Course Structure

	Ideal	Reality
1	Reg Office: Cmd Line	Reg Office: Cmd Line
2	Reg Office: Desktop v1	Reg Office: Desktop v1
3	Reg Office: Desktop v2	Reg Office: Web v1
4	Reg Office: Web v1	Reg Office: Web v2
5	Reg Office: Web v2	Reg Office: Desktop v2

Client-Side Web Programming: CSS

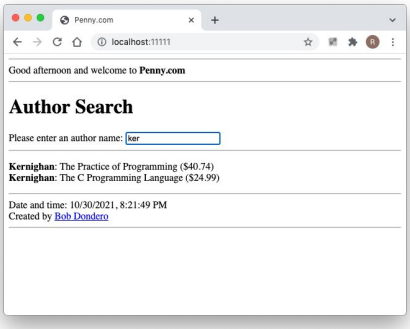
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - Cascading style sheets (CSS)
  - The smartphone viewport
  - Responsive web apps
  - CSS frameworks

Motivation

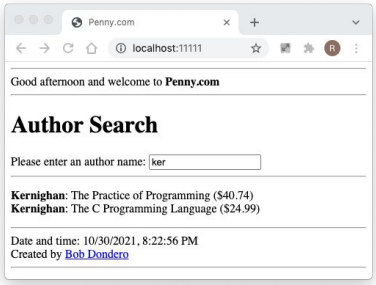
- Demos of PennyJQuery
  - Laptop computer (large browser window)



OK

Motivation

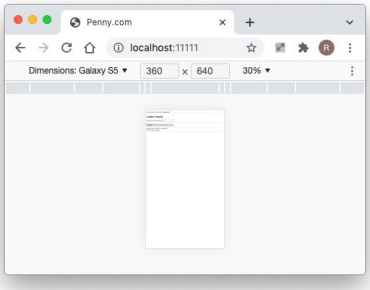
- Demos of PennyJQuery
  - Laptop computer (small browser window)



OK

Motivation

- Demos of PennyJQuery
  - Smartphone (very small browser window)



Chrome: More tools -> Developer Tools -> Toggle Device Toolbar

Unacceptable

Agenda

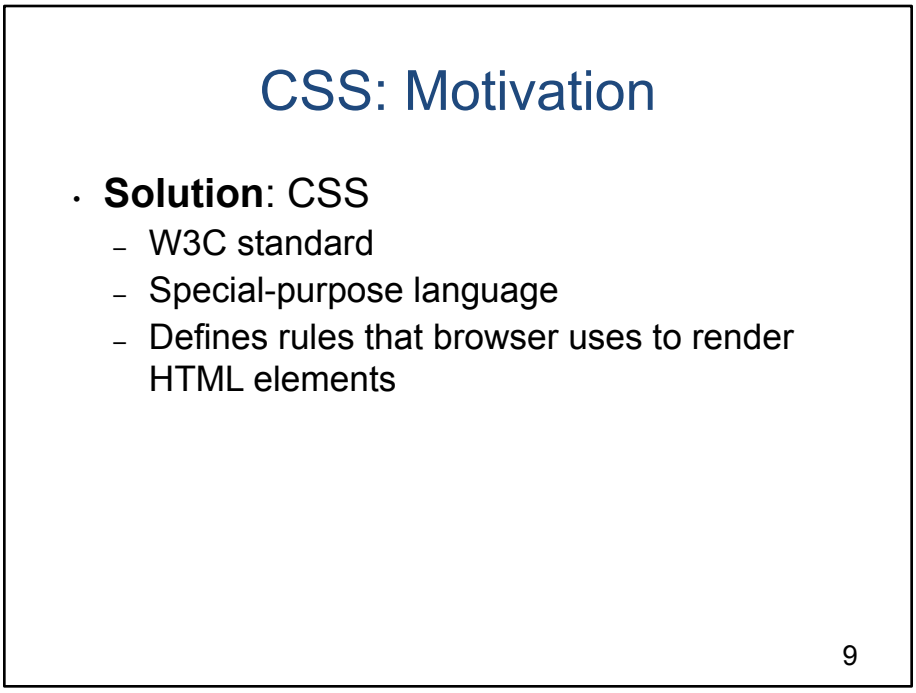
- **CSS**
- The smartphone viewport
- Responsive web apps
- CSS frameworks

CSS: Motivation

- **Problem**: Prior to CSS
  - Style information (font sizes, font colors, …) in HTML documents was repetitive, verbose, difficult to maintain

somefile.html

```
<html>
   …
   <body>
      <font color="red"><h1>First Heading</h1></font>
      …
      <font color="red"><h1>Second Heading</h1></font>
      …
   </body>
</html>
```



