

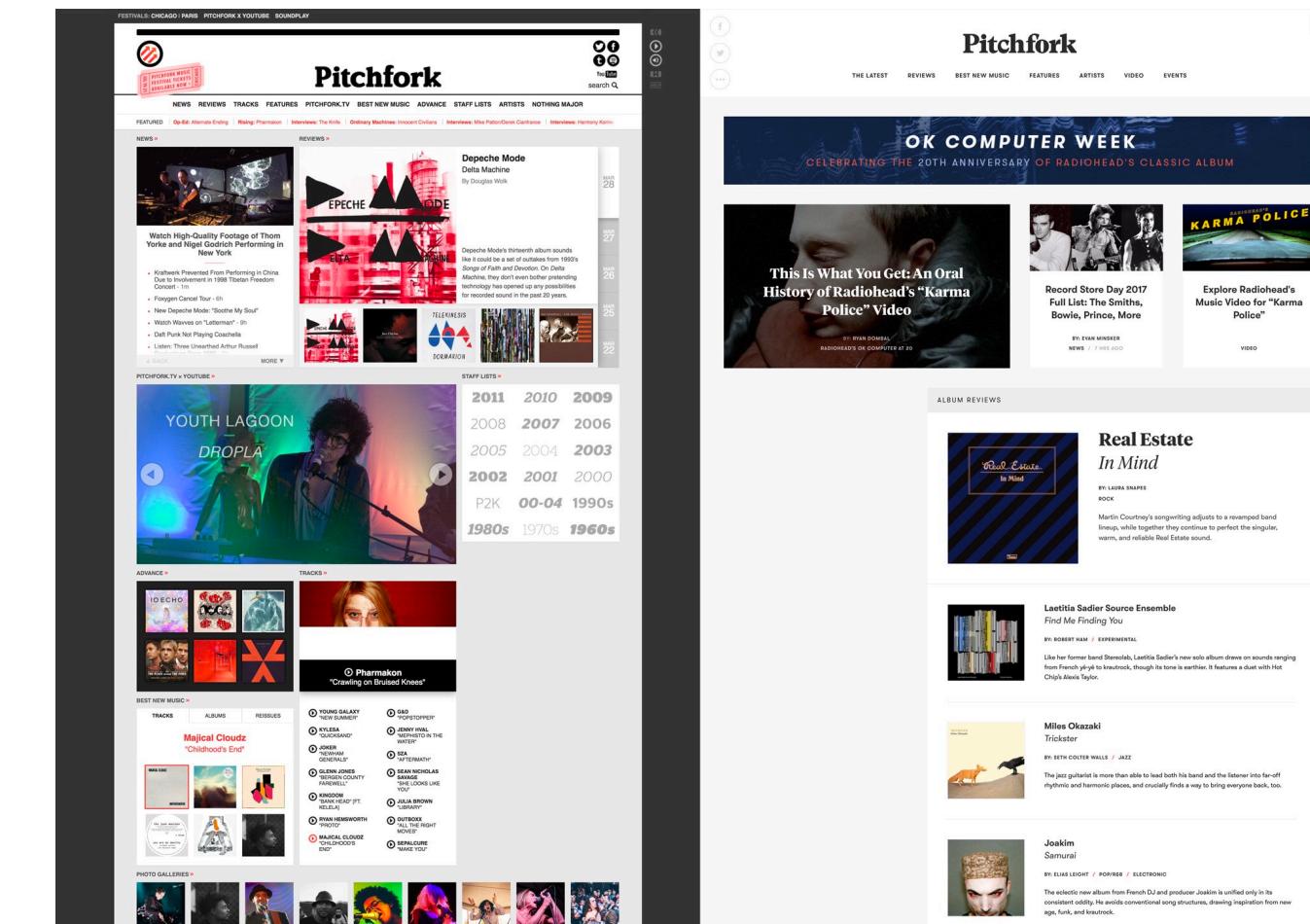
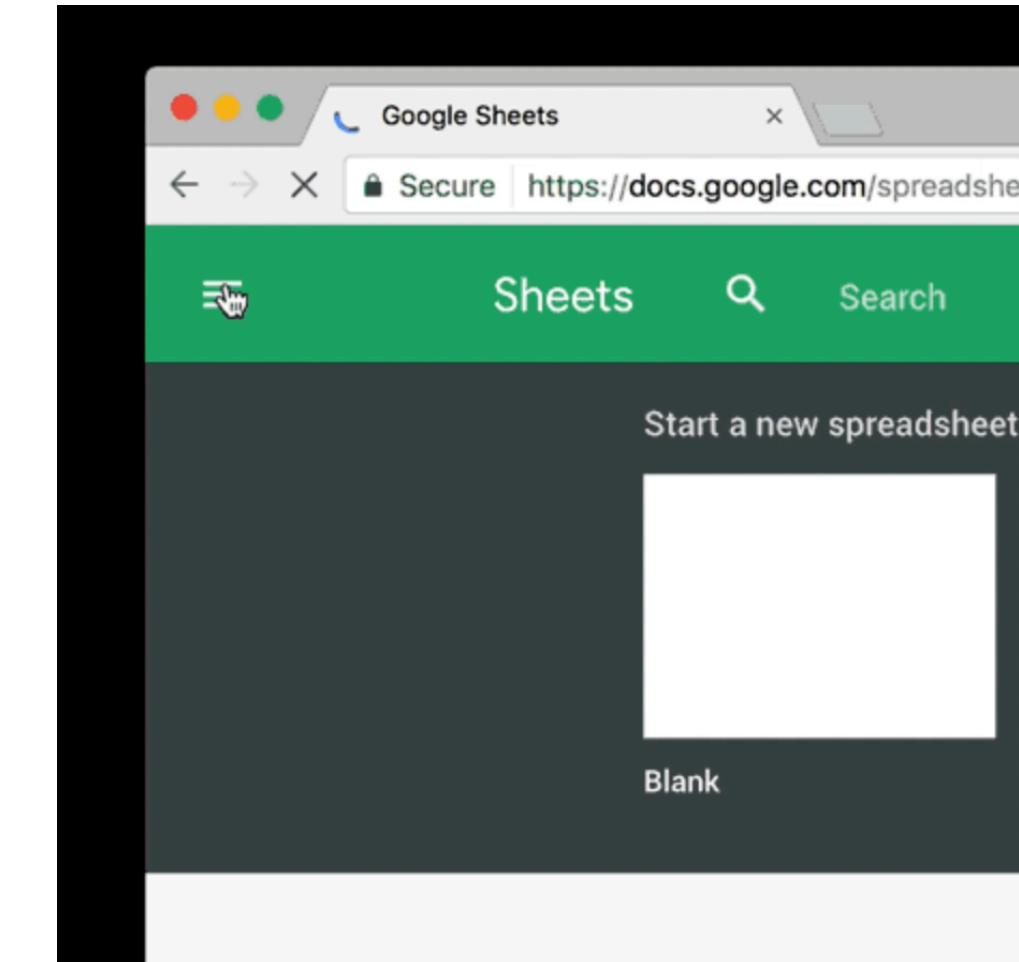
IN4MATX 133: User Interface Software

