IN4MATX 133: User Interface Software

Lecture 4: Responsive Design & Javascript 1

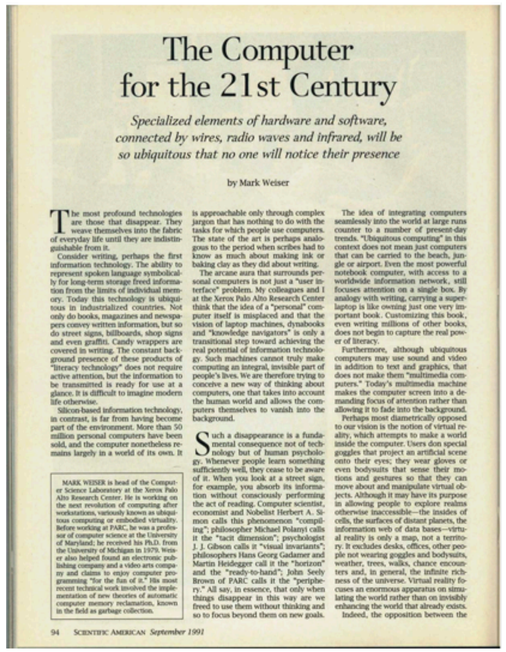
Professor Daniel A. Epstein
TA Lucas de Melo Silva
TA Jong Ho Lee

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
- Begin implementing responsive designs with Bootstrap
- Explain the role of JavaScript
- Implement fundamental programming concepts in JavaScript like variables, loops, and conditionals

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

The 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 symbolically for long-term storage freed information 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 constant background presence of these 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 otherwise.

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 nonetheless remains largely in a world of its own. It is approachable only through complex jargon that has nothing to do with the tasks for which people use computers. The state of the art is perhaps analogous to the period when scribes had to know as much about making ink or baking clay as they did about writing.

The arcane aura that surrounds personal computers is not just a "user interface" problem. My colleagues and I at the Xerox Palo Alto Research Center think that the idea of a "personal" computer itself is misplaced and that the vision of laptop machines, dynabooks and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computing an integral, invisible part of people's lives. We are therefore trying to conceive a new way of thinking about computers, one that takes into account the human world and allows the computers themselves to vanish into the background.

Such a disappearance is a fundamental consequence not of technology but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading. Computer scientist, economist and Nobelist Herbert A. Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist J. J. Gibson calls it "visual invariants"; philosophers Hans Georg Gadamer and Martin Heidegger call it the "horizon" and the "ready-to-hand"; John Seely Brown of PARC calls it the "periphery." All say, in essence, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals.

The idea of integrating computers seamlessly into the world at large runs counter to a number of present-day trends. "Ubiquitous computing" in this context does not mean just computers that can be carried to the beach, jungle or airport. Even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box. By analogy with writing, carrying a superlaptop is like owning just one very important book. Customizing this book, even writing millions of other books, does not begin to capture the real power of literacy.

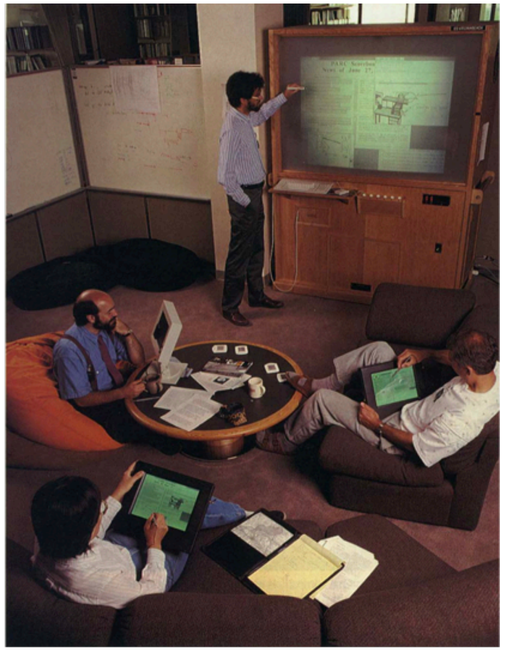
Furthermore, although ubiquitous computers may use sound and video in addition to text and graphics, that does not make them "multimedia computers." Today's multimedia machine makes the computer screen into a demanding focus of attention rather than allowing it to fade into the background.

Perhaps most diametrically opposed to our vision is the notion of virtual reality, which attempts to make a world inside the computer. Users don special goggles that project an artificial scene onto their eyes; they wear gloves or even bodysuits that sense their motions and gestures so that they can move about and manipulate virtual objects. Although it may have its purpose in allowing people to explore realms otherwise inaccessible—the insides of cells, the surfaces of distant planets, the information web of data bases—virtual reality is only a map, not a territory. It excludes desks, offices, other people not wearing goggles and bodysuits, weather, trees, walks, chance encounters and, in general, the infinite richness of the universe. Virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists.

Indeed, the opposition between the

MARK WEISER is head of the Computer Science Laboratory at the Xerox Palo Alto Research Center. He is working on the next revolution of computing after workstations, variously known as ubiquitous computing or embodied virtuality. Before working at PARC, he was a professor of computer science at the University of Maryland; he received his Ph.D. from the University of Michigan in 1979. Weiser also helped found an electronic publishing company and a video arts company and claims to enjoy computer programming "for the fun of it." His most recent technical work involved the implementation of new theories of automatic computer memory reclamation, known in the field as garbage collection.

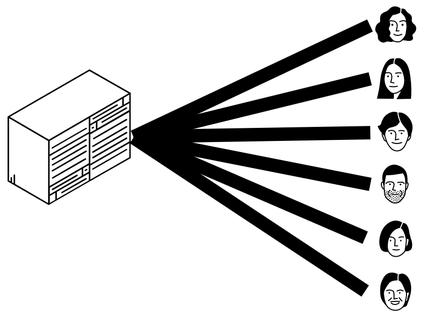
94 SCIENTIFIC AMERICAN *September 1991*



Three waves of computing

1

Mainframe computing



“Many to one”

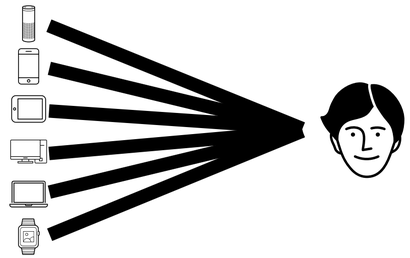
2

Personal computing

“One to one”

3

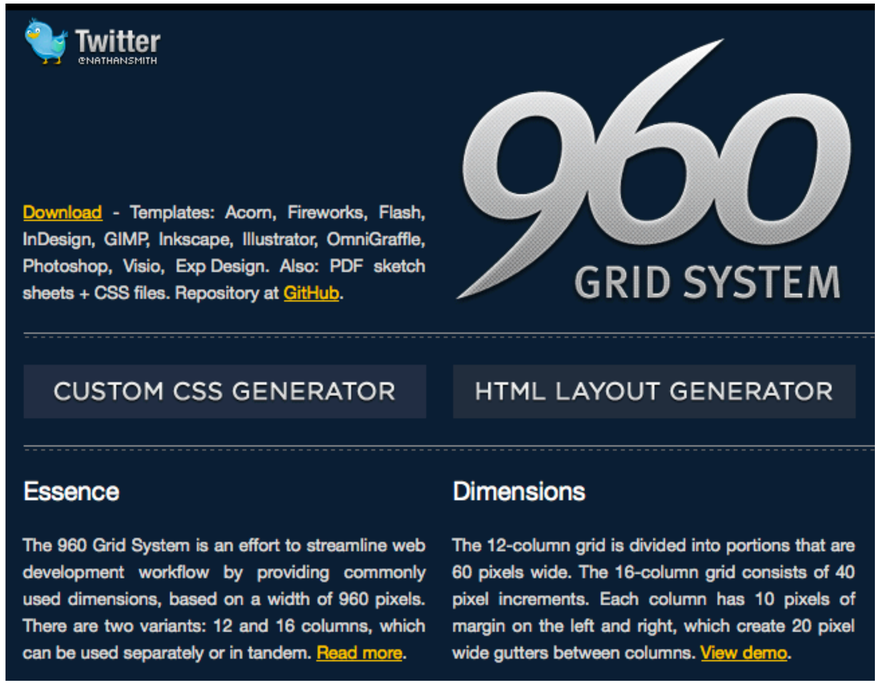
Ubiquitous computing



“One to many”