CSS: Motivation

Solution: CSS
- W3C standard
  - First version: 1994-1995
  - Current version: 2018
- Special-purpose language
- Defines *rules* that browser uses to render HTML elements

CSS: Fundamentals

- Can define ***external rules***: rules in an external style sheet

mystyle.css

```
h1 {color:red;}
```

External rules are easy to share across HTML documents

somefile.html

```
<html>
   <head>
      <link rel="stylesheet"
          href="mystyle.css">
      …
   </head>
   <body>
      <h1>First Heading</h1>
      …
      <h1>Second Heading</h1>
      …
   </body>
</html>
```

CSS: Fundamentals

- Can define ***embedded rules***: rules in an embedded style sheet

Embedded rules are specific to one HTML doc

somefile.html

```
<html>
   <head>
      <style>
         h1 {color:red;}
      </style>
      …
   </head>
   <body>
      <h1>First Heading</h1>
      …
      <h1>Second Heading</h1>
      …
   </body>
</html>
```

CSS: Fundamentals

- Can define ***inline rules***

Try to avoid

somefile.html

```
<html>
   <body>
      <h1 style="color:red;">
         First Heading
      </h1>
      …
      <h1 style="color:red;">
         Second Heading
      </h1>
      …
   </body>
</html>
```

CSS: “Cascading”

- To style a HTML document, browser applies:
  - Rules created by **browser developer**
  - Rules created by **browser user**
  - CSS rules created by **HTML document author**
    - CSS external rules
    - CSS embedded rules
    - CSS inline rules
- Browser “cascades” through rules

13

CSS: “Cascading”

[see slide]

There is more to “cascading” that we have time to cover
A slide at the end of this deck provides a reference to a book by Nixon that provides more information

CSS: Rules

- CSS rule syntax:

```
selector
{
   property:value;
   property:value;
   ...
}
```

CSS: Selectors

```
selector
{
    property:value;
    property:value;
    ...
}
```

CSS: Selectors

From https://www.w3schools.com/cssref/css_selectors.asp

Selector	Example	Example Description
.class	.xxx	Select all elements with a `class` attribute whose value is `xxx`
#id	#xxx	Select the element with with an `id` attribute whose value is `xxx`
*	*	Select all elements
element	p	Select all <p> elements
element,*element*	div, p	Select all elements and all <p> elements
element element	div p	Select all <p> elements inside elements
element>*element*	div > p	Select all <p> elements where the parent is a element

CSS: Selectors

From https://www.w3schools.com/cssref/css_selectors.asp

Selector	Example	Example Description
`[attribute]`	`[type]`	Select all elements with a `type` attribute
`[attribute=value]`	`[type=text]`	Select all elements with a `type` attribute whose value is `text`
`selector::after`	`p::after`	Insert something **after** the content of each `<p>` element
`selector::before`	`p::before`	Insert something **before** the content of each `<p>` element

And **many** more (58 of them)

CSS: Properties

```
selector
{
    property:value;
    property:value;
    ...
}
```

CSS: Properties

From https://www.w3schools.com/cssref/default.asp

Property	Example	Example Description
font-family	font-family:Arial	Set the font family of text within the selected element(s) to Arial
font-size	font-size:20px	Set the font size of text within the selected element(s) to 20 pixels
color	color:white	Set the color of text within the selected element(s) to white
background-color	background-color:#295078	Set the background color of the selected element(s) to r:29, g:50, b:78
text-align	text-align:center	Set the horizontal alignment of text within the selected element(s) to center
border-style	border-style:solid	Set the style of the four borders of the selected element(s) to solid
border-color	border-color:gray	Set the color of the four borders of the selected element(s) to gray
border-width	border-width:1px	Set the width of the four borders of the selected element(s) to 1 pixel

And **many** more (over 200 of them)

CSS: Example

- See **PennyCss** app
  - runserver.py
  - book.py, database.py
  - **penny.py** (same as previous)
  - **static/penny.css**
  - **index.html**

20

CSS: Example

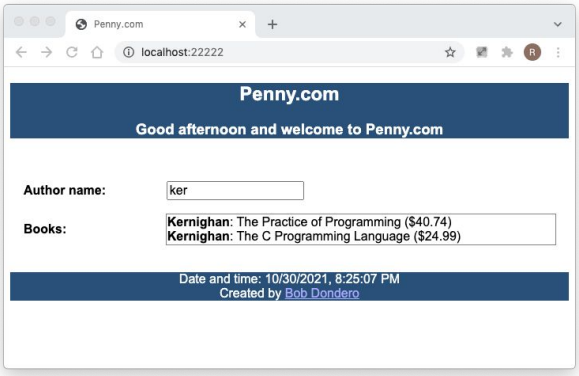
Code notes: penny.py
Same as in PennyJQuery