Lecture 4:
Bootstrap & Javascript 1

Professor Daniel A. Epstein
TA Lucas de Melo Silva
TA Jong Ho Lee

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>

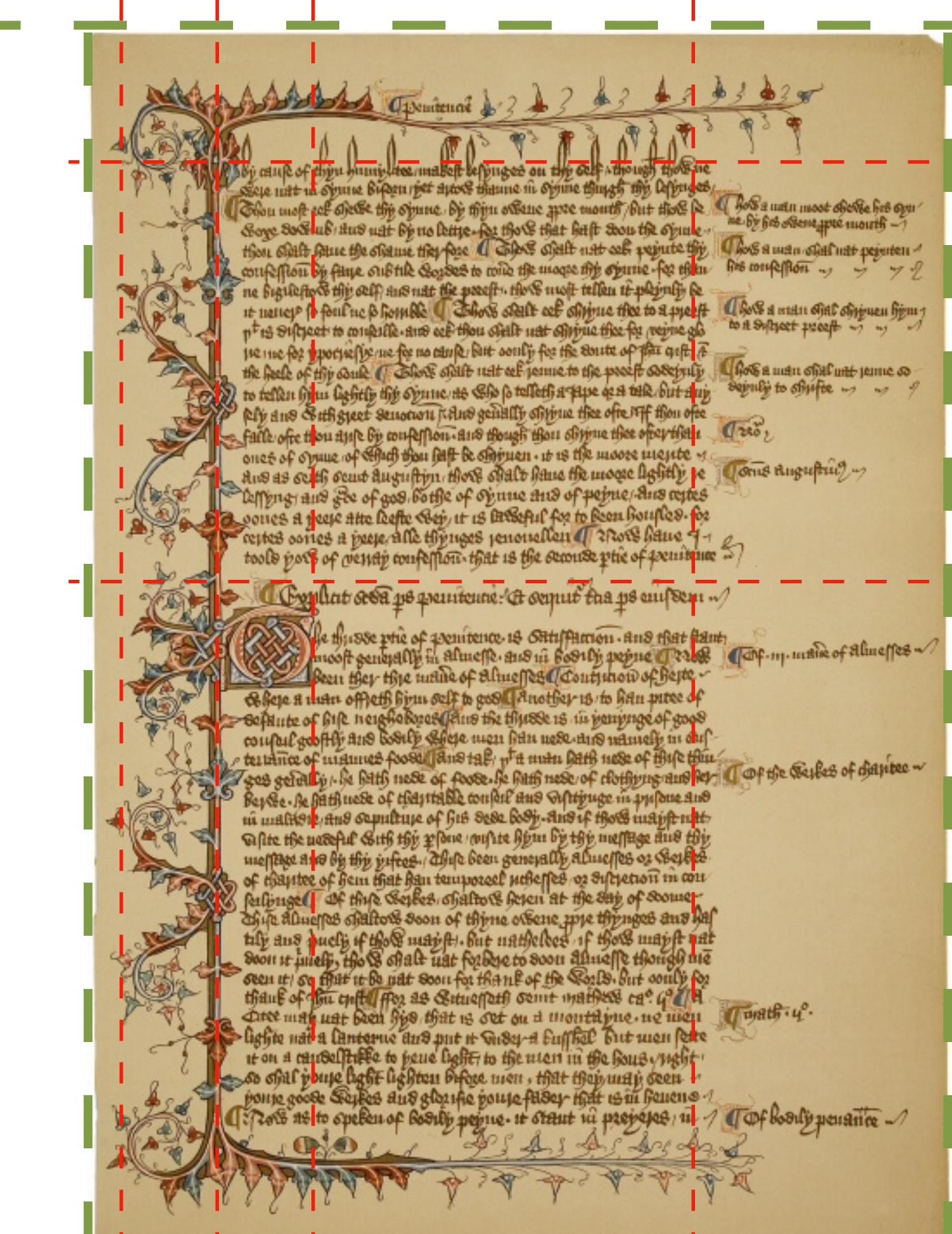
Mobile-first, not mobile-only

- 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

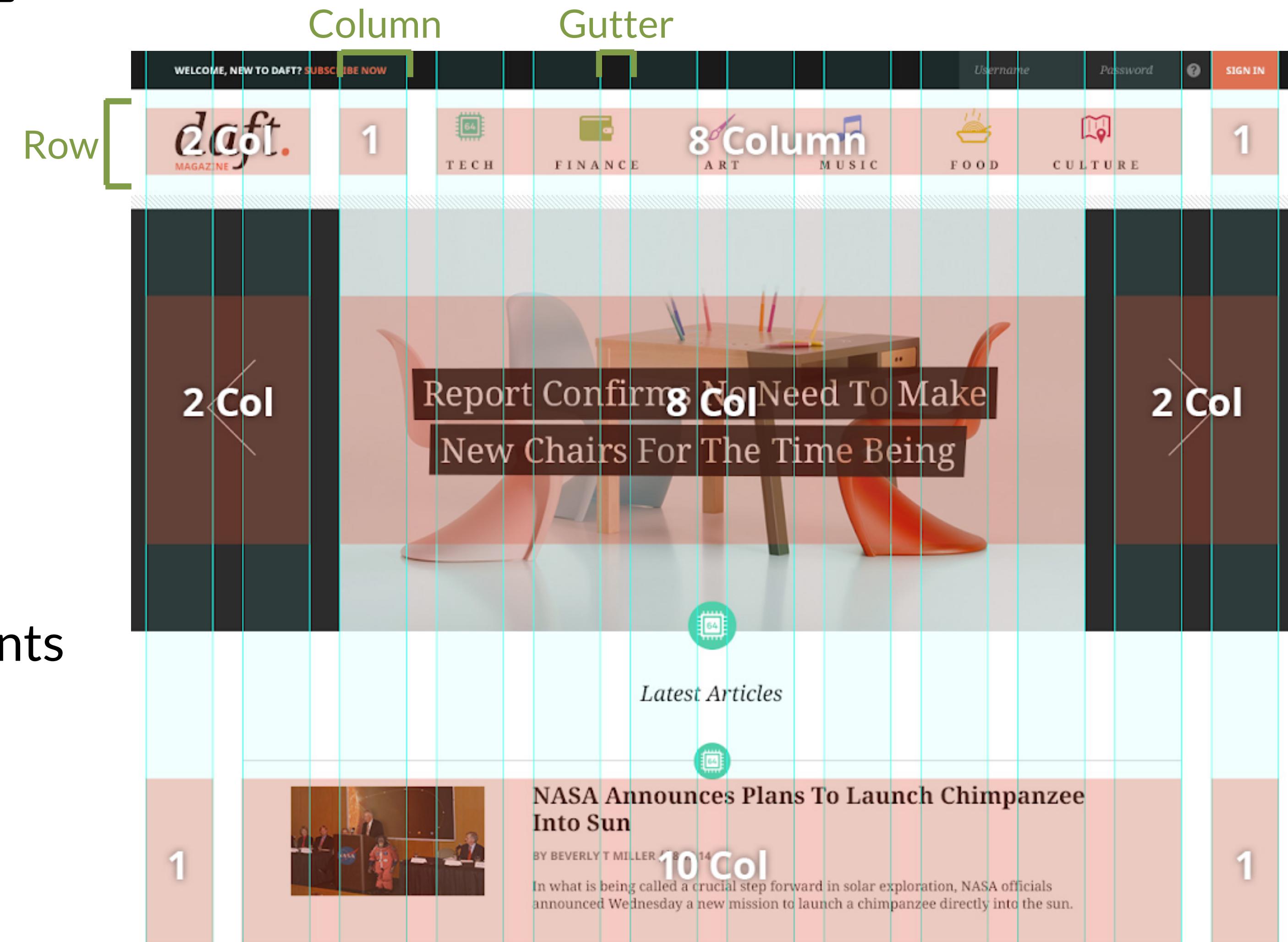
- Established tool for content arrangement
- Gridded content is familiar and easy to follow
- In general, it's good to target fewer lines
- But breaking that rule is important for creativity and attention-grabbing



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

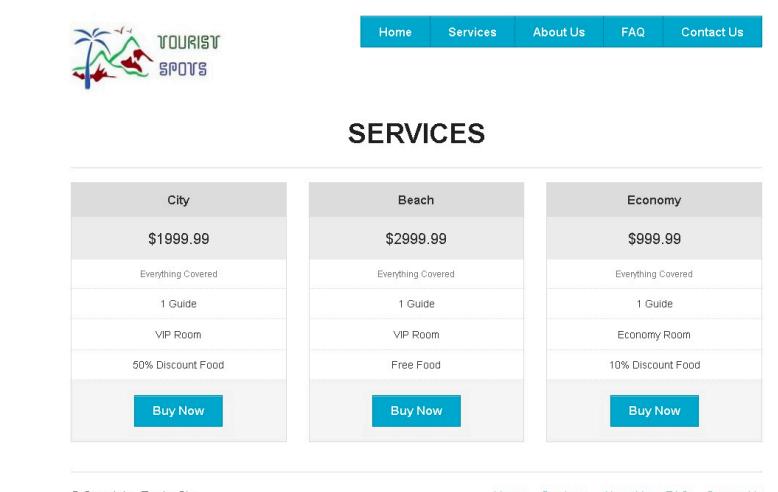
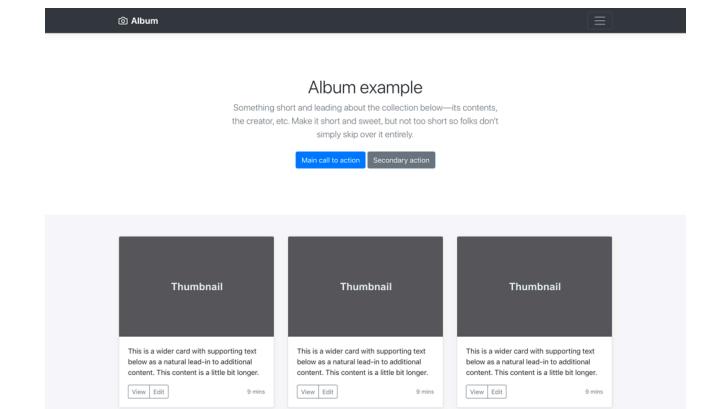
Grid-based layouts