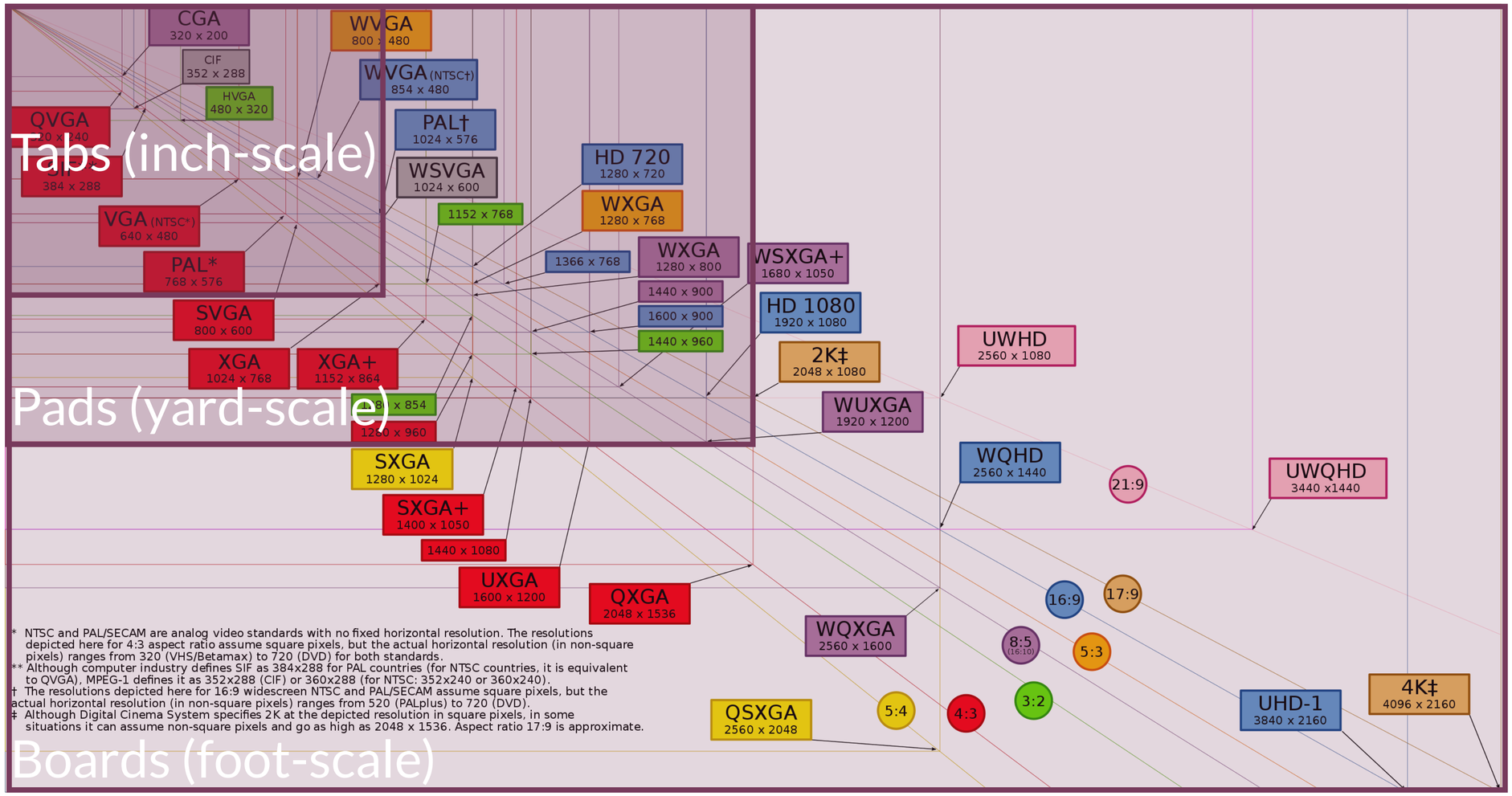
Websites in the personal computing era

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



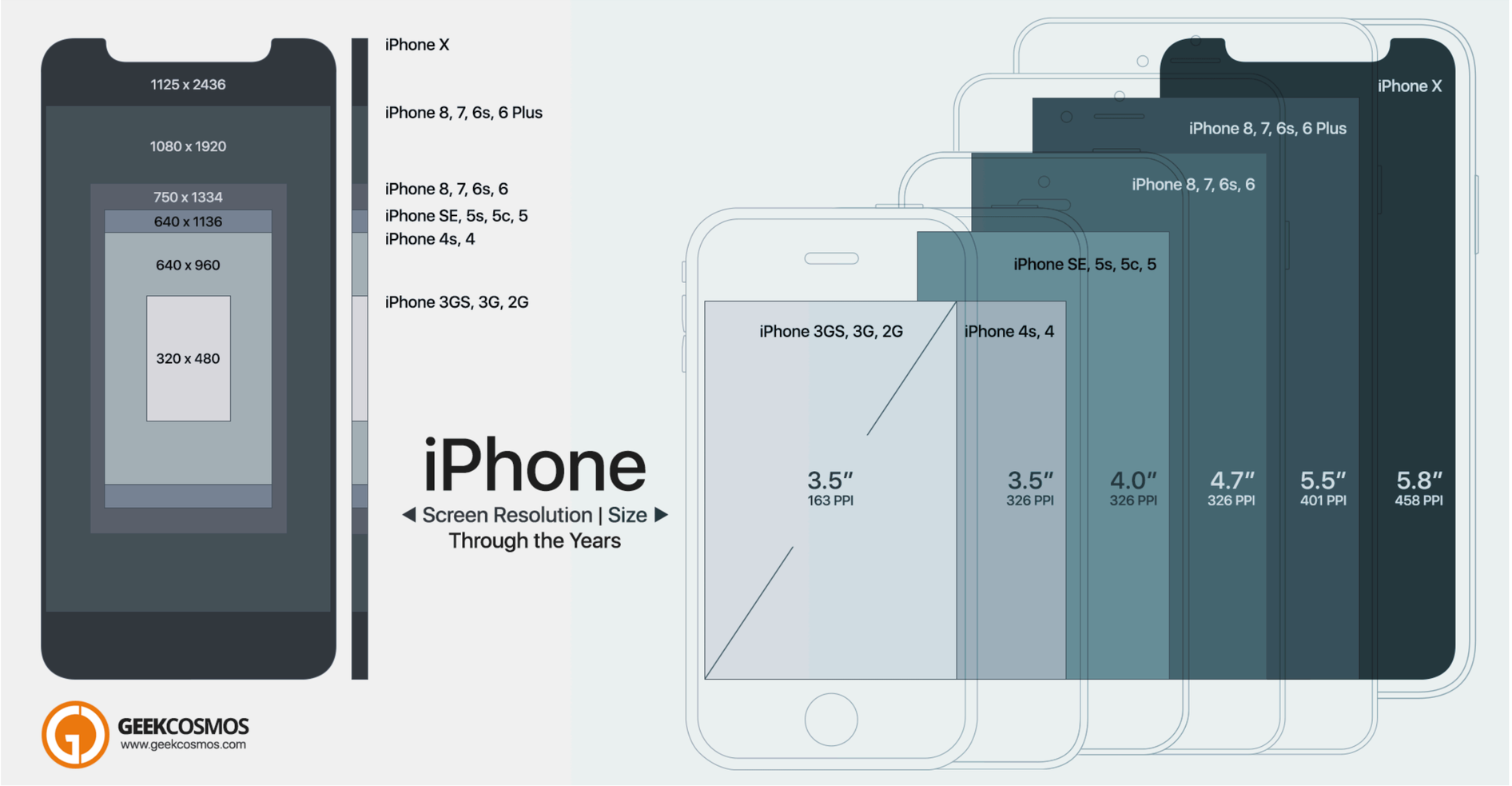
https://960.gs/

Websites today: ubiquitous computing



https://en.wikipedia.org/wiki/Display_resolution

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

https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/

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

https://css-tricks.com/the-difference-between-responsive-and-adaptive-design/

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

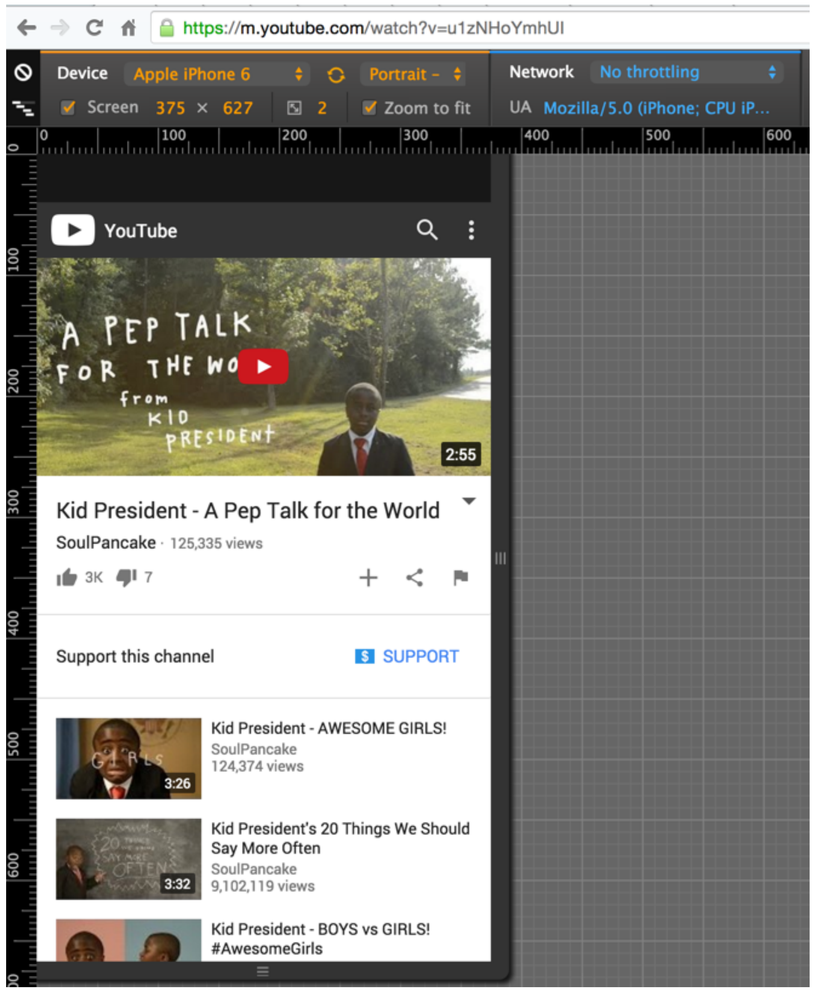
- Much more difficult to maintain separate codebases

