Lecture 6 – RWD

Web Application Development

September 15, 2022

Jeffrey L. Eppinger
Professor of the Practice
School of Computer Science

Lecture Schedule – 1st Half

(subject to change)

8/30	Intro
9/1	HTML & CSS
9/6	JavaScript & DOM
9/8	HTTP & Django
9/13	Cookies & Sessions
9/15	Responsive Web Design
9/20	Forms & Templates
9/22	Authentication
9/29	Models
10/1	File & Images
10/3	AJAX
10/8	jQuery & WebSockets
10/11	Cloud Deployment
10/13	Cloud DB, Email, S3

Agenda

→ Course Admin

Cross-site Scripting (XSS)

Cross-site Request Forgery

Homework #3 Discussion

Responsive Web Design

Quiz

USNews College Rankings!!!

- We’re #1!

Quiz Grades

- Ahhhh…..
  - Tomorrow? This weekend??

Warning about eduroam wifi

- You cannot access classroom demos on webapp.andrew.cmu.edu
- You must use CMU-SECURE wifi
- Connection instructions are here

Agenda

- ✓ Course Admin
- → Cross-site Scripting (XSS)
- Cross-site Request Forgery
- Homework #3 Discussion
- Responsive Web Design
- Quiz

Cross-Site Scripting (XSS)

- Attacker injects scripts (or just HTML) into a site's data
  - … that other users will see
- Example:
  - Put HTML or JavaScript into a shared to do list!
- By default, Django templates sanitize their output
  - E.g., `<script>` …
  - So don't use "safe" in templates … when there's user data
  - (I had to use "safe" in the Django template to permit XSS in this shared to do list example)
- Be very careful if you generate your own response
  - … using django.http.HttpResponse as shown in django-intro example

Shared To Do List!

- Shared “to do list” example:

  http://webapp.andrew.cmu.edu:8000/shared

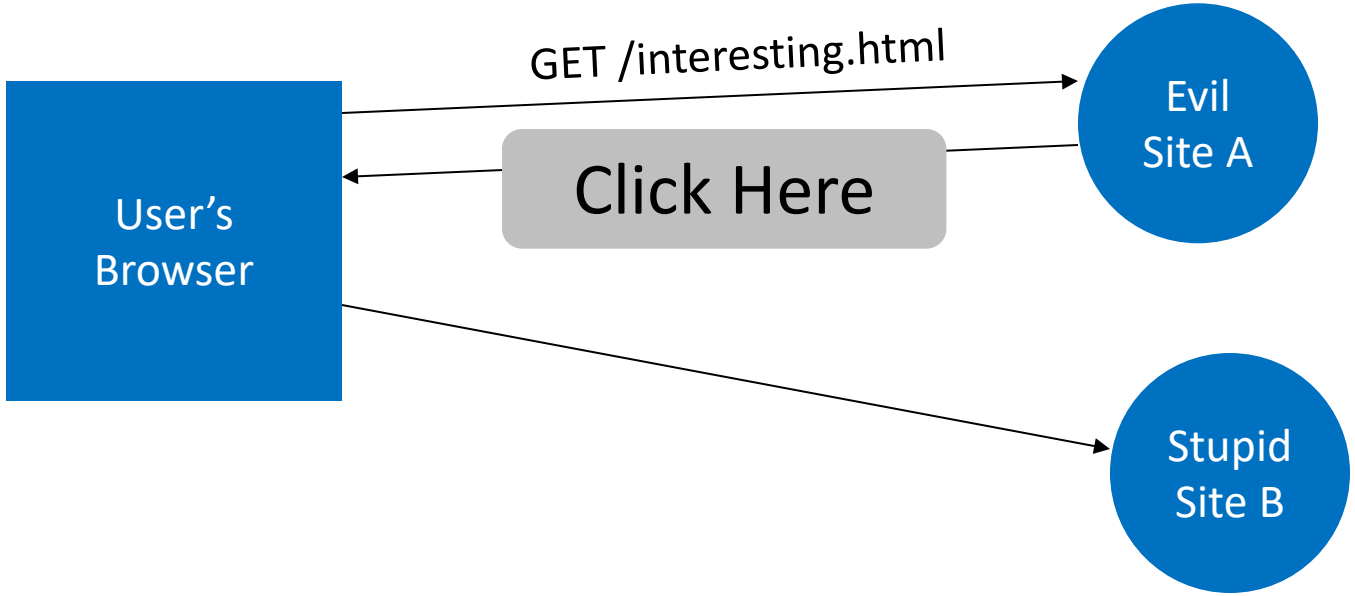
Agenda

- ✓ Course Admin
- ✓ Cross-site Scripting (XSS)
- → Cross-site Request Forgery
- Homework #3 Discussion
- Responsive Web Design
- Quiz