- 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]`

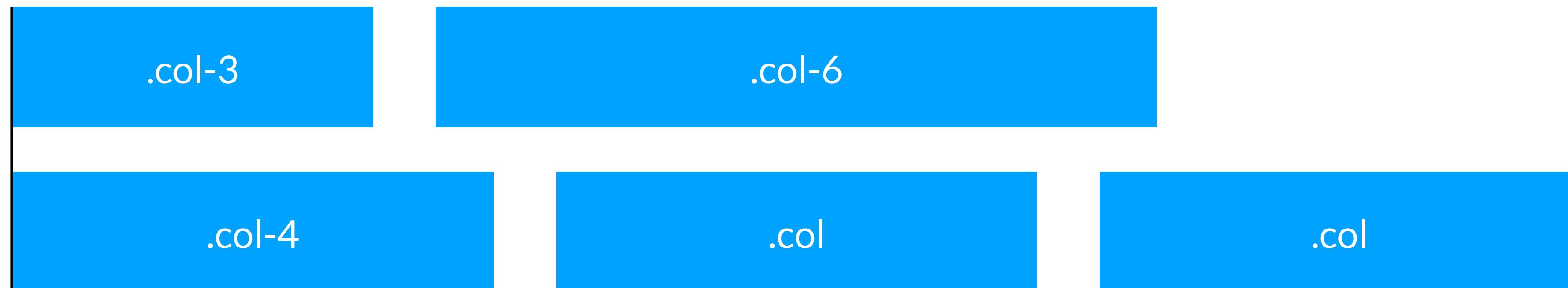
	Extra small devices Phones (<768px)	Small devices Tablets ( $\geq 768px$ )	Medium devices Desktops ( $\geq 992px$ )	Large devices Desktops ( $\geq 1200px$ )
<b>Grid behavior</b>	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
<b>Container width</b>	None (auto)	750px	970px	1170px
<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>
```

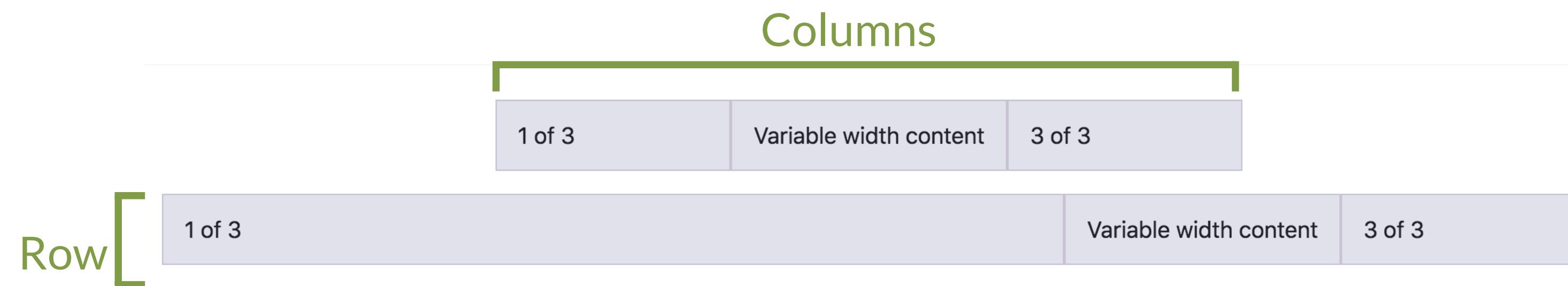


# Bootstrap

## Grid System

- Rows are block elements, while columns are inline

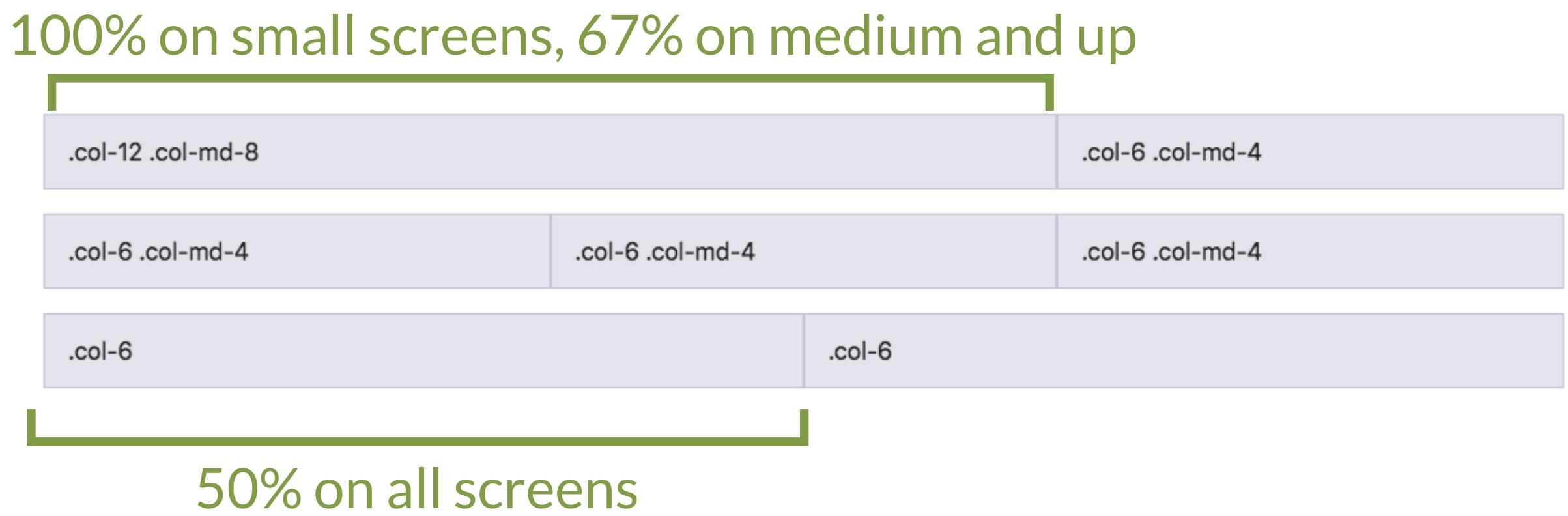
<https://getbootstrap.com/docs/4.1/layout/grid/>



# 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

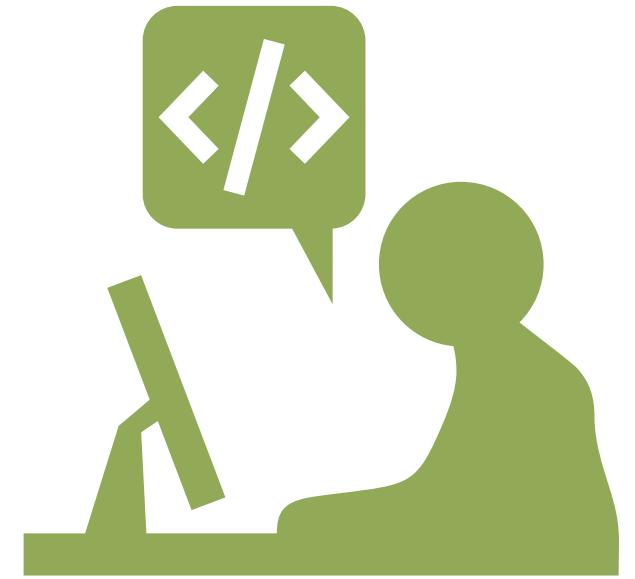


# Bootstrap

## Grid System

- Display property supports applying a display type (like `none` or `block`), can be combined with size properties
- `.d-none` will hide on all sizes
- `.d-md-none` will hide everything md and larger
- `.d-md-block .d-none` will hide only on xs

# Responsive class schedule

A screenshot of a Mac OS X application window titled "My 133 Schedule". The window contains a title bar, a menu bar with items like File, Edit, View, and Help, and a toolbar with icons for Back, Forward, Stop, Refresh, and others. Below the toolbar is a search field and a status bar. The main content area is titled "My 133 Schedule" and features a weekly grid from Monday to Friday. The grid has five columns labeled "Monday", "Tuesday", "Wednesday", "Thursday", and "Friday". The first two columns (Monday and Tuesday) have green header bars, while the others have white header bars. The grid contains the following data:

Monday	Tuesday	Wednesday	Thursday	Friday
Discussion	Lecture		Lecture	
Discussion	Lecture		Lecture	
Assignment due				

My 133 Schedule

Monday	Tuesday	Wednesday	Thursday	Friday
Discussion	Lecture		Lecture	
Discussion	Lecture		Lecture	
Assignment due				

# Question

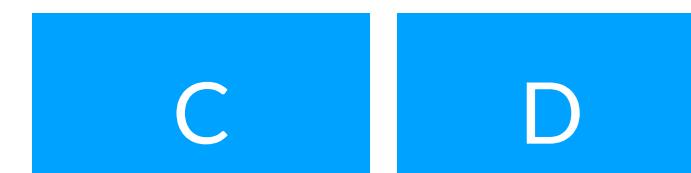
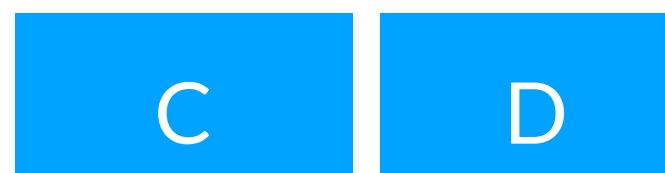
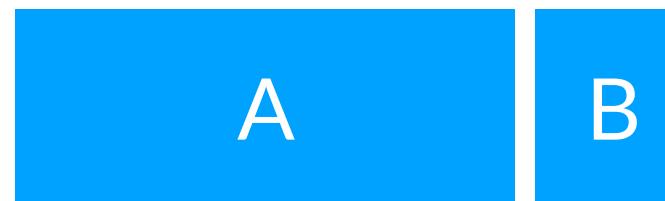


Which code creates

xs screen (phone)



lg screen (laptop)



?

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>
```

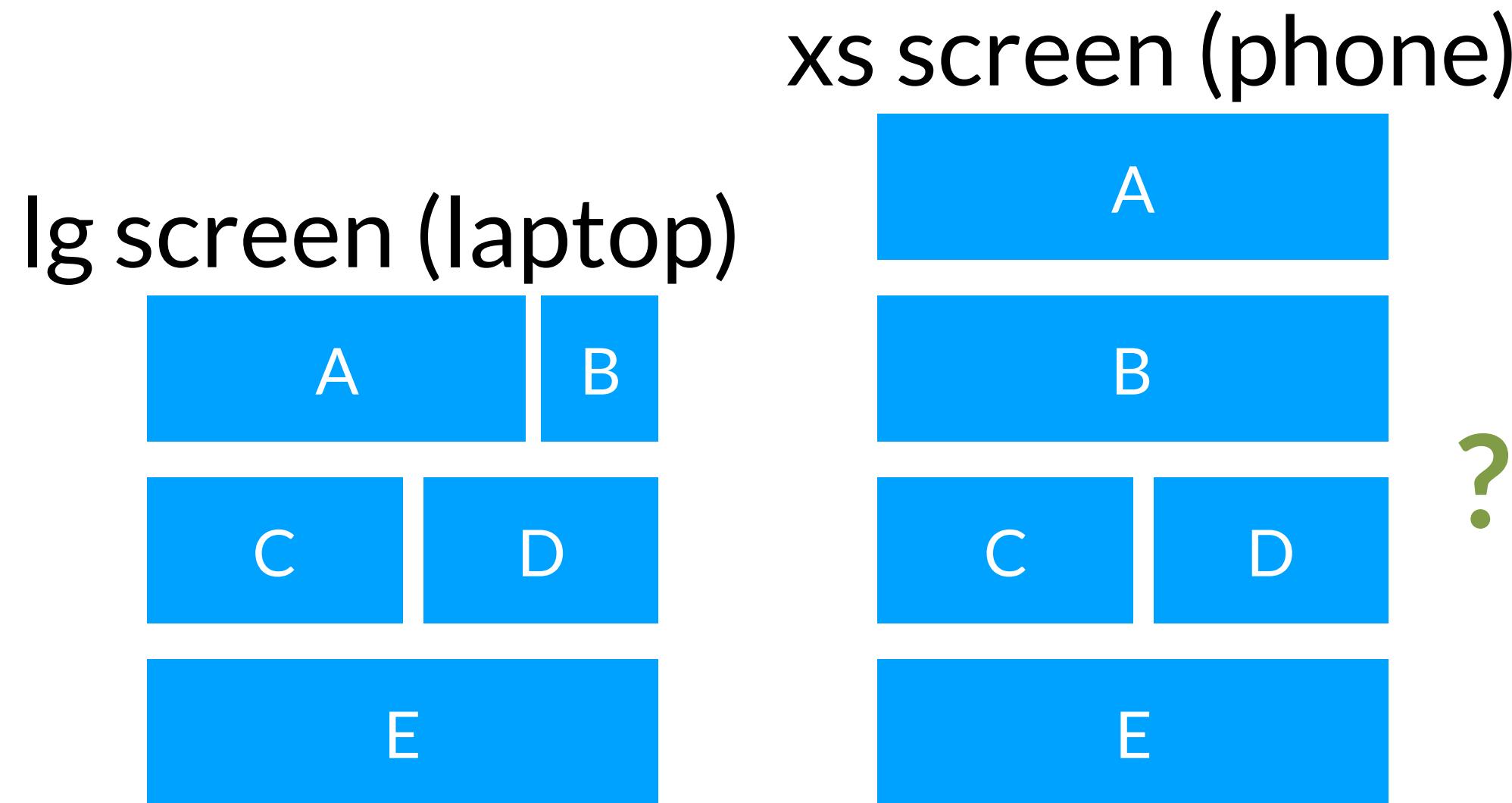
D A and B

E B and C

# 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](#)

Copy

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

```
<!-- 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>
```