- 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

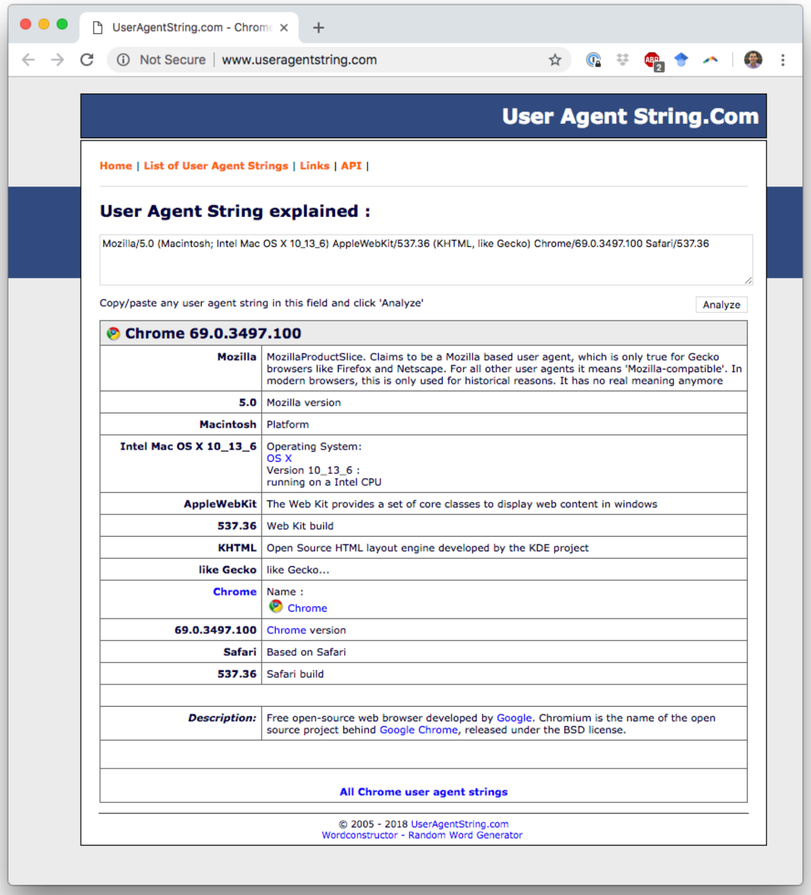
Adaptive design

- 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

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

Moving away from adaptive design, 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