Django CSRF Tokens

- Checked on every income POST request
- Request must include a hidden field called `csrfmiddlewaretoken`
- Django template language will generate it with `{% csrf_token %}`
  - See this in all our example uses of `<form method="POST">`
- Check it out in the Browser's DevTools
  - Try changing it

Cross-Site Request Forgery (CSRF)



- Web page from Site A tricks user into submitting request to Site B
  - Problem if the user is already logged in to Site B (and Site B isn't careful)

Private To Do List!

- Keeps private to do list data in sessions
- Has a CSRF vulnerability

http://webapp.andrew.cmu.edu:8000/private

CSRF Example

- The private to do list is vulnerable to CSRF attack
  - Donald Trump survey: click here
- What's link?

```
<a href="http://webapp.andrew.cmu.edu:8000/private/delete-all"> click here </a>
```

- How do you prevent this?
  - Use POST when making changes to the site!
    - Because Django checks CSRF tokens on incoming POST requests
  - Set the cookie's SameSite attribute to "Strict"
    - People directed to the site from elsewhere would have to log in
- Why don't we check CSRF tokens on incoming GET requests?
  - We often create links to other sites for legitimate purposes
    - But these are only for reading or starting to access a site
  - Reading a site does not divulge information to the attacker
    - Unless you're using an unencrypted connection and attacker is eavesdropping!
  - Default SameSite cookie attribute is "Lax"
    - Blocks cookies on 3rd-party image loads

Agenda

- ✓ Course Admin
- ✓ Cross-site Scripting (XSS)
- ✓ Cross-site Request Forgery
- → Homework #3 Discussion
- Responsive Web Design
- Quiz

Homework Situation

- HW#3 due on Monday
- Looking pretty good …
  - Seems like AutoGrader is working!

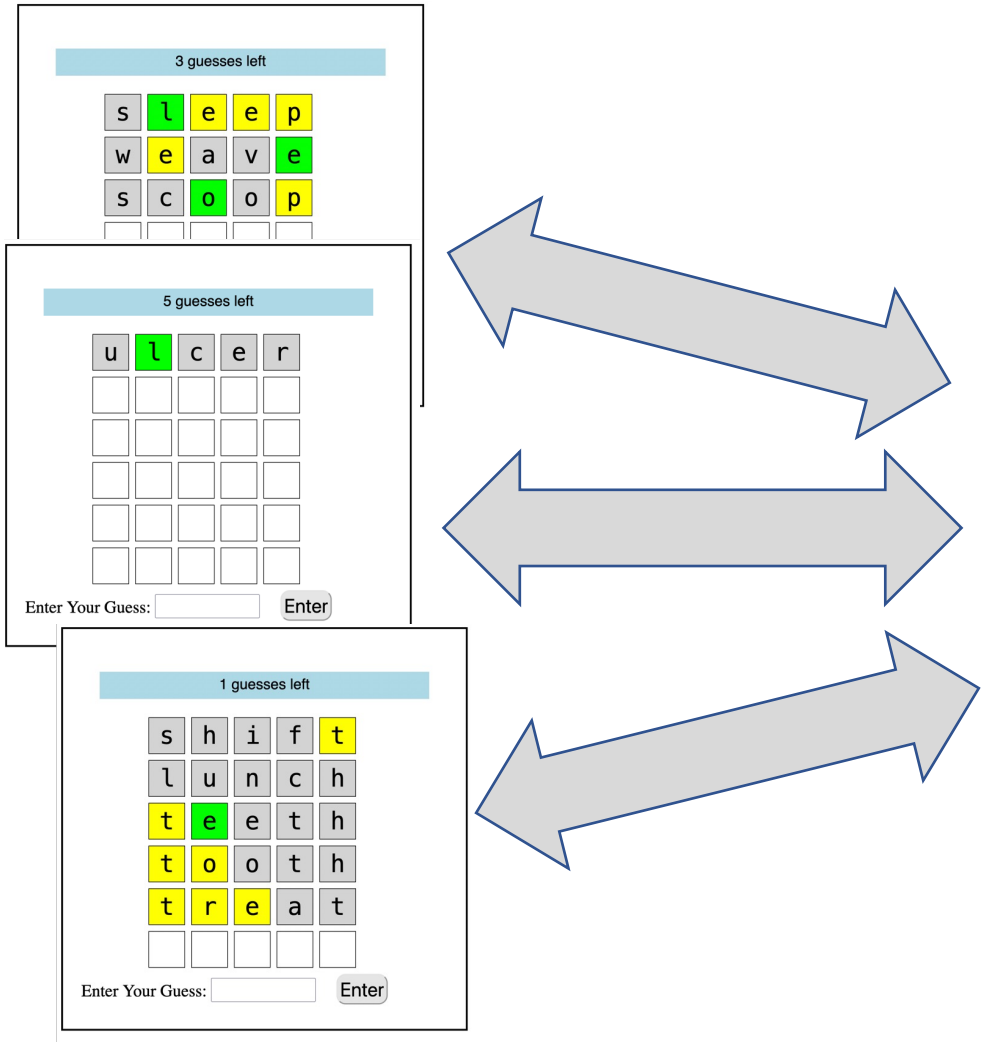
	hw1	hw2	hw3
100	117	112	9
90s	4	7	0
80s	0	1	0
70s	0	0	0
60s	0	0	2
50s	0	0	1
40s	0	0	0
30s	0	0	0
20s	0	0	0
10s	0	0	0
00s	0	0	2
users	121	120	14
runs	708	904	111