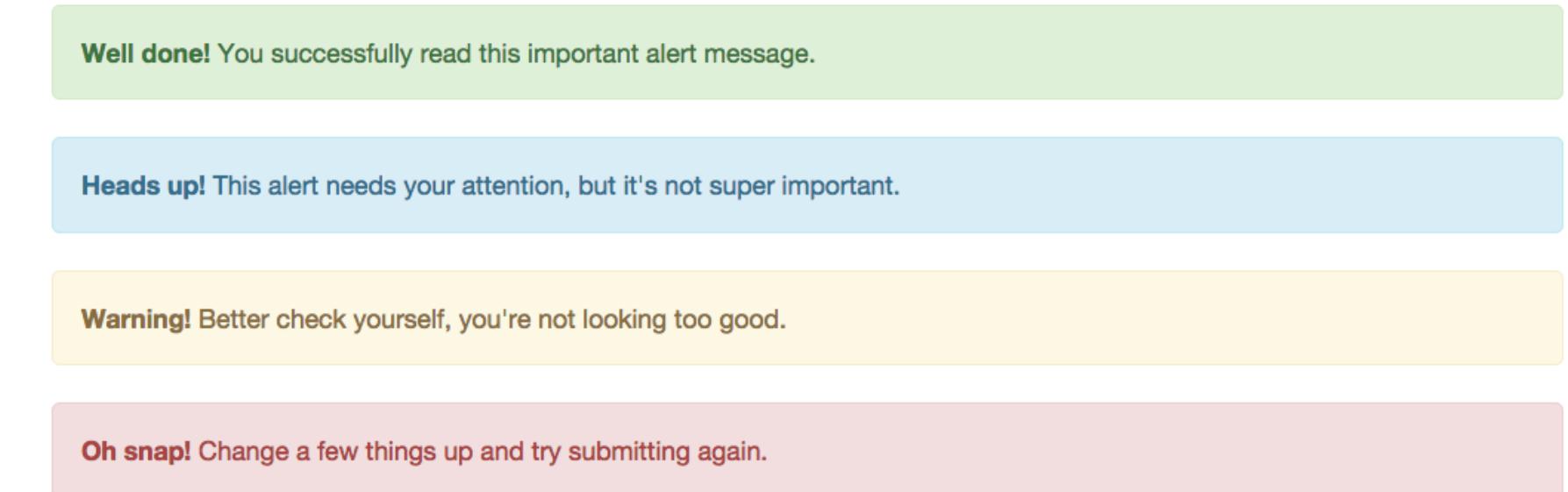
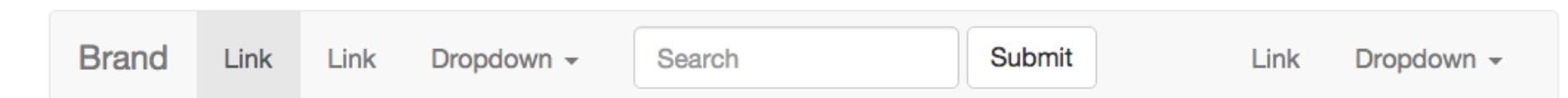
<http://getbootstrap.com/css>

# 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>



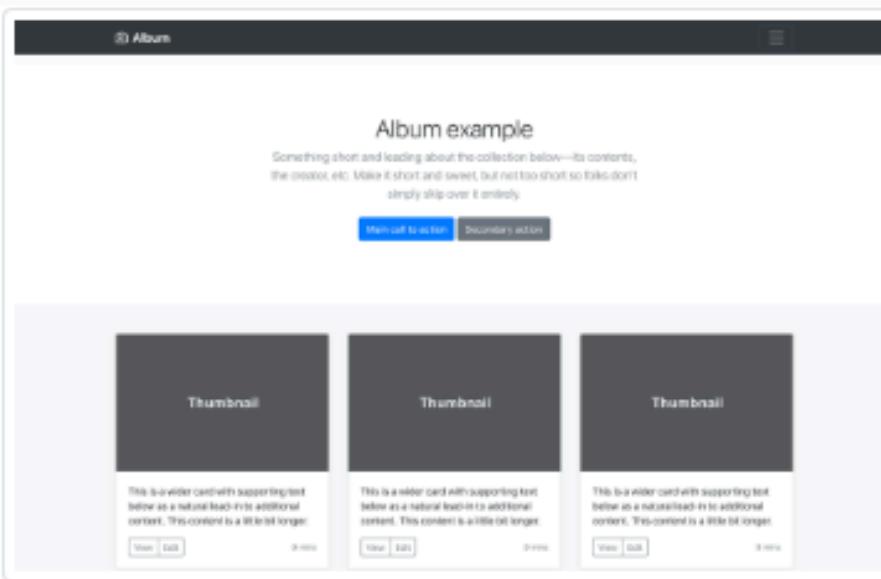
Panel heading
Some default panel content here. Nulla vitae elit libero, a pharetra augue. Aenean lacinia bibendum nulla sed consectetur. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Nullam id dolor id nibh ultricies vehicula ut id elit.
Cras justo odio
Dapibus ac facilisis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