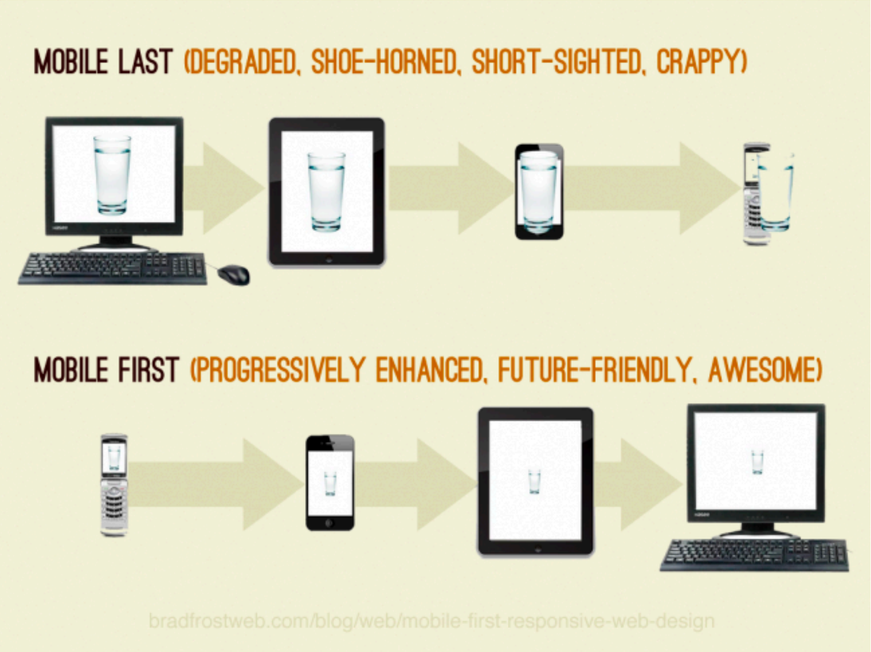
```
@media screen and (max-width: 640px) {
  /* small screens */
}

@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`

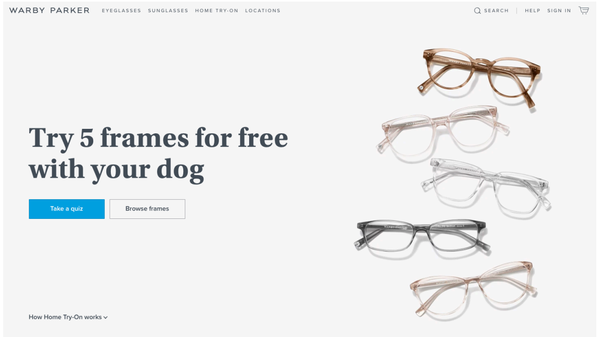
Mobile-first design

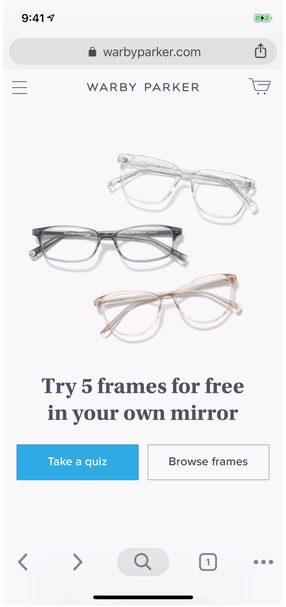
- “Graceful degradation” vs. “progressive enhancement”
- Plan your design for mobile
- Then make your app *better* with more real estate
  - Add more features
  - Make existing features easier to navigate



A few tips for mobile design

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

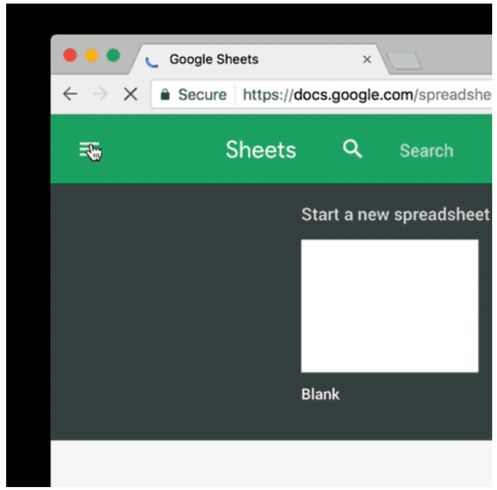


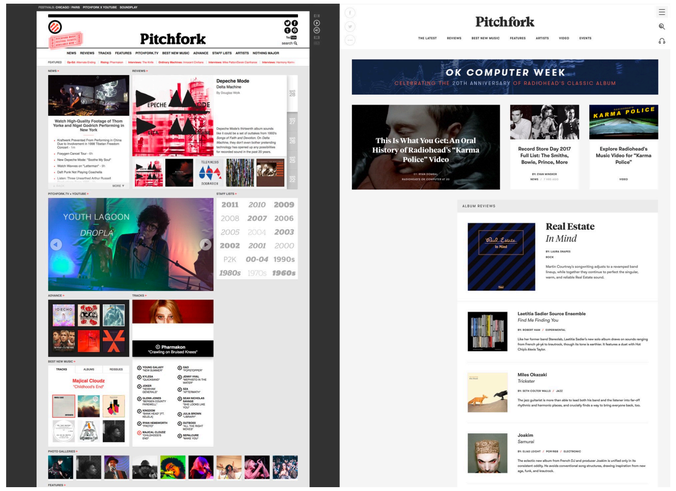


https://www.bluefountainmedia.com/blog/desktop-vs-mobile-three-key-website-design-differences

Mobile-first, not mobile-only

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





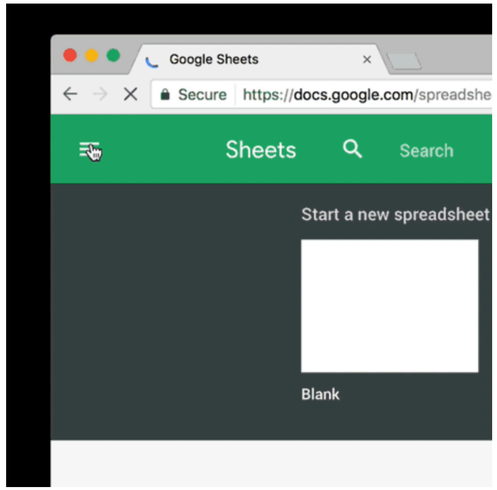
https://blog.prototypr.io/mobile-first-desktop-worst-f900909ae9e2

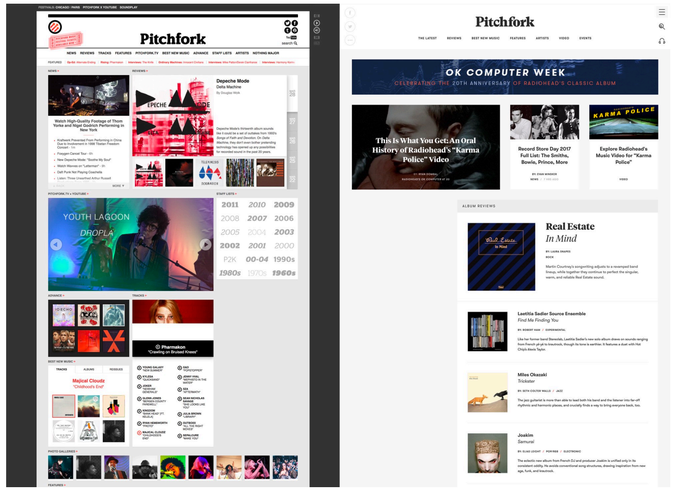
Mobile-first, not mobile-only

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

Mobile-first, not mobile-only

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





https://blog.prototypr.io/mobile-first-desktop-worst-f900909ae9e2

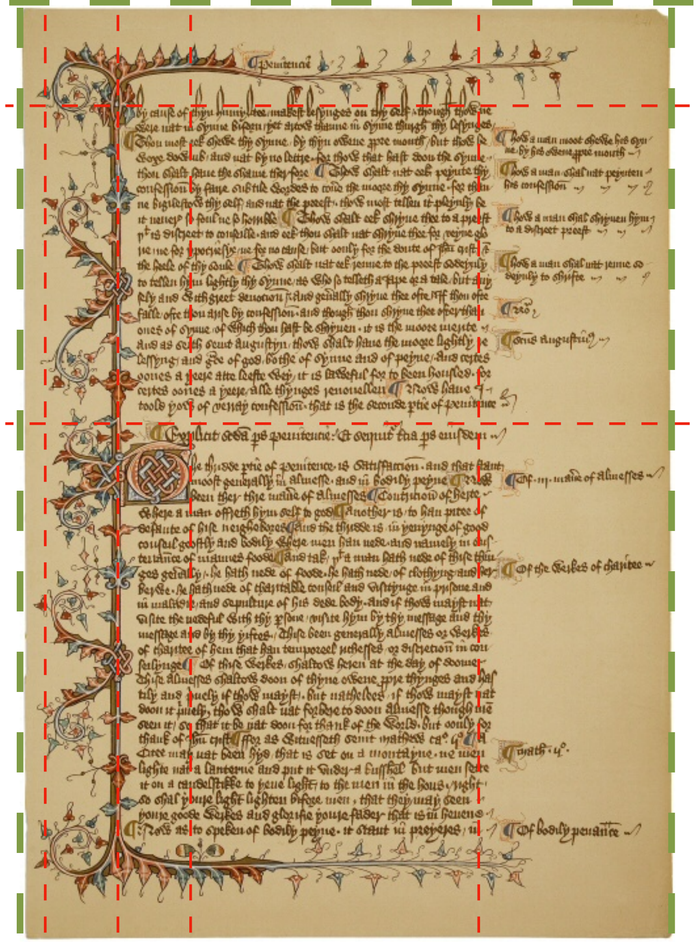
Mobile-first, not mobile-only

- 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

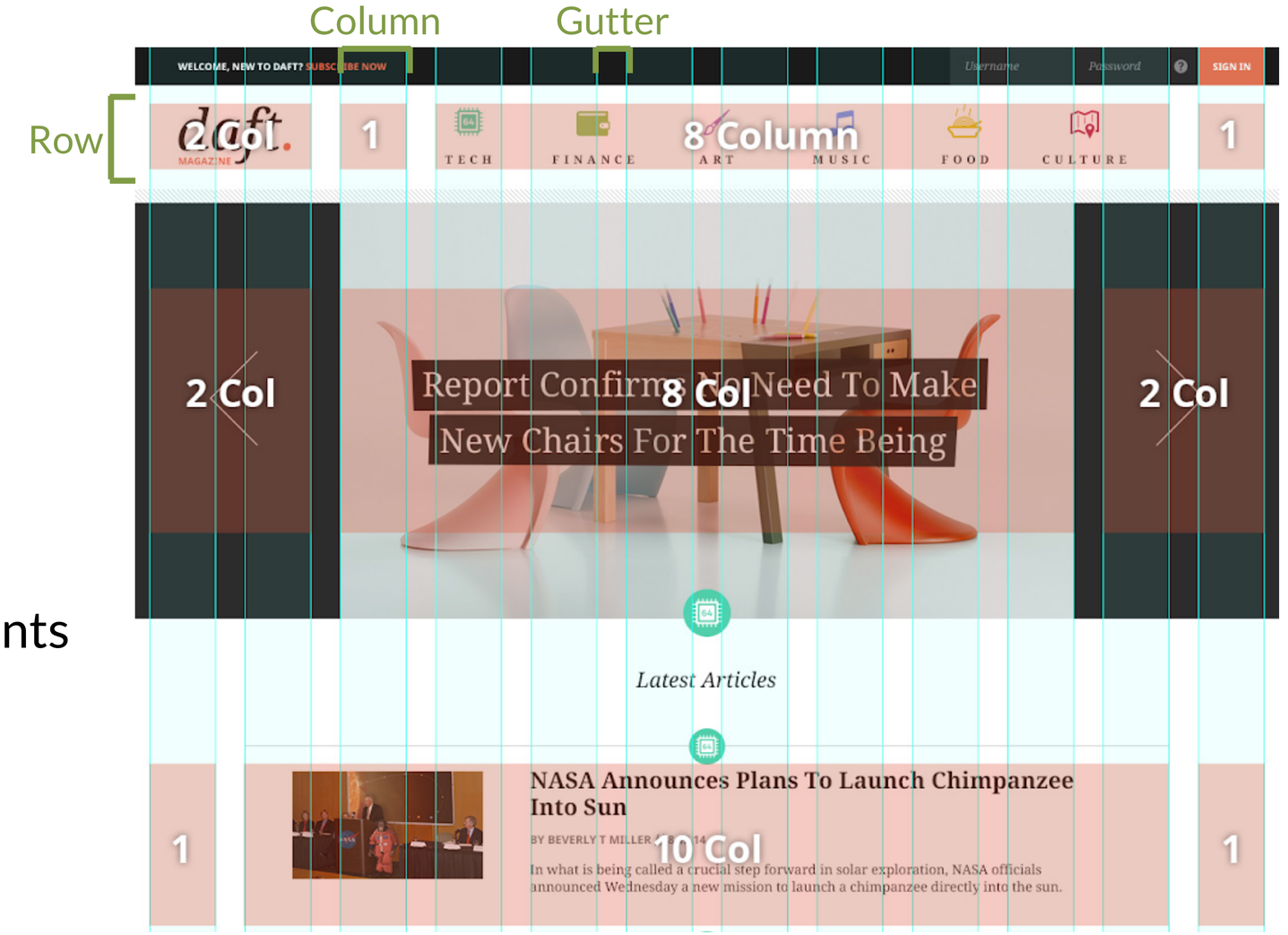
- Established tool for content arrangement
- Gridded content is familiar and easy to follow
- In general, it's good to target fewer lines
  - But breaking that rule is important for creativity and attention-grabbing



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