Where to start on Homework #3 – improved

- Follow the DjangoQuickstart Guide (Lecture 4)
  - Install software -- laptop or Timeshare Linux or …
  - Try out django-intro and session-hidden-examples
  - Specifically, you will want to understand GreetPost and hidden examples
- Set up your project/app in your repo's hw3 folder
  - Set up action function (in views.py) & routing (urls.py) to send your HW2 Wordish
    - Your HTML file becomes a rendered template
    - Your JS and CSS files go into static folder, fix the links in the HTML
- Then change your HW2 into HW3
  - Specifically, you'll need to remove all the JavaScript

What are your choices? (for keeping the state in HW#3)

- ~~Global variables?~~
- Hidden Fields?
- ~~Cookies?~~
- ~~Sessions?~~



Why not cookies/sessions?

1. Because all tabs share the same cookies, therefore all tabs share the same sessions, so calculators in different tabs would conflict
2. Back button not correlated with session state
3. HW spec says no state to be kept on server

Convert your Homework #2 HTML

- Buttons now send HTTP requests to the server
- Buttons need to be in a <form>, so clicking causes the form to submit
- Internal “state” needs to be in hidden fields, in the the <form>
- You must use method="POST"

wordish.html ala HW2

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status">Welcome to Wordish</div>
    <div id="matrix">
        <div class="..." id="cell_0_0">   </div>
        <div class="..." id="cell_0_1">   </div>
        <div class="..." id="cell_0_2">   </div>
        ...
    </div>
    <div class="...">
        <label>Guess:</label>
        <input id="guess_text" type="text">
        <button id="guess_button" onclick="...">Submit</button>
    </div>
</body>
...
```

wordish.html ala HW2 … with for loops

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status">Welcome to Wordish</div>
    <div id="matrix">
        <script>
            for (let i = 0; i < 6; i++) {
                for (let j = 0; j < 5; j++) {
                    document.write(`<div class="..." id="cell_${i}_${j}">   </div>`)
                }
            }
        </script>
    </div>
    <div class="...">
        <label>Guess:</label>
        <input id="guess_text" type="text">
        <button id="guess_button" onclick="...">Submit</button>
```

