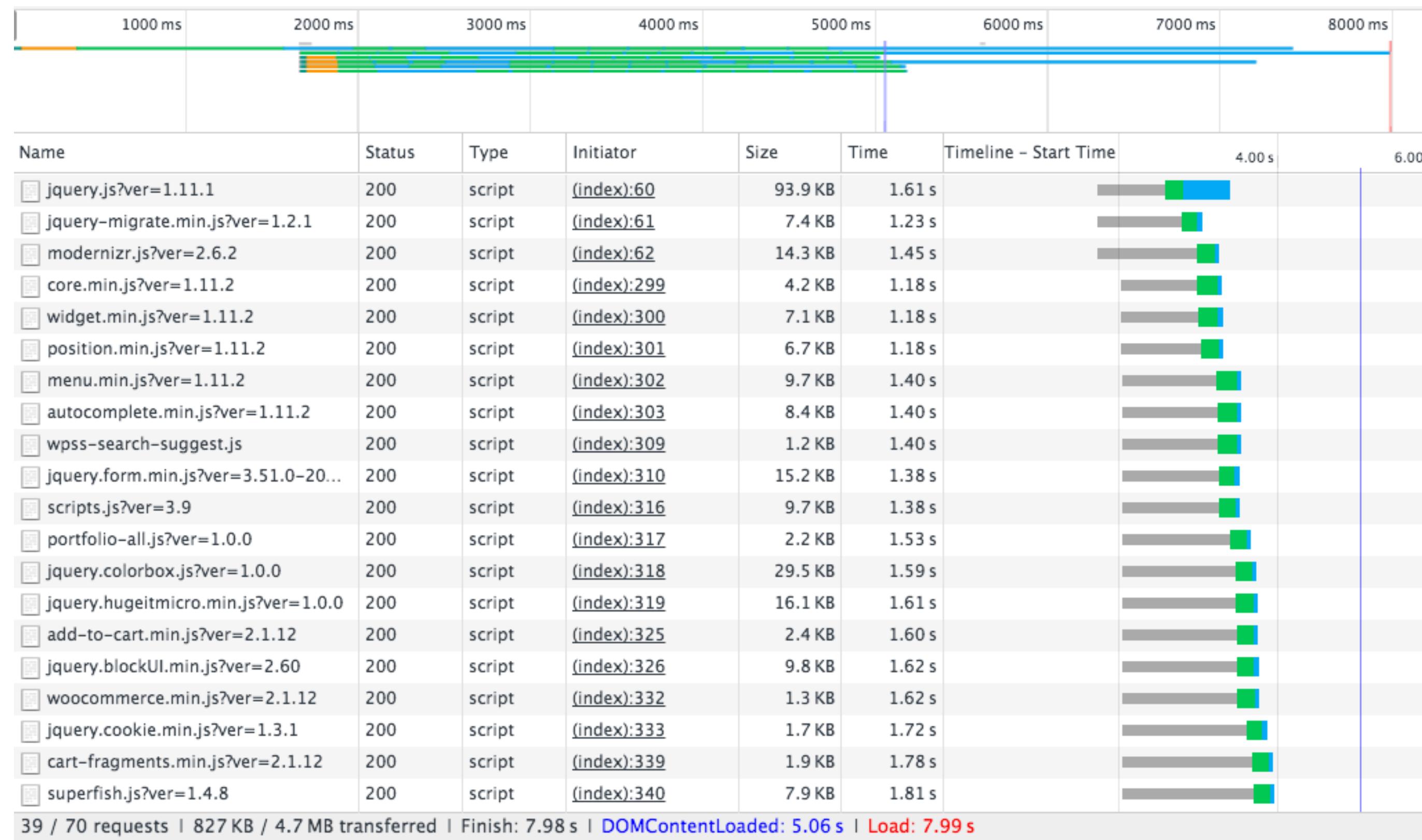


## Dashboard

Basic admin dashboard shell with fixed sidebar and nav bar.

# Opposition to Grid-based frameworks

Can involve loading many files, hurting performance



# Opposition to Grid-based frameworks

## Can stifle creativity

Themes built by or reviewed by Bootstrap's creators.

Why our themes?