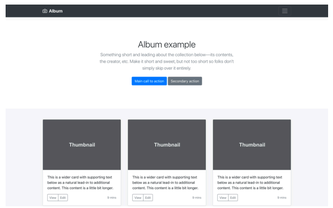
Grid-based layouts

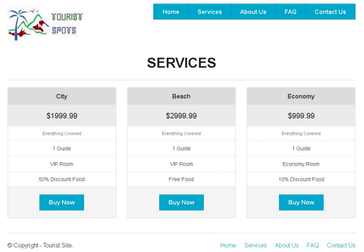
- Rows
- Columns
- Gutters
- Padding/spacing
  - Defined by specific elements



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

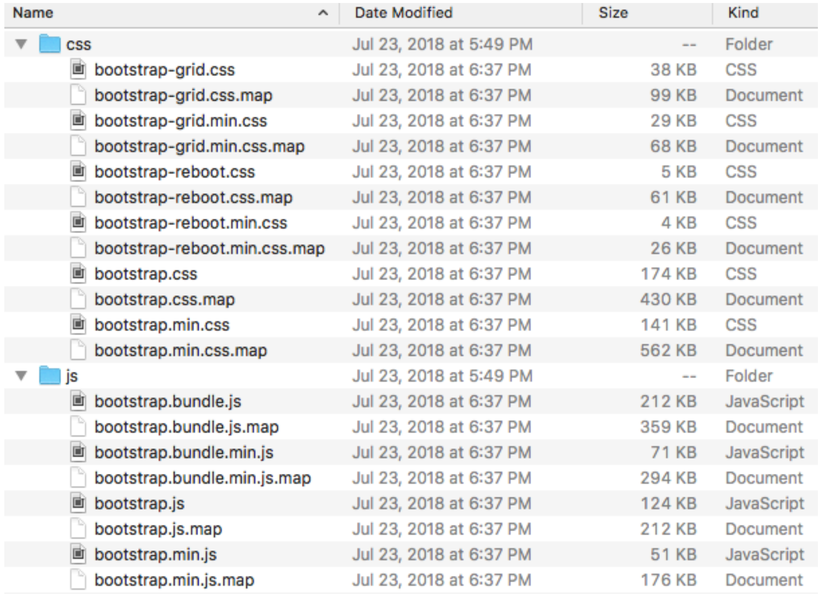




Digging into Bootstrap

Bootstrap

- 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]`

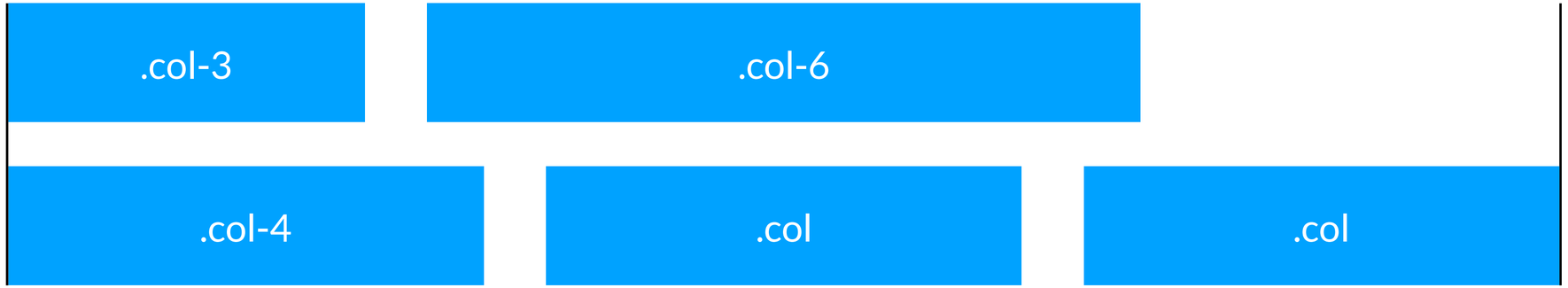
	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`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

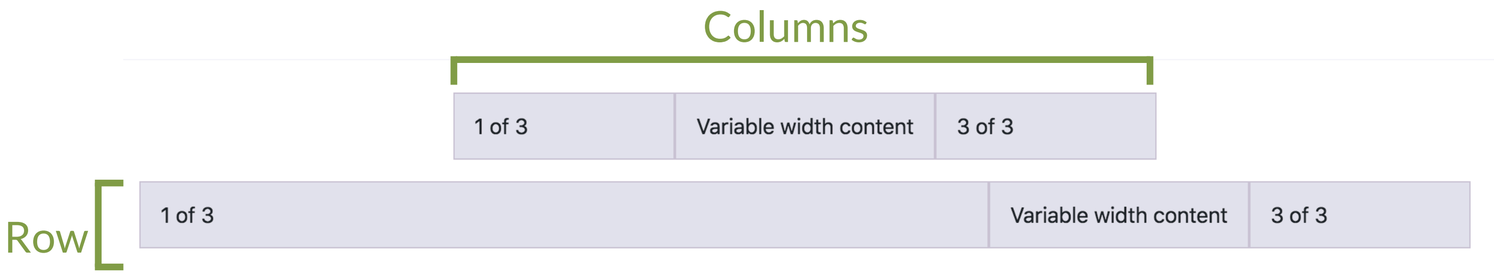


Bootstrap

Grid System

- Rows are block elements, while columns are inline

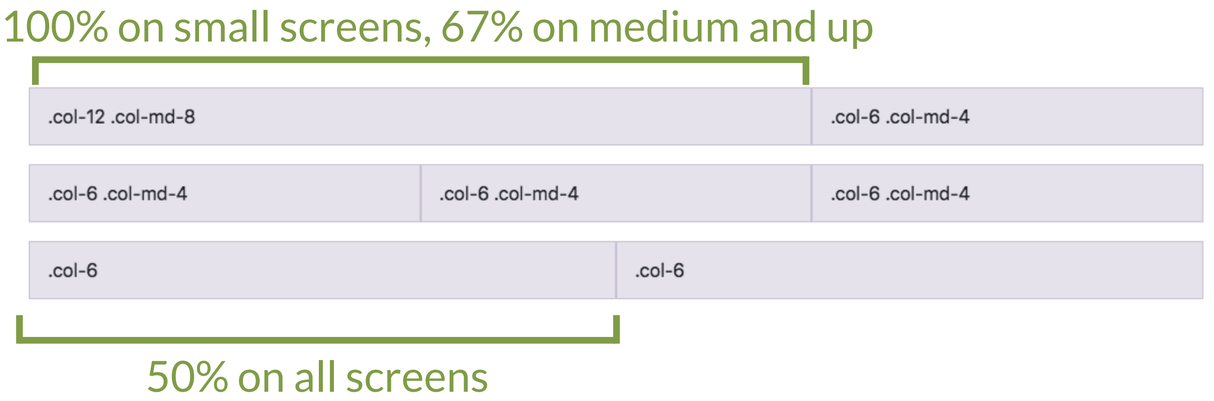
https://getbootstrap.com/docs/4.1/layout/grid/



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

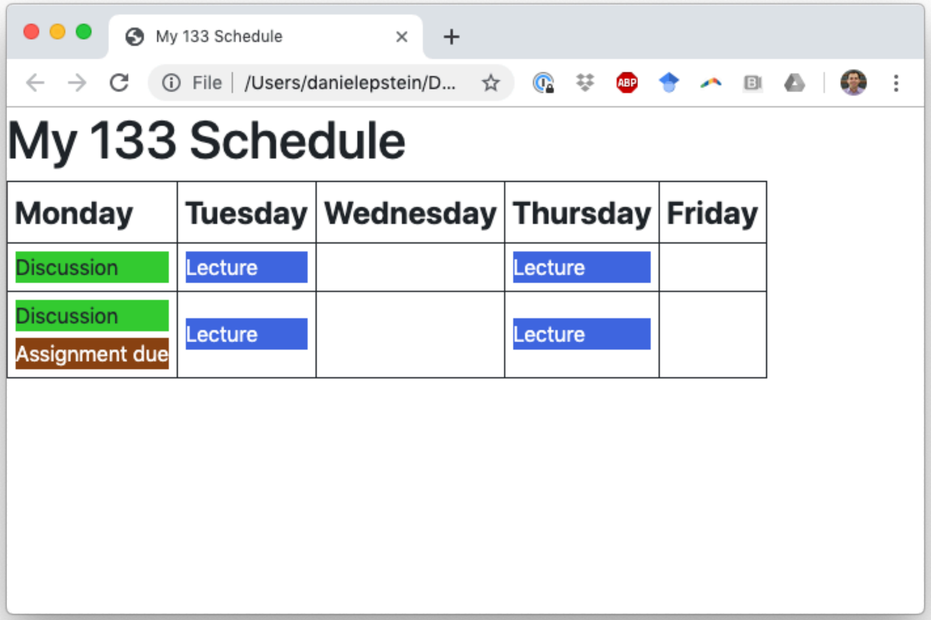


Bootstrap

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

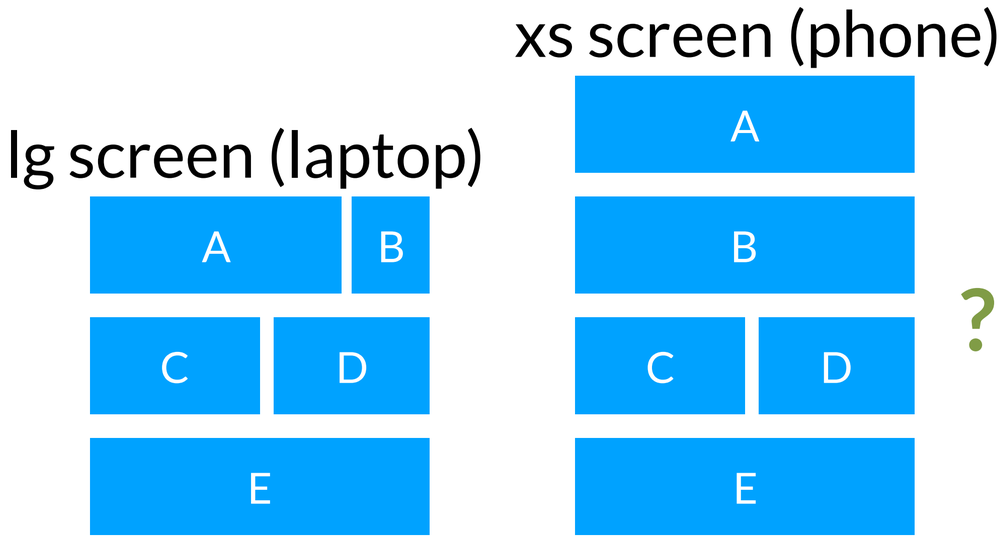


My 133 Schedule

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

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

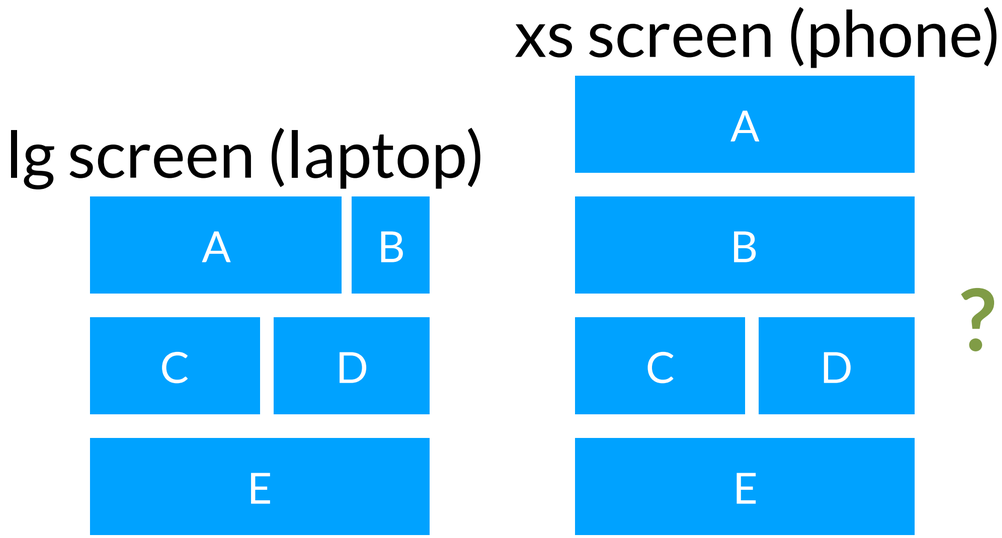
```
<div class="row">
   <div class="col-8">A</div>
   <div class="col-4">B</div>
   <div class="col-xs-6">C</div>
   <div class="col-xs-6">D</div>
</div>
<div class="row">
   <div class="col-md-6 col-xl-4">E</div>
</div>
```

D A and B

E B and C

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.

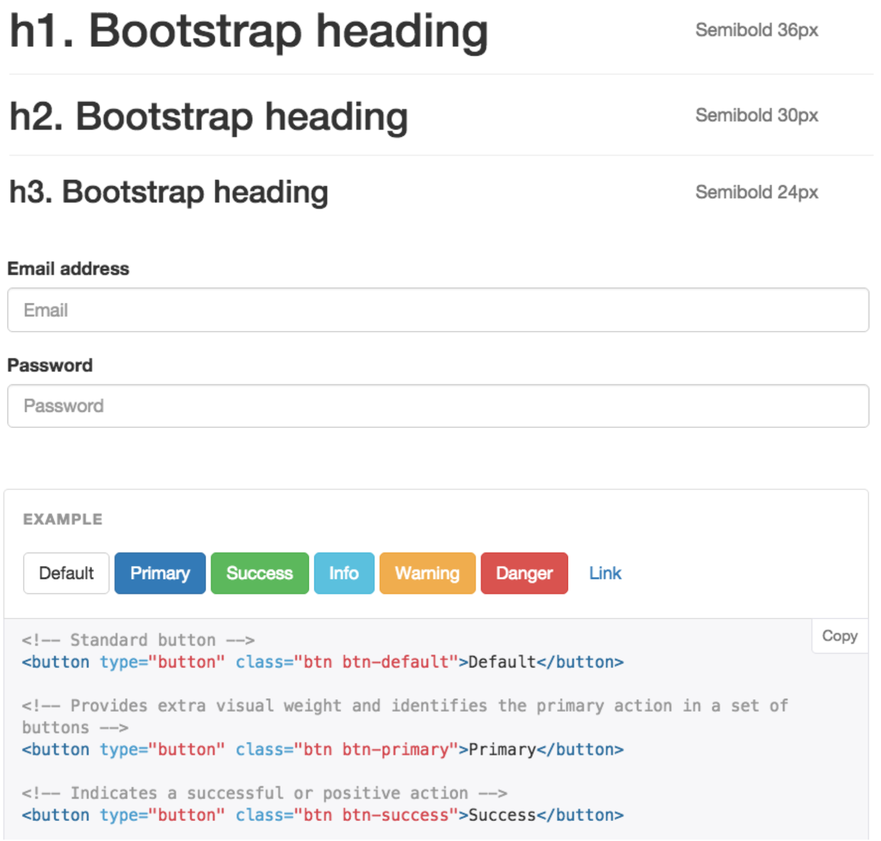
	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
`.visible-xs-*`	Visible	Hidden	Hidden	Hidden
`.visible-sm-*`	Hidden	Visible	Hidden	Hidden
`.visible-md-*`	Hidden	Hidden	Visible	Hidden