**Grid frameworks  
make development easier.**

**What are the downsides?**

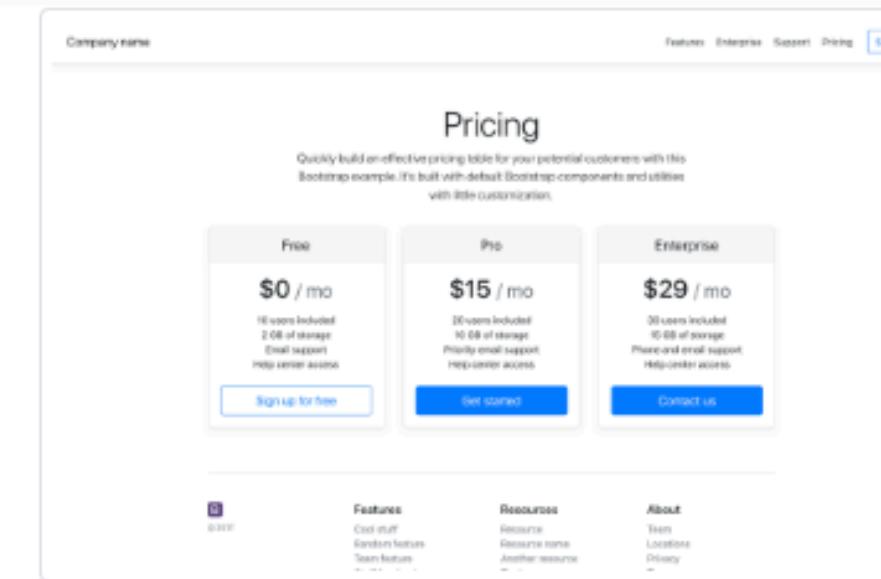
# Opposition to Grid-based frameworks

## Can lead to similar-looking webpages



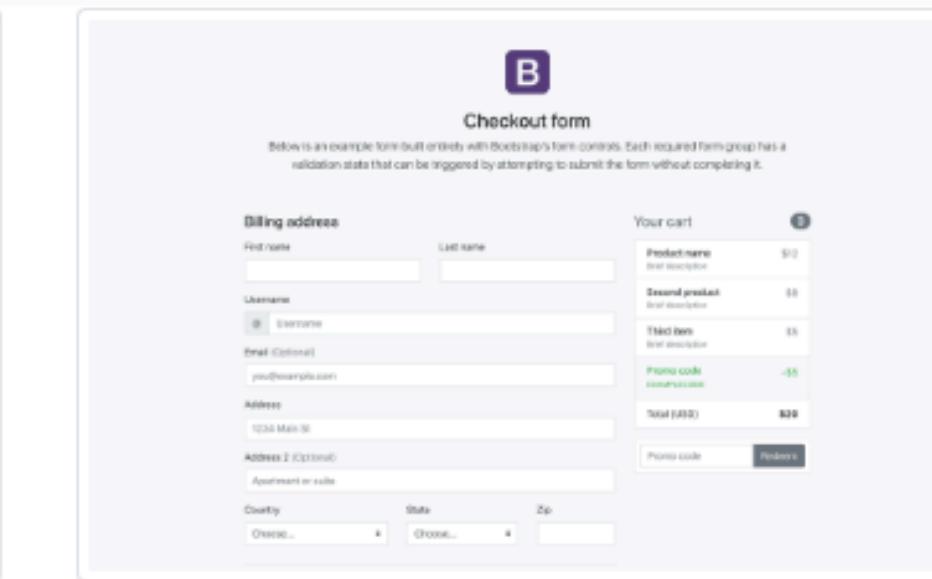
### Album

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



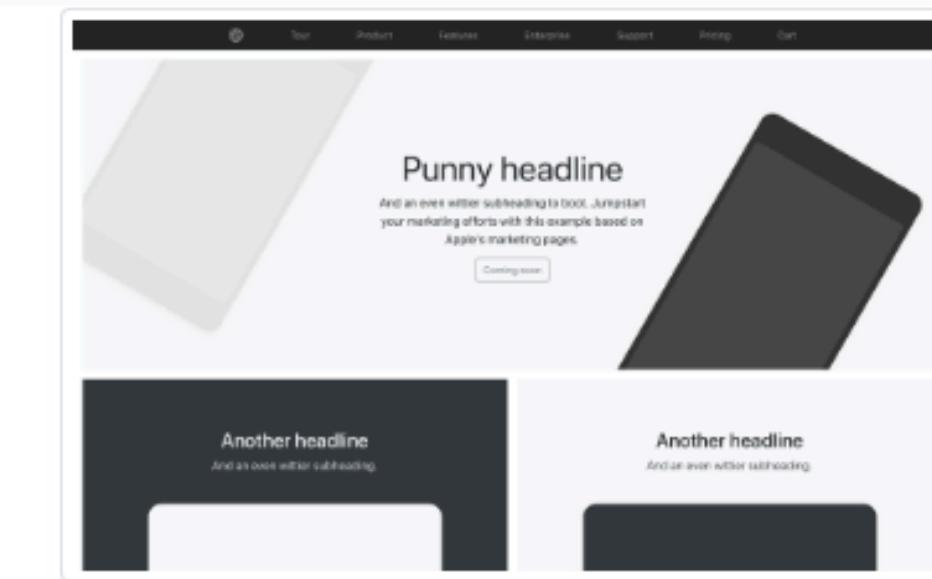
### Pricing

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



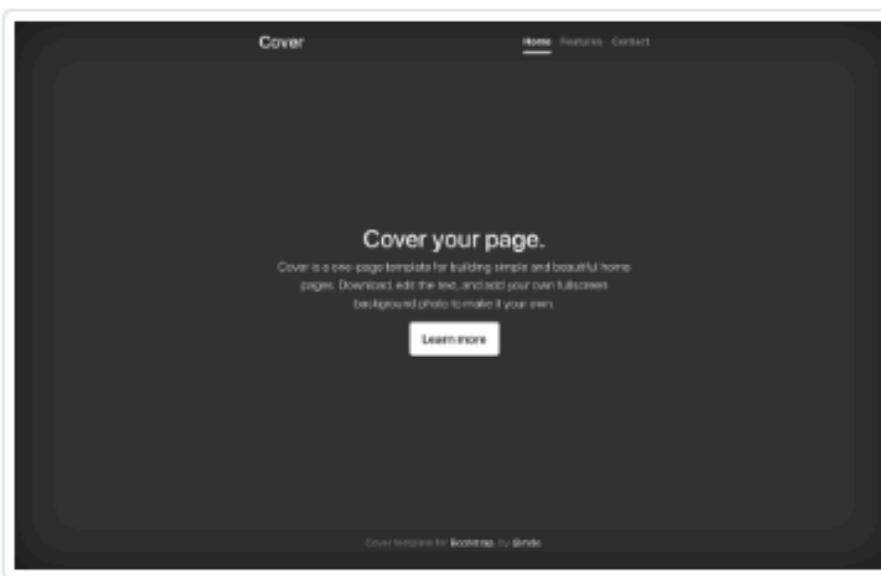
### Checkout

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



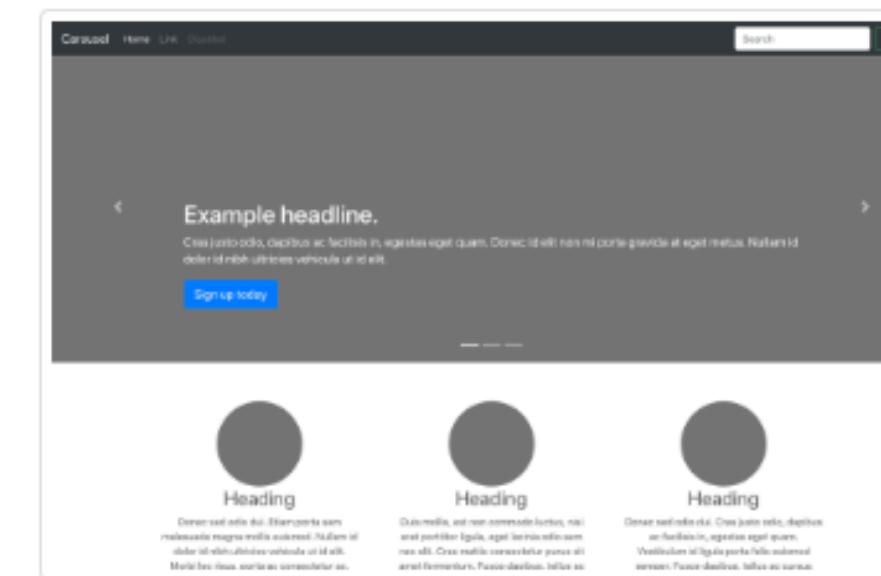
### Product

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



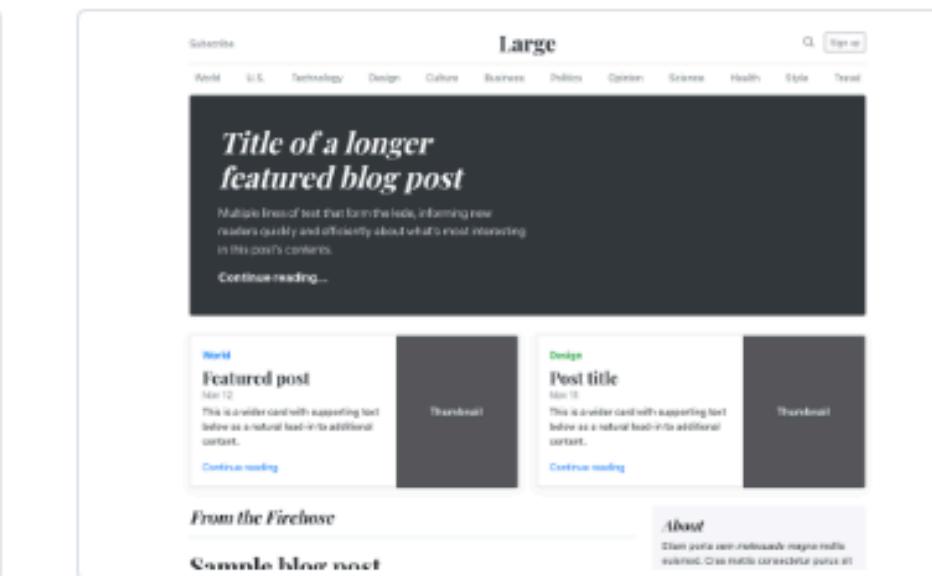
### Cover

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



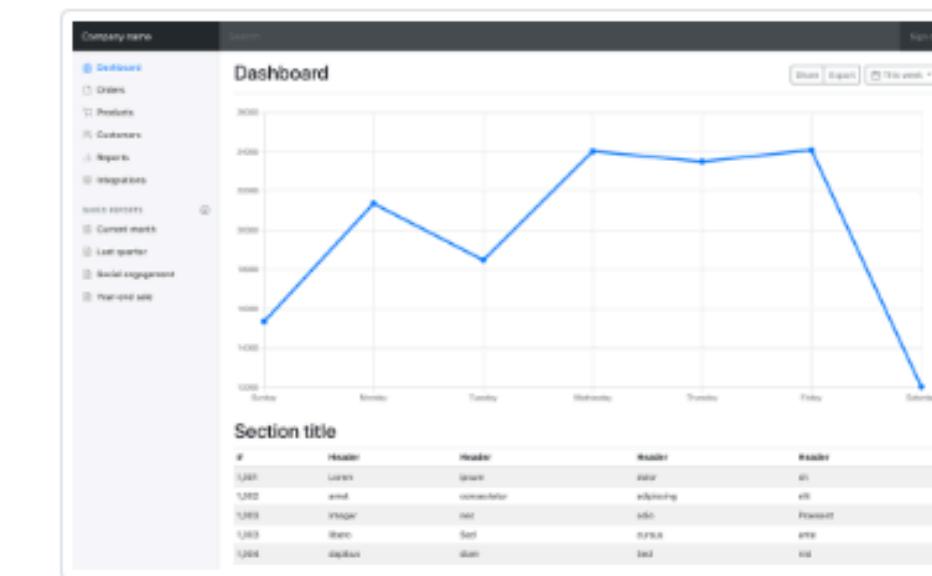
### Carousel

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?

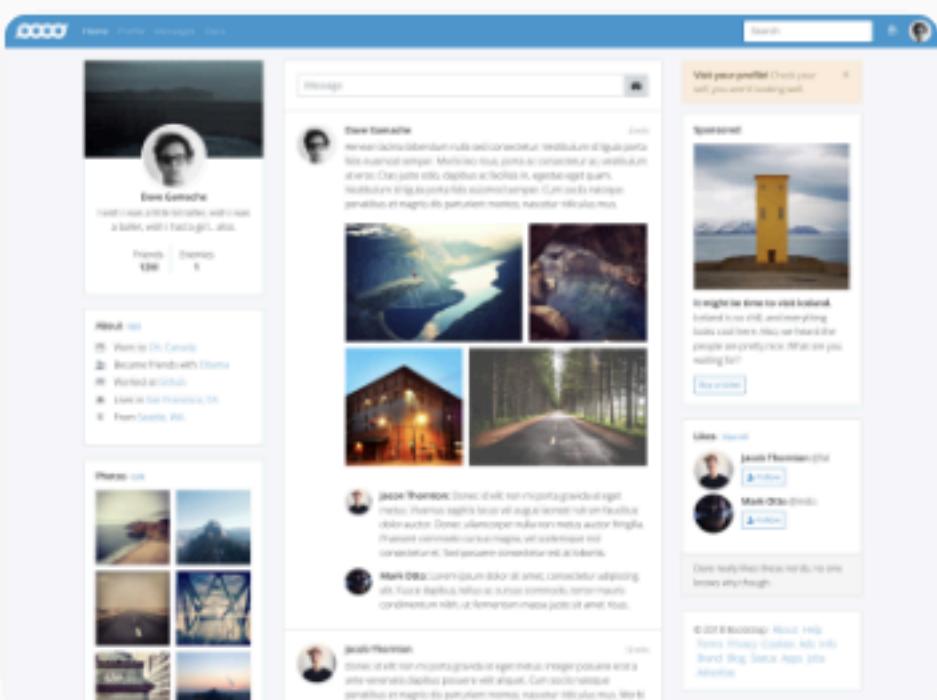
### Built by Bootstrap Team

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



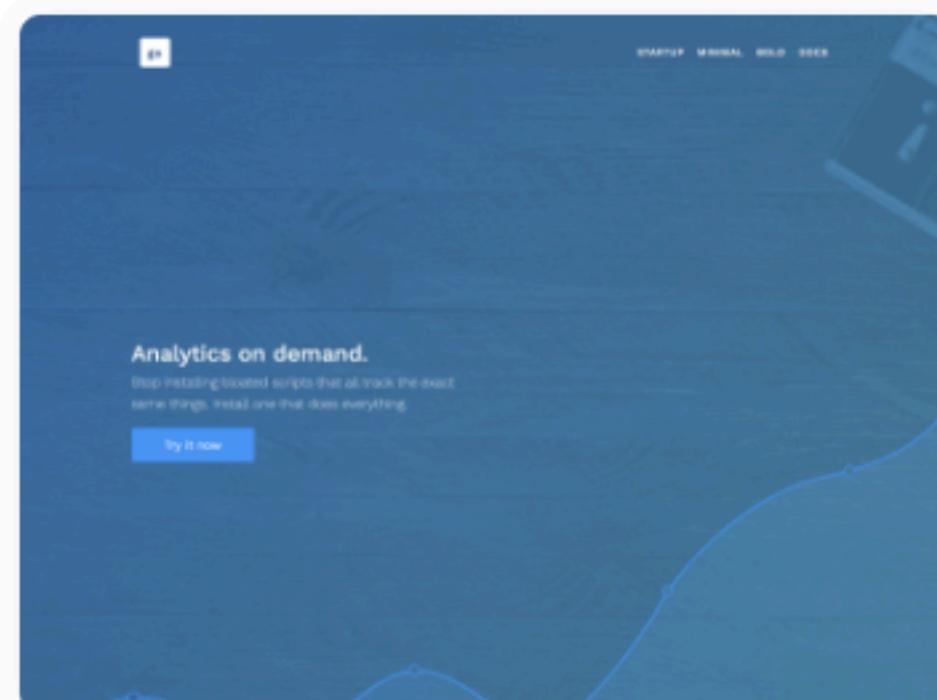
Dashboard  
Admin & Dashboard

\$49.00  
★★★★★



Application  
Application

\$49.00  
★★★★★



Marketing  
Landing & Corporate

\$49.00  
★★★★★

# **Switching gears: Javascript**

# Language Roles



Specify how content  
is rendered



Visually style  
content



Dynamically  
manipulate content

# Language Roles



Markup  
language



Styling  
language



Programming  
language

# Why JavaScript?

- Make pages dynamic
- Make pages personalized
- Make pages interact with other sources, like databases and APIs