Code notes: static/penny.css
Defines element rules
body, input (type=text), a, table
Defines class rules
.header, .footer, .grayborder

Code notes: index.html
Uses container elements
span => no effect on rendering
Used to “mark spots” that will be referenced by JavaScript code
div => no effect on rendering, except new line
Used to cluster elements by style
Uses CSS

CSS: Example

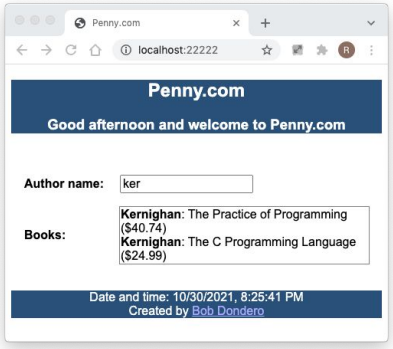
- Demos
  - Laptop computer (large browser window)



Good

CSS: Example

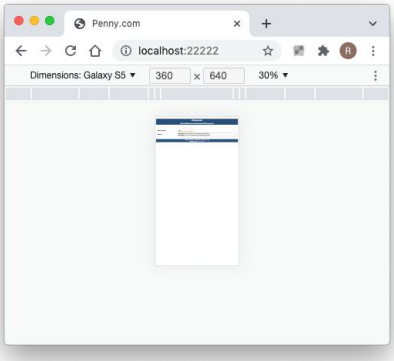
- Demos
  - Laptop computer (small browser window)



OK

CSS: Example

- Demos
  - Smartphone (very small browser window)



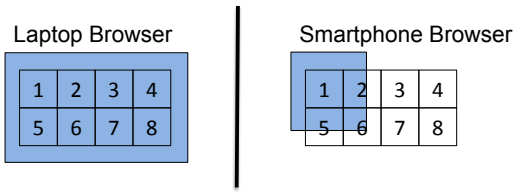
Unacceptable

Agenda

- CSS
- **The smartphone viewport**
- Responsive web apps
- CSS frameworks

Smartphone Viewport

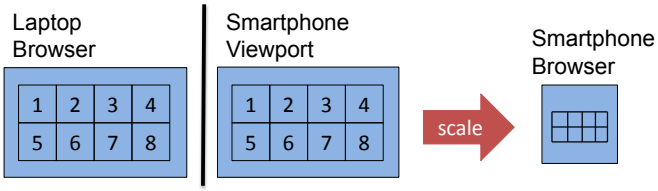
Problem: Truncation



In the early days of smartphones…
There were no responsive web apps
Web apps were not responsive to (very small) window sizes
Truncation was common

Smartphone Viewport

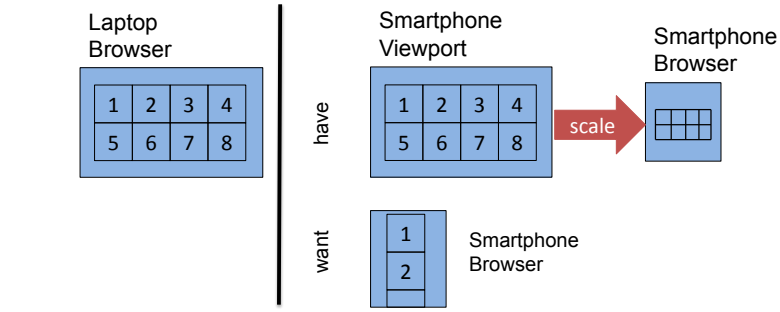
Solution: Smartphone ***viewport***



Viewport: Virtual window used by iPhone browser
And later by all smartphone browsers
Smartphone browser
Renders HTML doc to viewport
Scales viewport to physical window
No truncation, but too small!