`.visible-lg-*`	Hidden	Hidden	Hidden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

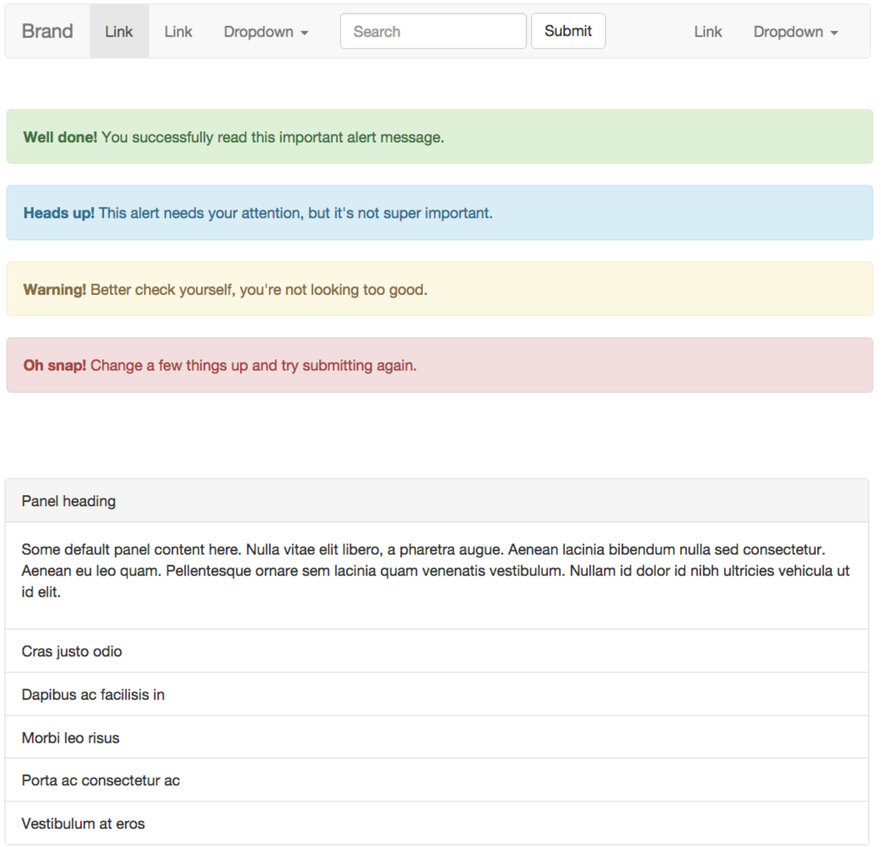


http://getbootstrap.com/css

Bootstrap

Components

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

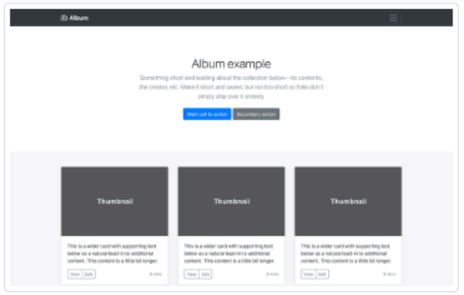


http://getbootstrap.com/css

Grid frameworks
make development easier.
What are the downsides?

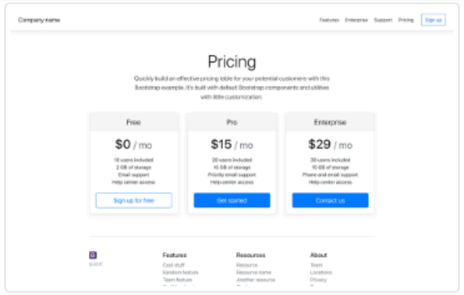
Opposition to Grid-based frameworks

Can lead to similar-looking webpages



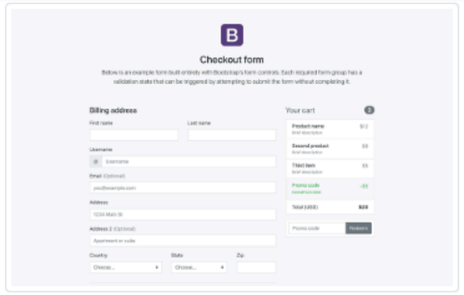
Album

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



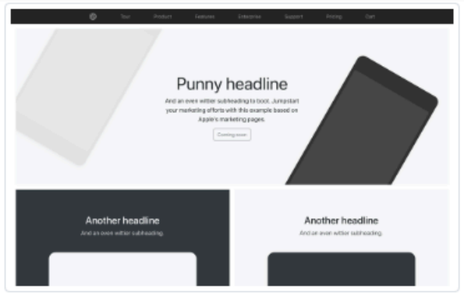
Pricing

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



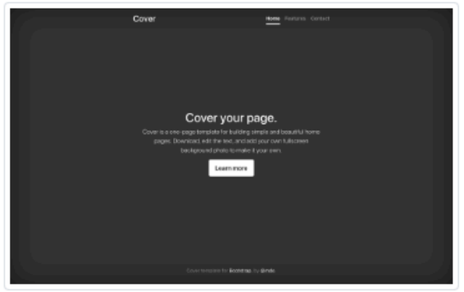
Checkout

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



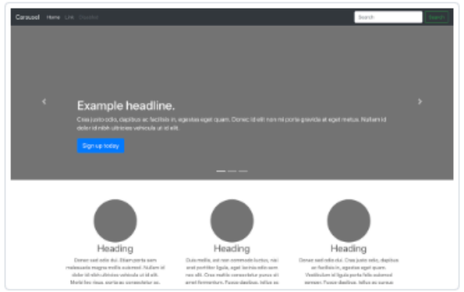
Product

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



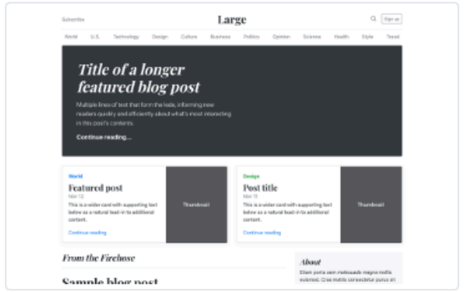
Cover

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



Carousel

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



Blog

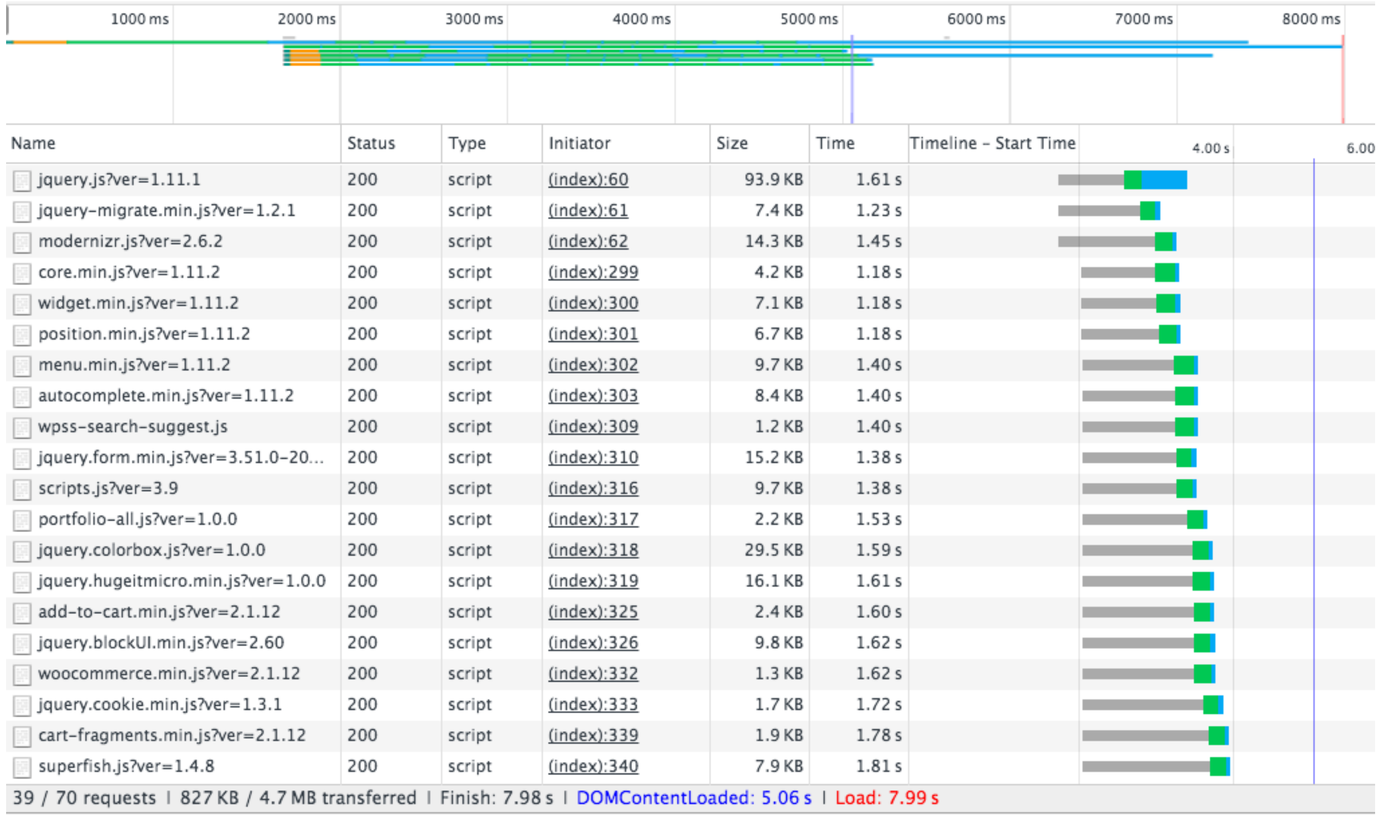
Magazine like blog template with header, navigation, featured content.

Dashboard

Basic admin dashboard shell with fixed sidebar and navbar.

Opposition to Grid-based frameworks

Can involve loading many files, hurting performance



Opposition to Grid-based frameworks

Can stifle creativity

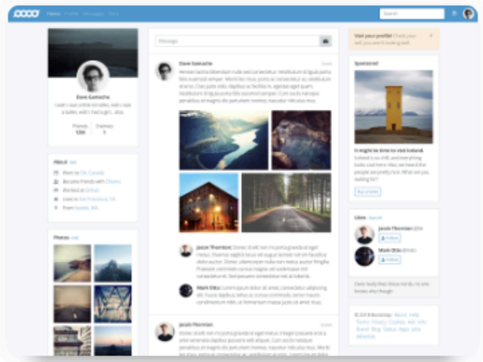
Themes built by or reviewed by Bootstrap's creators.

Why our themes?

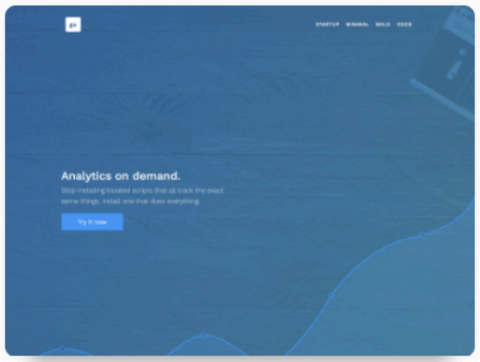
Built by Bootstrap Team

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

Dashboard $49.00
Admin & Dashboard ★★★★★



Application $49.00