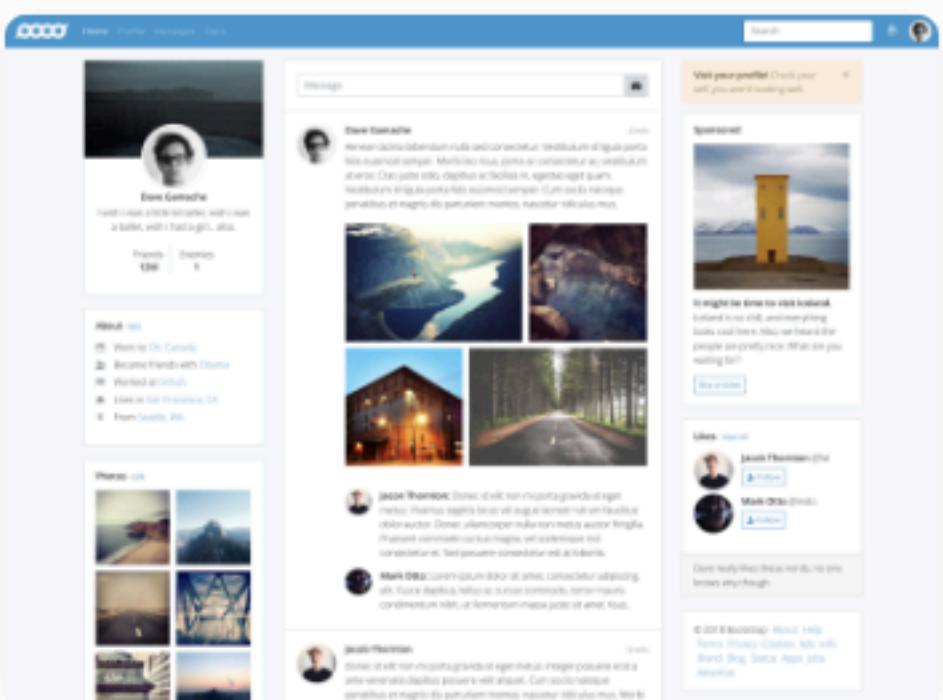
### Built by Bootstrap Team

Component-based frameworks designed, built, and supported by the Bootstrap Team.



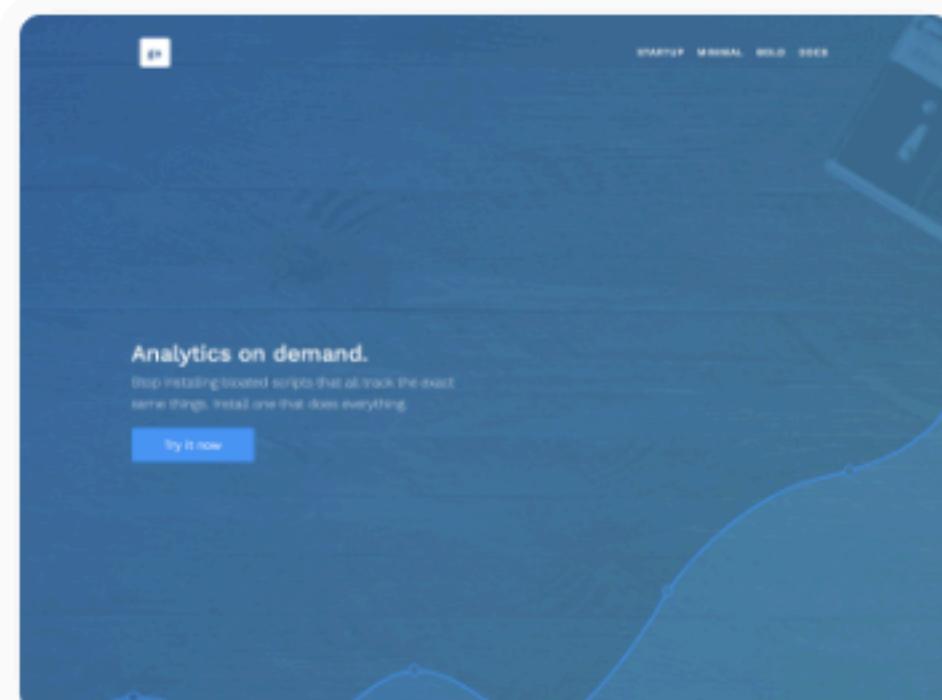
Dashboard  
Admin & Dashboard

\$49.00  
★★★★★



Application  
Application

\$49.00  
★★★★★



Marketing  
Landing & Corporate

\$49.00  
★★★★★

# Today's goals

By the end of today, you should be able to...

- Describe how responsive and adaptive design differ and when you might prefer one or the other
- Explain the advantages and disadvantages of a mobile-first design
- Utilize media queries to create responsive layouts
- Develop grid-based layouts using Bootstrap

# **IN4MATX 133: User Interface Software**

**Lecture 4:**  
**Responsive Design & Bootstrap**

Professor Daniel A. Epstein  
TA Jamshir Goorabian  
TA Simion Padurean