Smartphone Viewport

Problem: Viewport inappropriate for responsive apps

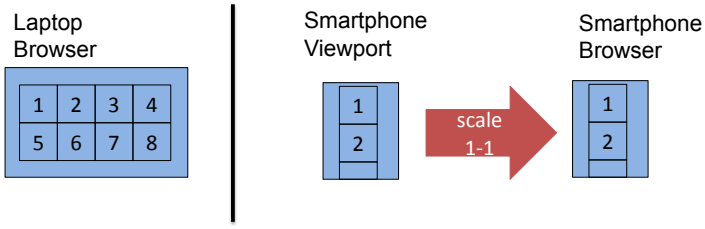


Responsive app

- Doesn't want browser to render to (large) viewport and then scale
- Wants browser to render to smartphone browser window

Smartphone Viewport

Solution: Adjust viewport



Set size of viewport to size of smartphone browser window
Scale 1-to-1

Smartphone Viewport: Example

- See **PennyViewport** app
  - runserver.py
  - book.py, database.py
  - penny.py
  - static/penny.css
  - **index.html**

29

Smartphone Viewport: Example

Code notes: index.html

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

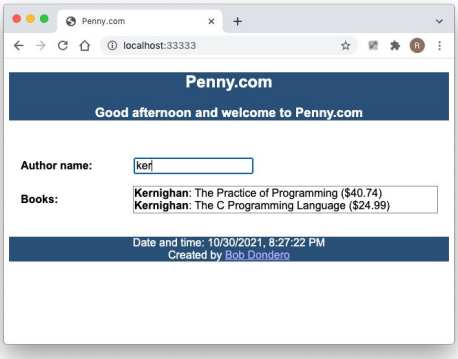
Commands browser to:

Set viewport width to device width

Scale viewport to browser window 1-to-1

Smartphone Viewport: Example

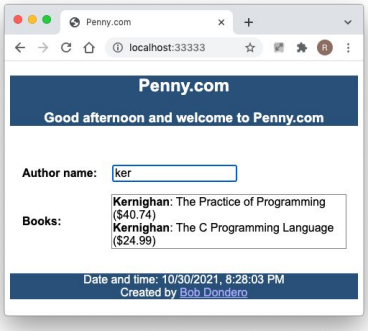
- Demos
  - Laptop computer (large browser window)



Good

Smartphone Viewport: Example

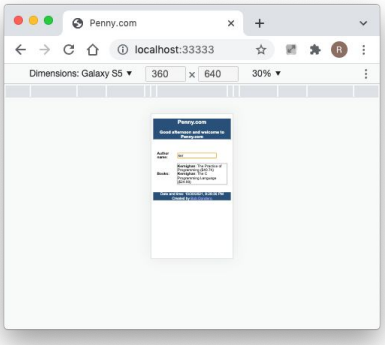
- Demos
  - Laptop computer (small browser window)



OK

Smartphone Viewport: Example

- Demos
  - Smartphone (very small browser window)



Maybe OK?

Agenda

- CSS
- The smartphone viewport
- **Responsive web apps**
- CSS frameworks

Responsive Apps: Motivation

- **Problem**:
  - Web apps should adjust themselves based upon browser characteristics (esp. window size)
  - Web apps should be ***responsive*** to browser characteristics (esp. window size)
- **Solution**:
  - Responsive web apps

Responsive Apps

- Responsive web app
  - Rearranges tables
    - Large window => multiple columns
    - Small window => fewer columns
  - Collapses menus
    - Large window => ordinary menu
    - Small window => drop-down menu
  - Hides some content
    - Large window => display all content
    - Small window => hide some content
  - Etc.

Responsive Apps

- Implementing responsive web apps
  - **Step 1**: Set the viewport
  - **Step 2**: Design app to be responsive to the viewport

Responsive Apps: Example

- See **PennyResponsive** app (cont.)
  - runserver.py
  - book.py, database.py
  - penny.py
  - **static/penny.css**
  - **index.html**

37

Responsive Apps: Example

[see slide]

Code notes: static/penny.css

I won't describe the details of static/penny.css because:

My knowledge level is shallow

You won't use this approach for Assignment 5

You probably won't use this approach for your project

Code notes: static/penny.css (not covered)

Defines class rules that create a ***grid system***

`.col-1`, ..., `.col-12`, `.row::after`

Commonly used in front-end libraries

Defines a ***media query***

`@media` rule

Detects viewport size

Adjusts styles as appropriate

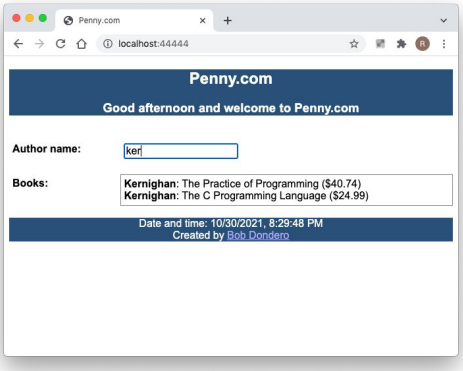
Code notes: index.html

Instead of `table` ...