Application ★★★★★



Marketing $49.00
Landing & Corporate ★★★★★

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

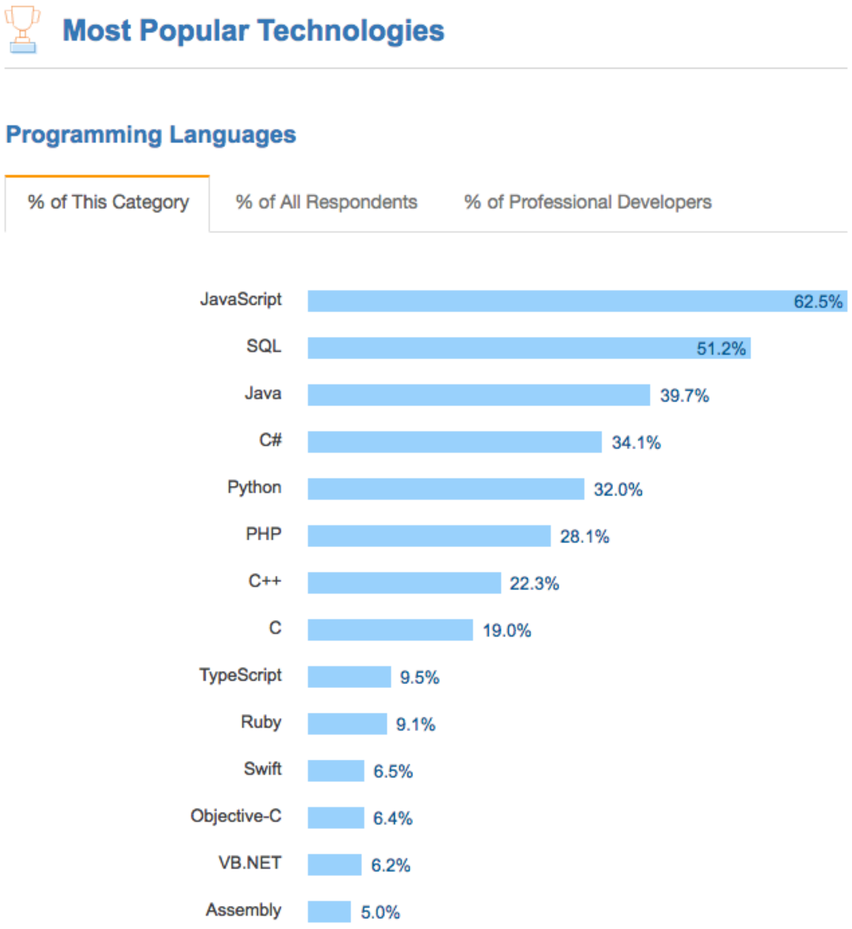
Other web programming languages

- 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



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”

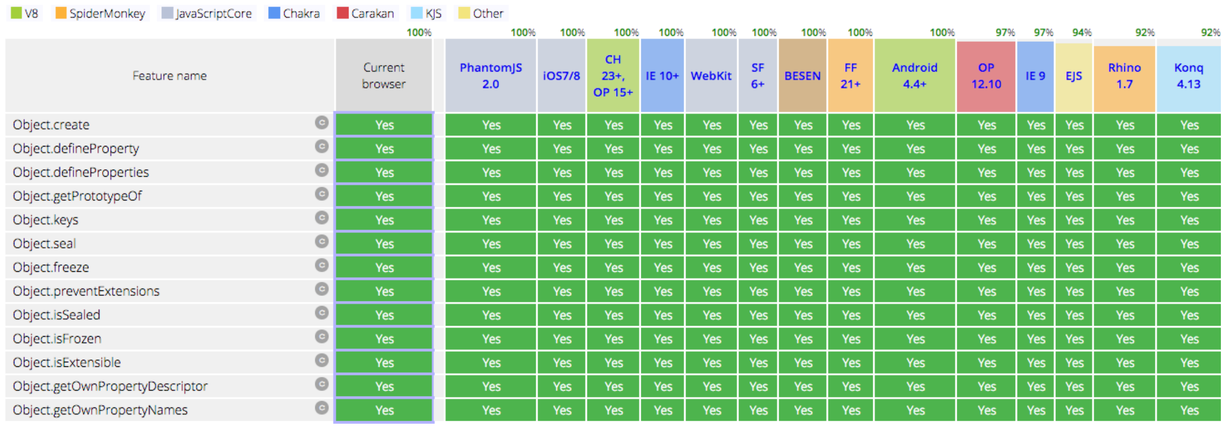
European Computer Manufacturers Association

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

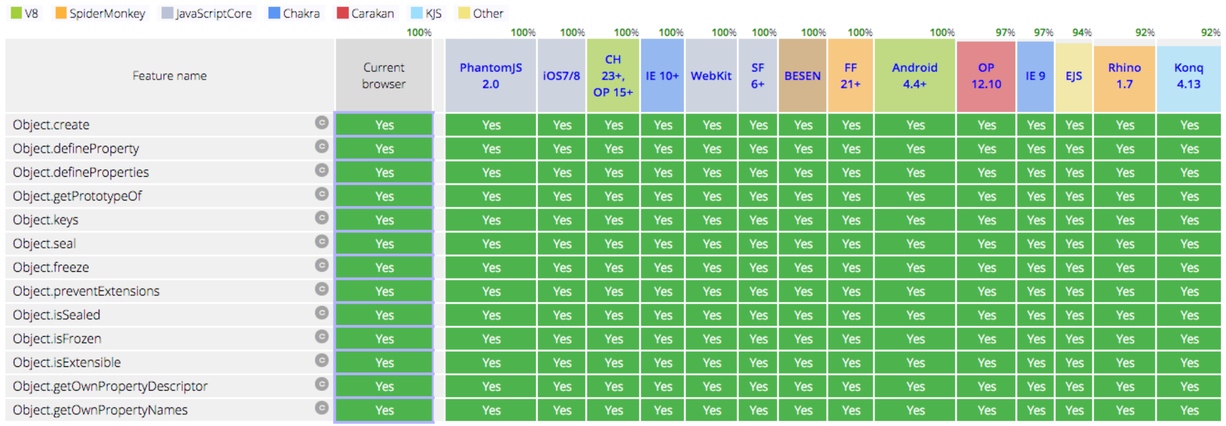


V8 SpiderMonkey JavaScriptCore Chakra Carakan KJS Other

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
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	97%	97%	94%	92%	92%
Object.create	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	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)



V8 SpiderMonkey JavaScriptCore Chakra Carakan KJS Other

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
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	97%	97%	94%	92%	92%
Object.create	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	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

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

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

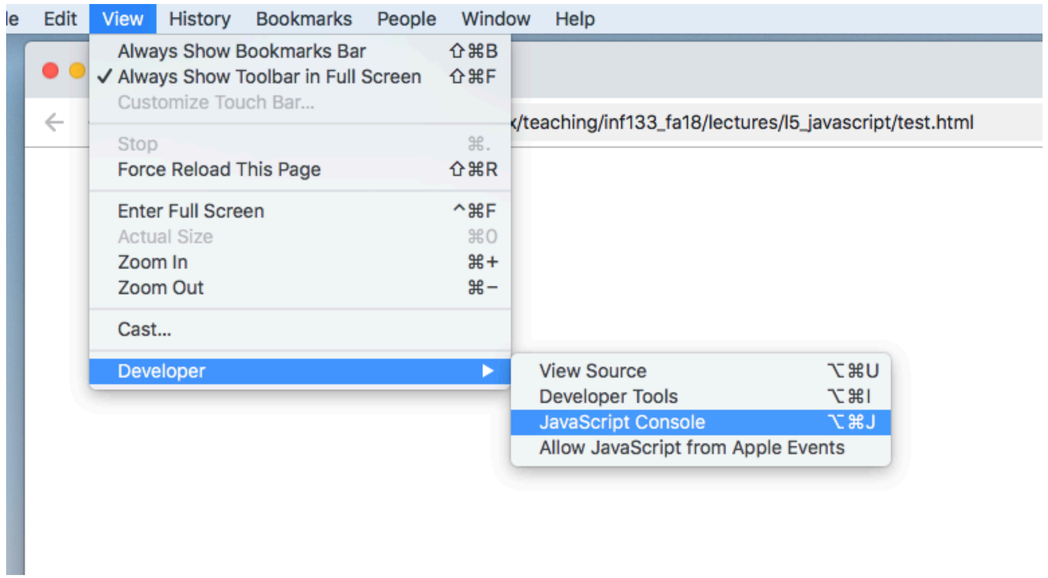
Loading JavaScript

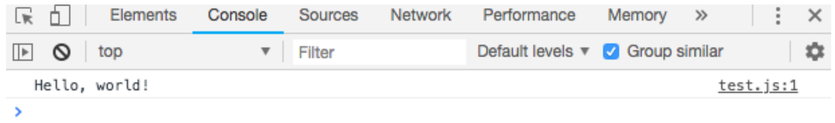
```
<html>
  <head>
    <script src="test.js"></script>
  </head>
</html>
```

Printing in JavaScript

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

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





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

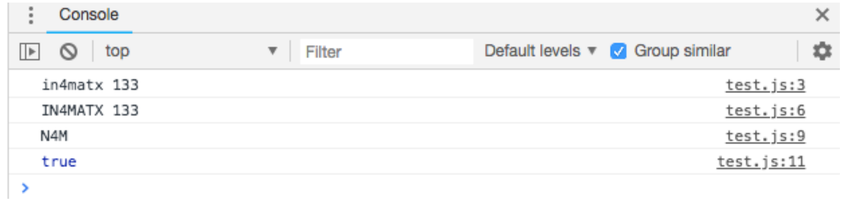
```
var num = 10;
var str = '10';