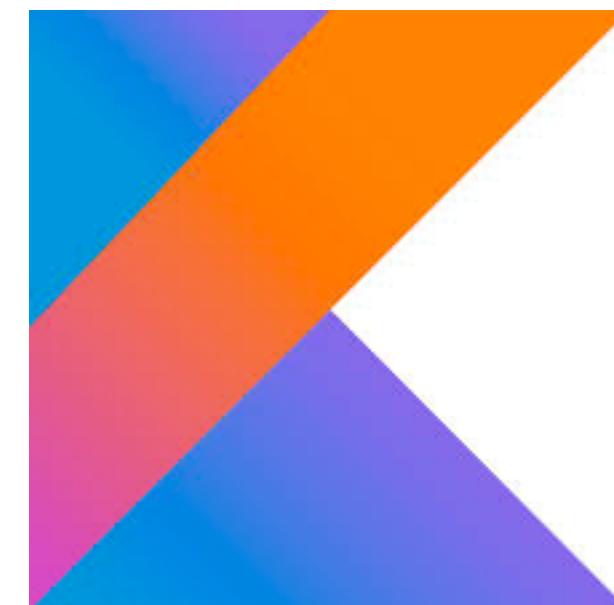
# Other web programming languages

- Ruby, via Ruby on Rails
- Python, via Django or web2py
- These days, you can  
create a dynamic website  
in almost any language



# Other web programming languages

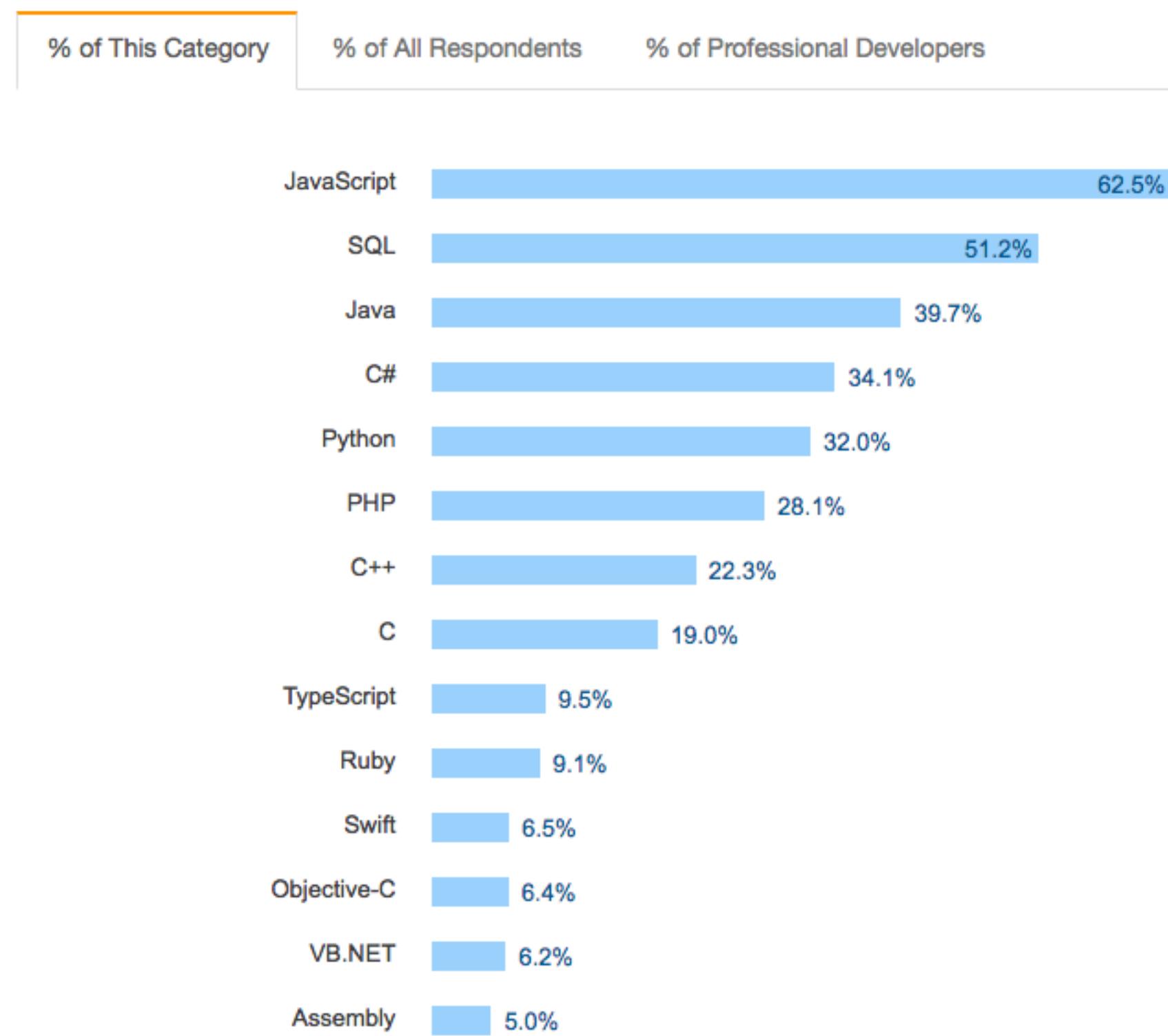
- Some languages transpile to JavaScript
- TypeScript, by Microsoft, introduces types
  - More on TypeScript later
- Kotlin, by Google, runs on the Java virtual machine and compiles to JavaScript
  - Links all of Google's platforms



# JavaScript's popularity

## 🏆 Most Popular Technologies

### Programming Languages

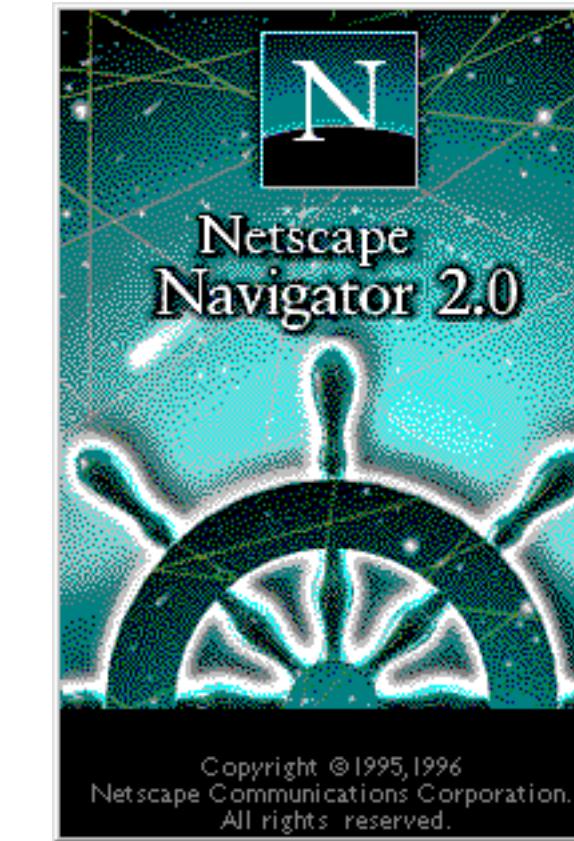


<https://insights.stackoverflow.com/survey/2017#technology-programming-languages>

**How did JavaScript become  
the most popular language  
for web development?**

# History of JavaScript

- “Developed under the name Mocha, the language was officially called LiveScript when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it later was renamed JavaScript”



- Java's popularity was on the rise
  - Marketing ploy
  - Intended to be the “web” language to Java’s “desktop”

<https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17>

# History of JavaScript

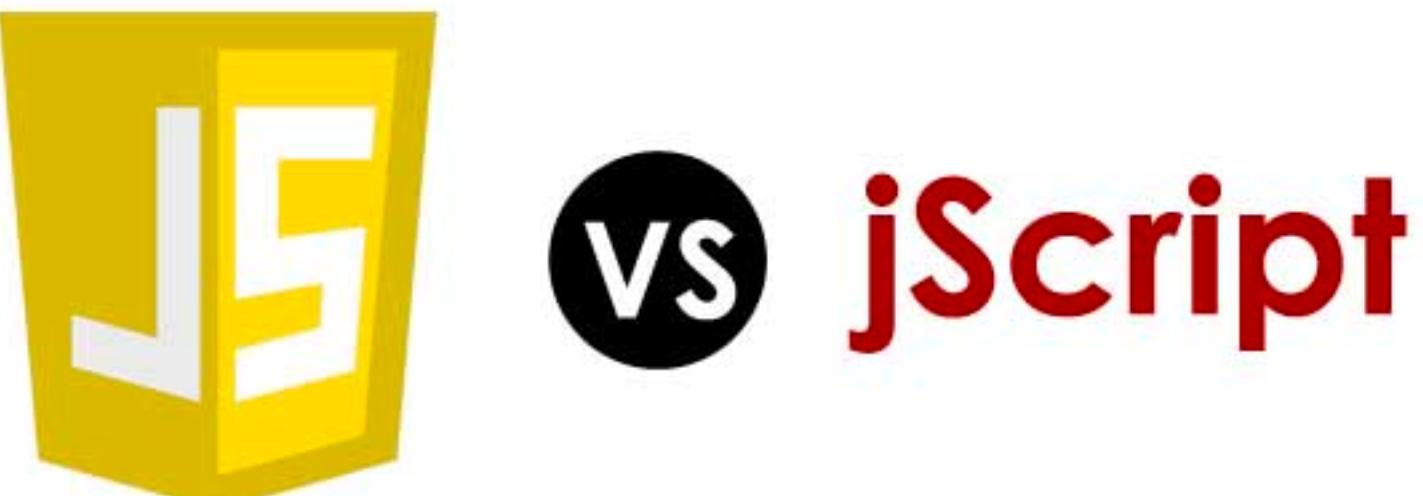
- Netscape submitted JavaScript to ECMA International for consideration as an industry standard
- Subsequent versions were standardized as “ECMAScript”