Uses elements of class `row`, `col-3`, `col-9`

Responsive Apps: Example

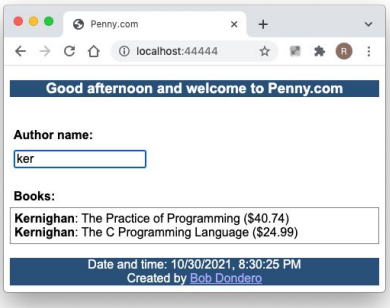
- Demos
  - Laptop computer (large browser window)



Good

Responsive Apps: Example

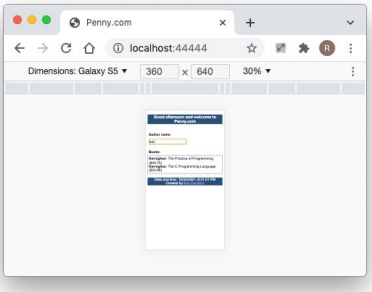
- Demos
  - Laptop computer (small browser window)



Good!

Responsive Apps: Example

- Demos
  - Smartphone (very small browser window)



Good!!!

You now know
how to compose
a mobile web app!!!

Agenda

- CSS
- The smartphone viewport
- Responsive web apps
- **CSS frameworks**

CSS Frameworks

- **Problem**:
  - Composing CSS rules is tedious
  - CSS patterns (e.g., grid system) are common
    - Redundancy across apps

CSS Frameworks

- **Solution**: ***CSS frameworks***
  - Define CSS rules
  - Examples:
    - **Bootstrap**
    - Foundation
    - YAML
    - Blueprint
    - **W3.CSS**
    - …

CSS Frameworks

- ***Bootstrap***
  - From Twitter
  - The most popular CSS framework
    - JavaScript:jQuery :: CSS:Bootstrap
  - Supported versions: 3, 4, 5
    - Version 5 doesn't work with IE11 or older
    - We'll use version 5
- ***W3.CSS***
  - From W3Schools
  - Not popular, but simple & pleasant

CSS Frameworks: Example

- See **PennyBootstrap** app
  - runserver.py
  - book.py, database.py
  - penny.py
  - ~~static/penny.css~~
  - **index.html**

45

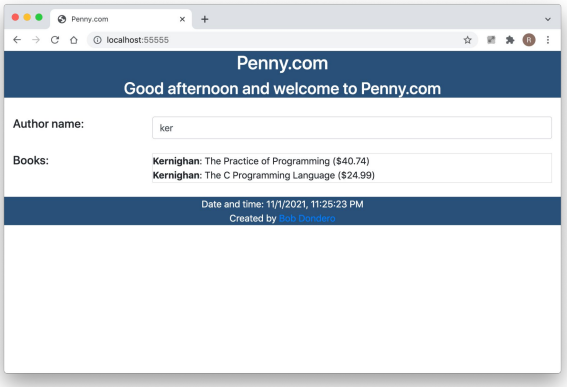
Front-End Libraries: Example

[see slide]

Code notes: search.html
Uses CSS rules defined in Bootstrap

CSS Frameworks: Example

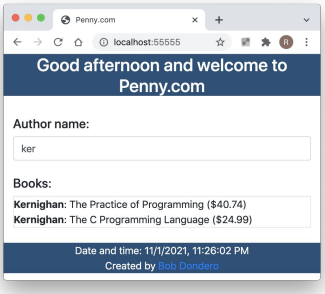
- Demos
  - Laptop computer (large browser window)



Good

CSS Frameworks: Example

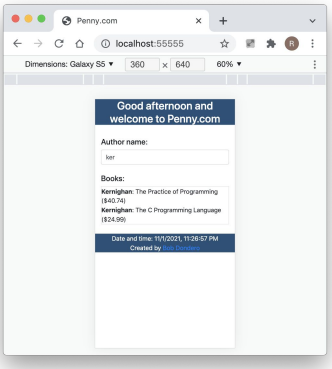
- Demos
  - Laptop computer (small browser window)



Good!

CSS Frameworks: Example

- Demos
  - Smartphone (very small browser window)



Good!!!

You now know **the easiest way** to compose a mobile web app!!!

More Information

- To learn more about **CSS**
  - *Learning PHP, MySQL, & JavaScript, 4th Edition* (Robin Nixon), Chapters 18, 19
  - https://www.w3schools.com/css/
- To learn more about **Bootstrap**
  - https://www.w3schools.com/bootstrap5/

Summary

- We have covered:
  - Cascading style sheets (CSS)
  - The smartphone viewport
  - Responsive web apps
  - CSS frameworks
    - Bootstrap