//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

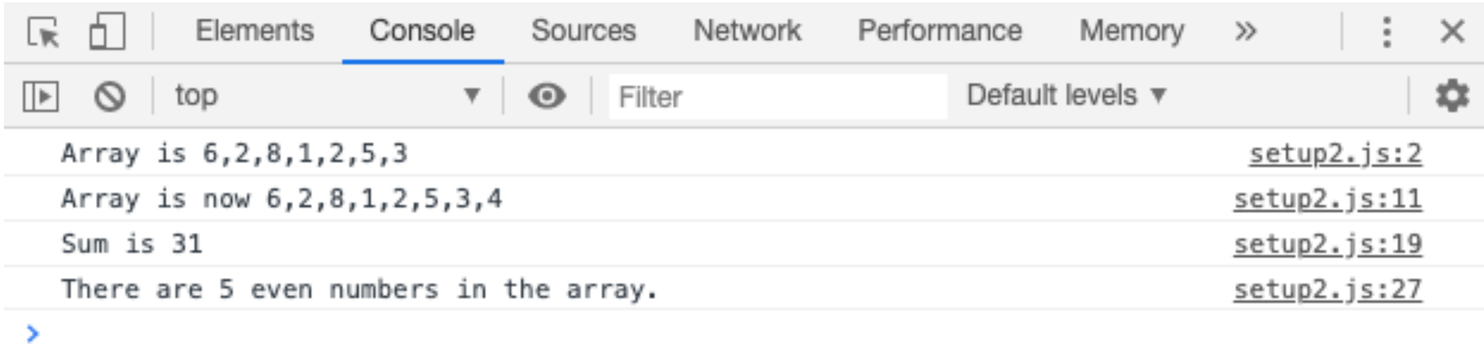
```
var letters = ['a', 'b', 'c'];
var numbers = [1, 2, 3];
var things = ['raindrops', 2.5, true, [5, 9, 8]]; //arrays can be nested
var empty = [];
var blank5 = new Array(5); //empty array with 5 items

//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



Question 

What will be shown in the console?

```
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('*'));
```

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

Question 

What will be shown in the console?

```
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('*'));
```

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`

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;
}
var msg = sayHello("IN4MATX 133");
```

Parameters are comma-separated

No access modifier or return type

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...

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
- Begin implementing responsive designs with Bootstrap
- Explain the role of JavaScript
- Implement fundamental programming concepts in JavaScript like variables, loops, and conditionals

IN4MATX 133: User Interface Software

Lecture 4: Responsive Design & Javascript 1

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