European Computer Manufacturers Association

# History of JavaScript

- Alternatives started springing up in the late 1990s and early 2000's
  - Microsoft introduced JScript engine
  - Macromedia Flash was popular for facilitating the dynamic web
- Both were vaguely JavaScript-like, but standards differed



# History of JavaScript

- Standards later converged
  - Firefox came out in 2005
  - Adobe bought Flash
  - JScript followed the standards
- But browser's implementations of the language still vary

Feature name	Current browser	PhantomJS 2.0	iOS7/8	CH 23+, OP 15+	IE 10+	WebKit	SF 6+	BESEN	FF 21+	Android 4.4+	OP 12.10	IE 9	EJS	Rhino 1.7	Konq 4.13
Object.create	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

# History of JavaScript

## ● JavaScript Engines

- SpiderMonkey (Firefox)
- V8 (Chrome)
- JavaScriptCore (Safari)
- Carakan (Opera)
- Chakra (IE & Edge)

Feature name	Current browser	PhantomJS 2.0	iOS7/8	CH 23+, OP 15+	IE 10+	WebKit	SF 6+	BESEN	FF 21+	Android 4.4+	OP 12.10	IE 9	EJS	Rhino 1.7	Konq 4.13
Object.create	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	c Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

# Versions of JavaScript

- You may see references to ECMAScript
- ECMAScript is just the standard for JavaScript
  - The last “major” release was ECMAScript 6, or ES6, or ECMAScript 2015, or ES2015
  - The latest is ECMAScript 2018, or ES9, or ES2018

# Versions of JavaScript

- Engines/Browsers continually play catch-up,  
so many tools support slightly older versions of the standard

Feature name	►	Current browser	Compilers/polyfills						Desktop browsers															
			Traceur 2%	Babel 6+ core-js 28%	Babel 7+ core-js 39%	Closure 2018.09 0%	Type-Script + core-js 28%	IE 11 0%	Edge 16 1%	Edge 17 1%	Edge 18 Preview 1%	FF 60 ESR 2%	FF 61 2%	FF 62 5%	FF 63 Beta 8%	FF 64 Nightly 8%	CH 68, OP 55 5%	CH 69, OP 56 8%	CH 70, OP 57 9%	CH 71, OP 58 9%				
<b>Candidate (stage 3)</b>																								
• <a href="#">string.trimming</a>	►	4/4	0/4	4/4	4/4	0/4	4/4	0/4	2/4	2/4	2/4	2/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
• <a href="#">global</a>	►	0/2	0/2	2/2	2/2	0/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
• <a href="#">String.prototype.matchAll</a>	►	No	No	Yes <sup>[4]</sup>	Yes <sup>[4]</sup>	No	Yes <sup>[5]</sup>	No	No	No	No	No	No	No	No	No	Flag <sup>[9]</sup>							
• <a href="#">instance class fields</a>	►	0/3	1/3	1/3	1/3	0/3	1/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
• <a href="#">static class fields</a>	►	0/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
• <a href="#">Function.prototype.toString revision</a>	►	7/7	0/7	0/7	0/7	0/7	0/7	1/7	4/7	4/7	4/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
• <a href="#">Array.prototype.{flat, flatMap}<sup>[10]</sup></a>	►	2/2	0/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
• <a href="#">Symbol.prototype.description</a>	►	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	Yes	No	Flag <sup>[9]</sup>	Yes	Yes	Yes	Yes	Yes	Yes
• <a href="#">BigInt</a>	►	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8
• <a href="#">Object.fromEntries</a>	►	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	No								

# Versions of JavaScript

- Polyfills ensure a user’s browser has the latest libraries
  - Downloads “fill” versions of added functions, re-written using existing functions
- Sometimes called a “shim” or a “fallback”



## ► About

[Browsers and features](#)

[API reference](#)

[Live examples](#)

[Usage stats](#)

[Contributing](#)

[Privacy Policy](#)

[Terms and Conditions](#)

Just the polyfills you need for your site, tailored to each browser. Copy the code to unleash the magic:

```
<script src="https://cdn.polyfill.io/v2/polyfill.min.js"></scr
```

# JavaScript

- Interpreted language
- Executed by a JavaScript engine
- Engine runs the same code that a programmer writes

# Java

- Compiled language (into bytecode)
- Run in a Java Virtual Machine (JVM)
- Bytecode is unreadable by people

# JavaScript

- Standardized through ECMAScript, but discrepancies exist
- Debugging dependent on execution environment
- Prototype based
- Used in every browser without a plugin

# Java

- “Write once, deploy anywhere”
- Bugs found at compile time
- Class-based
- Requires a plugin to be run in most browsers

**JavaScript is just  
a programming language**

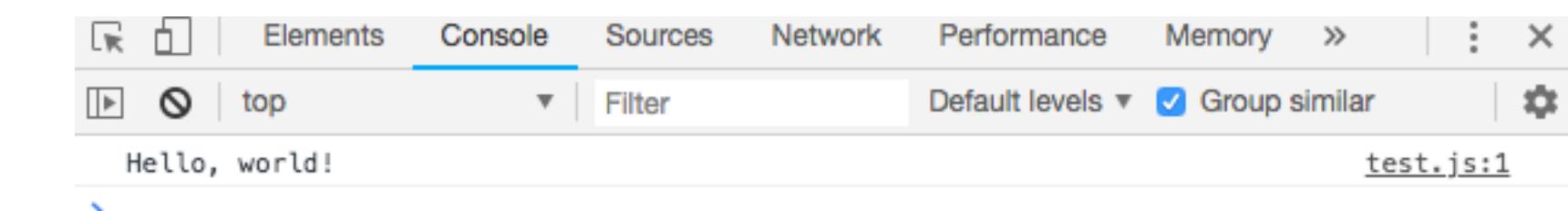
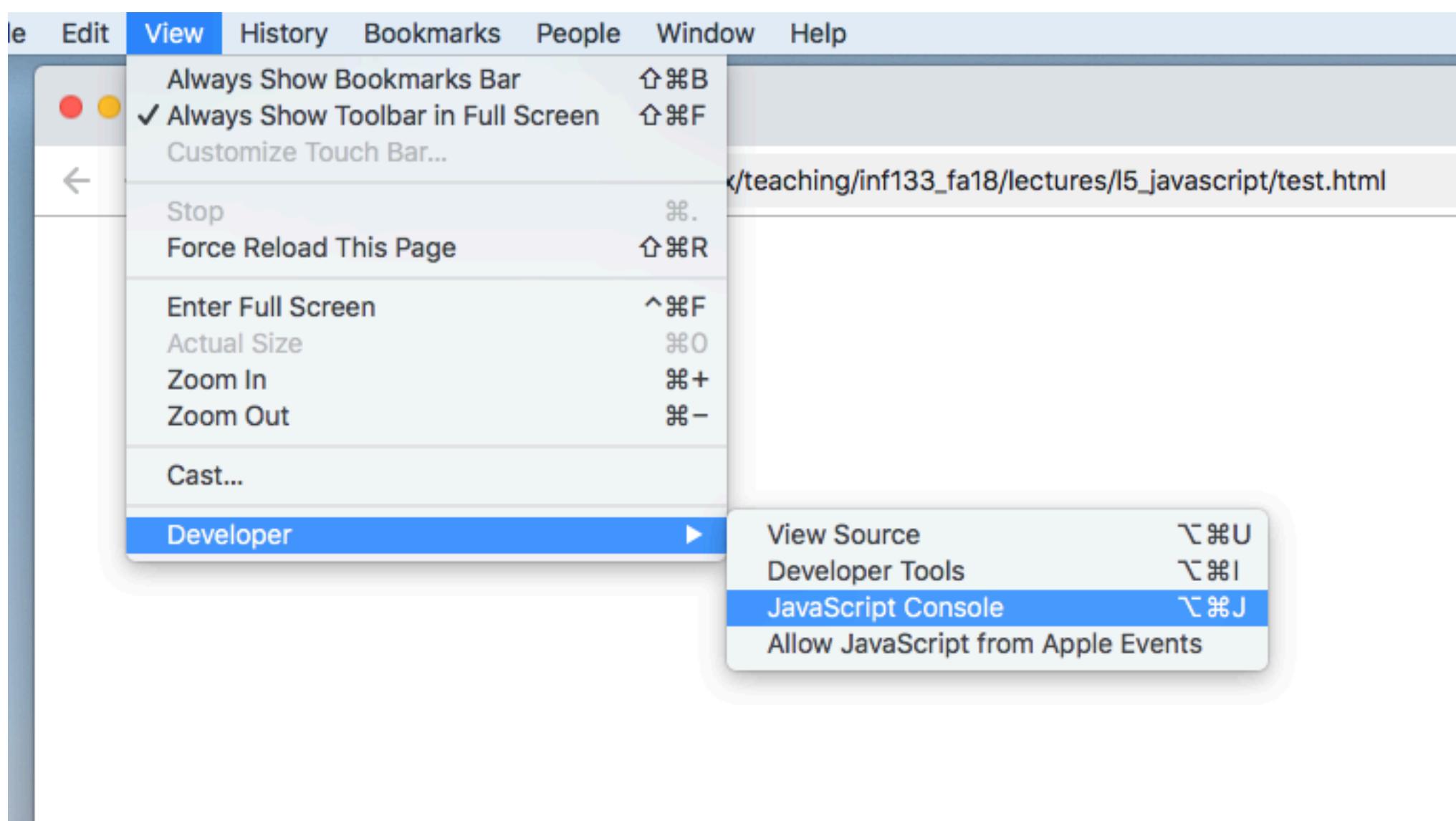
# Loading JavaScript

```
<html>
 <head>
 <script src="test.js"></script>
 </head>
</html>
```

# Printing in JavaScript

```
console.log("Hello, world!");
```

- Won't be visible in the browser
- Shows in the JavaScript Console



&lt;/

# JavaScript Syntax

- Has functions and objects
  - `foo()` `bar.baz`
  - They look like Java, but act differently

# JavaScript Variables

- Variables are dynamically typed

```
var x = 'hello'; //value is a string
console.log(typeof x); //string
```

```
x = 42; //value is now a Number
console.log(typeof x); //number
```

- Unassigned variables have a value of `undefined`

```
var hoursSlept;
console.log(hoursSlept);
```

# JavaScript types

```
console.log('40' + 2); // '402'
console.log('40' - 4); // 36 ← Minus isn't defined for strings,
 so JavaScript knows to convert this
```

```
var num = 10;
var str = '10';
```

```
// comparisons: these will all be booleans (true/false)
console.log(num == str); // true
console.log(num === str); // false
console.log('' == 0); // true
```

# JavaScript loops and conditionals

```
var i = 4.4;
```

```
if(i > 5) {
 console.log('i is bigger than 5');
} else if(i >= 3) {
 console.log('i is between 3 and 5');
} else {
 console.log('i is less than 3');
}
```

```
for(var x = 0; x < 5; x++) {
 console.log(x);
}
```

# JavaScript methods

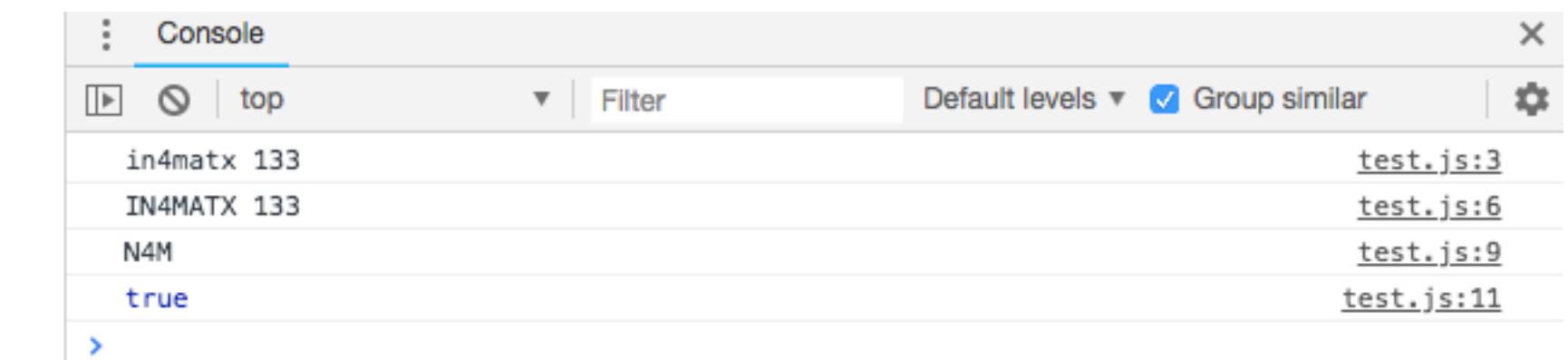
- Called with dot notation

```
var className = 'in4matx 133';
console.log(className);
```

```
className = className.toUpperCase();
console.log(className);
```

```
var part = className.substring(1, 4);
console.log(part);
```

```
console.log(className.indexOf('MATX') >= 0); //whether
the substring appears
```



# JavaScript arrays

- Similar to Java, but can be a mix of different types

```
var letters = ['a', 'b', 'c'];
var numbers = [1, 2, 3];
var things = ['raindrops', 2.5, true, [5, 9, 8]]; //arrays can be nested
var empty = [];
var blank5 = new Array(5); //empty array with 5 items
```

```
//access using [] notation like Java
console.log(letters[1]); //=> "b"
console.log(things[3][2]); //=> 8
```

```
//assign using [] notation like Java
letters[0] = 'z';
console.log(letters); //=> ['z', 'b', 'c']
```

```
//assigning out of bounds automatically grows the array
letters[10] = 'g';
console.log(letters);
 //=> ['z', 'b', 'c', , , , , , , 'g']
console.log(letters.length); //=> 11
```

# JavaScript arrays

- Arrays have their own methods

```
//Make a new array
var array = ['i','n','f','x'];

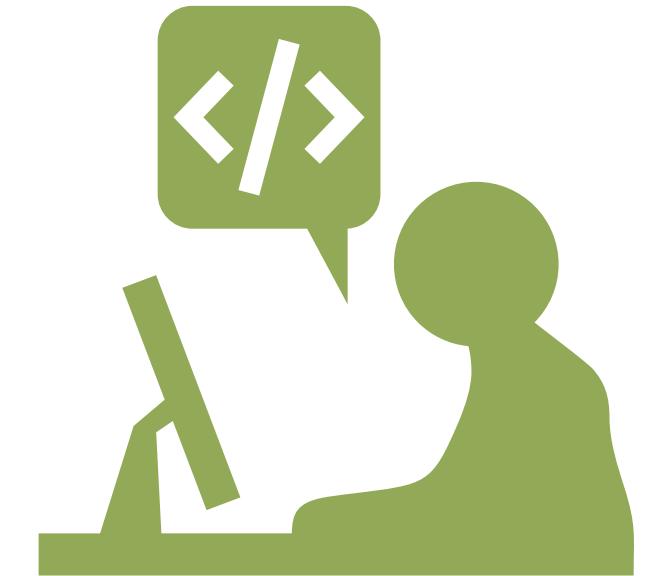
//add item to end of the array
array.push('133');
console.log(array); //=> ['i','n','f','x','133']

//combine elements into a string
var str = array.join('-');
console.log(str); //=> "i-n-f-x-133"

//get index of an element (first occurrence)
var oIndex = array.indexOf('x'); //=> 3

//remove 1 element starting at oIndex
array.splice(oIndex, 1);
console.log(array); //=> ['i','n','f','133']
```

# Array methods



A screenshot of a browser's developer tools Console tab. The tab bar shows 'Console' is active. The console output displays the following text:

```
Array is 6,2,8,1,2,5,3
Array is now 6,2,8,1,2,5,3,4
Sum is 31
There are 5 even numbers in the array.
```

The right side of each line shows the file name and line number where the output was generated:

- Array is 6,2,8,1,2,5,3 setup2.js:2
- Array is now 6,2,8,1,2,5,3,4 setup2.js:11
- Sum is 31 setup2.js:19
- There are 5 even numbers in the array. setup2.js:27

# Question



What will be shown in the console?

- A number\*fish\*2\*-1\*dog\*0
- B undefined\*fish\*2\*undefined\*dog\*2
- C string\*fish\*2\*24\*dog\*2
- D undefined\*fish\*2\*undefined\*dog\*2
- E number\*fish\*2\*blue\*3\*dog\*2

```
var array = ['1', 'fish', 2, 'blue'];
array[5] = 'dog';
array.push('2');
array[2] = array[array.length - 1] - 4;
array[0] = typeof array[2];
array[4] = array.indexOf('blue');

console.log(array.join('*'));
```

# Question



What will be shown in the console?

- A number\*fish\*2\*-1\*dog\*0
- B undefined\*fish\*2\*undefined\*dog\*2
- C string\*fish\*2\*24\*dog\*2
- D undefined\*fish\*2\*undefined\*dog\*2
- E number\*fish\*2\*blue\*3\*dog\*2

```
var array = ['1', 'fish', 2, 'blue'];
array[5] = 'dog';
array.push('2');
array[2] = array[array.length - 1] - 4;
array[0] = typeof array[2];
array[4] = array.indexOf('blue');

console.log(array.join('*'));
```

# JavaScript objects

- An unordered set of key and value pairs

- Like a HashMap in Java or a dictionary in Python

- Sometimes called *associative arrays*

Quotes around keys are optional



```
ages = {alice:40, bob:35, charles:13}
extensions = {'daniel':1622, 'in4matx':9937}
num_words = {1:'one', 2:'two', 3:'three'}
things = {num:12, dog:'woof', list:[1,2,3]}
empty = {}
empty = new Object(); //empty object
```

# JavaScript Object Notation (JSON)

```
{
 "first_name": "Alice",
 "last_name": "Smith",
 "age": 40,
 "pets": ["rover", "fluffy", "mittens"],
 "favorites": {
 "music": "jazz",
 "food": "pizza",
 "numbers": [12, 42]
 }
}
```

- Used in many APIs to send/receive data

# Accessing properties

- Values (or properties) can be referenced with the array[] syntax

```
ages = {alice:40, bob:35, charles:13}
```

```
//access ("look up") values
console.log(ages['alice']); //=> 40
console.log(ages['bob']); //=> 35
console.log(ages['charles']); //=> 13
```

```
//keys not in the object have undefined values
console.log(ages['fred']); //=> undefined
```

```
//assign values
ages['alice'] = 41;
console.log(ages['alice']); //=> 41
```

```
ages['fred'] = 19; //adds the key and assigns
 //a value to it
```

# Accessing properties

- Values can also be referenced with dot notation

```
var person = {
 firstName: 'Alice',
 lastName: 'Smith',
 favorites: {
 food: 'pizza',
 numbers: [12, 42]
 }
};
```

```
var name = person.firstName; //get value of 'firstName' key
person.lastName = 'Jones'; //set value of 'lastName' key
console.log(person.firstName+ ' '+person.lastName); //"Alice Jones"
```

```
var topic = 'food'
var favFood = person.favorites.food; //object in the object
 //object //value
```

```
var firstNumber = person.favorites.numbers[0]; //12
person.favorites.numbers.push(7); //push 7 onto the Array
```

# Functions

- Functions in JavaScript are like static methods in Java

```
//Java
public static String sayHello(String name){
 return "Hello, "+name;
}
public static void main(String[] args){
 String msg = sayHello("IN4MATX 133");
}
```

Parameters have no type

```
//JavaScript
function sayHello(name){
 return "Hello, "+name;
}
No access modifier
var msg = sayHello("IN4MATX 133");
```

Parameters are comma-separated

# Functions

- In Javascript, all parameters are optional

```
function sayHello(name)
{
 return "Hello, "+name;
}

//expected; parameter is assigned a value
sayHello("IN4MATX 133"); //Hello, IN4MATX 133

//parameter not assigned value (left undefined)
sayHello(); //Hello, undefined

//extra parameters (values) are not assigned
//to variables, so are ignored
sayHello("IN4MATX","133"); //Hello, IN4MATX"
```

**We'll make things more confusing  
next class...**

# **IN4MATX 133: User Interface Software**

**Lecture 4:**  
**Bootstrap & Javascript 1**

Professor Daniel A. Epstein  
TA Lucas de Melo Silva  
TA Jong Ho Lee