HW3: Make status dynamic

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        <script>
            for (let i = 0; i < 6; i++) {
                for (let j = 0; j < 5; j++) {
                    document.write(`<div class="..." id="cell_${i}_${j}">   </div>`)
                }
            }
        </script>
    </div>
    <div class="...">
        <label>Guess:</label>
        <input id="guess_text" type="text">
        <button id="guess_button" onclick="...">Submit</button>
```

HW3: Fix the for loops

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        {% for row in matrix %}
            {% for cell in row %}
                <div class="{{cell.color}}" id="{{cell.id}}"> {{cell.letter}} </div>
            {% endfor %}
        {% endfor %}
    </div>
    <div class="...">
        <label>Guess:</label>
        <input id="guess_text" type="text">
        <button id="guess_button" onclick="...">Submit</button>
    </div>
</body>
```

HW3: Make guess use a form

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        ...
    </div>
    <form class="..." action="..." method="POST">
        <label>Guess:</label>
        <input id="guess_text" type="text" name="new-guess">
        <button id="guess_button">Submit</button>
    </form>
</body>
</html>
```

HW3: Add the CSRF Token

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        ...
    </div>
    <form class="..." action="..." method="POST">
        <label>Guess:</label>
        <input id="guess_text" type="text" name="new-guess">
        <button id="guess_button">Submit</button>
        {% csrf_token %}
    </form>
</body>
</html>
```

HW3: Add the hiddens to keep the game context

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        ...
    </div>
    <form class="..." action="..." method="POST">
        <label>Guess:</label>
        <input id="guess_text" type="text" name="new-guess">
        <button id="guess_button">Submit</button>
        <input type="hidden" name="..." value="...">
        <input type="hidden" name="..." value="...">
        {% csrf_token %}
    </form>
</body>
</html>
```

You can use any hiddens you wish …

You can have separate hiddens:

```
<input type="hidden" name="target" value="{{target}}">
<input type="hidden" name="guess0" value="{{guess.0}}">
...
```

You can use a comma separated list of words for the guesses:

```
<input type="hidden" name="target" value="{{target}}">
<input type="hidden" name="guesses" value="{{guesses}}">
```

You can use any string with your own data format :

```
<input type="hidden" name="data" value="{{data}}">
```

A start action in views.py

```
def start_action(request):
    if request.method == "GET":
        context = {"message": "Welcome to Wordish"}
        return render(request, "wordish/start.html", context)

    try:
        target = _process_param(request.POST, "target")
        context = _compute_context(target, guesses=[])
        return render(request, "wordish/game.html", context)
    except Exception as e:
        context = {"message": str(e)}
        return render(request, "wordish/start.html", context)
```

A guess action in views.py

```
def guess_action(request):
    if request.method == "GET":
        context = {"message": "You're hacking.  Try again!"}
        return render(request, "wordish/start.html", context)

    try:
        target = _process_param(request.POST, "target")
        guesses = _process_old_guesses(request.POST)
        guesses.append(_process_param(request.POST, "new-guess"))
        context = _compute_context(target, guesses)
        return render(request, "wordish/game.html", context)
    except Exception as e:
        msg = str(e)
        if "Invalid input:" in e:
            return render(request, "wordish/game.html", {"status": str(e)})
        return render(request, "wordish/start.html", {"status": str(e)})
```

Compute the context so that this page can be rendered

```
<!doctype html>
<html>
<head>...</head>
<body>
    <div id="status"> {{ status }} </div>
    <div id="matrix">
        {% for row in matrix %}
            {% for cell in row %}
                <div class="{{cell.color}}" id="{{cell.id}}"> {{cell.letter}} </div>
            {% endfor %}
        {% endfor %}
    </div>
    <form class="..." action="..." method="POST">
        ...
        <input type="hidden" name="target" value="{{ target }}">
        <input type="hidden" name="old-guesses" value="{{ old_guesses }}">
</form>
</body>
```

Computing the context

```
def compute_context(target, guesses):
    matrix = {}
    for ...:
        cell = {"id": ..., "letter": ..., "color": ...}
        matrix[row][column] = cell
    status = ...
    context = {
        "status": status,
        "matrix": matrix,
        "target": target,
        "old_guesses": guesses
    }
    return context
```

You can do it the hard way

(… but instead please just use functions & exceptions to structure your code)

```
def game_action(request):
    if request.method == 'GET':
        return ...

    if "new-guess" not in request.POST:
        return render(request, "wordish/start.html", {"message": "Fatal error…

    new_guess = request.POST["new-guess"]
    if len(new_guess) != 5:
        return render(request, "wordish/start.html", {"status": "Invalid input…

    ...
```

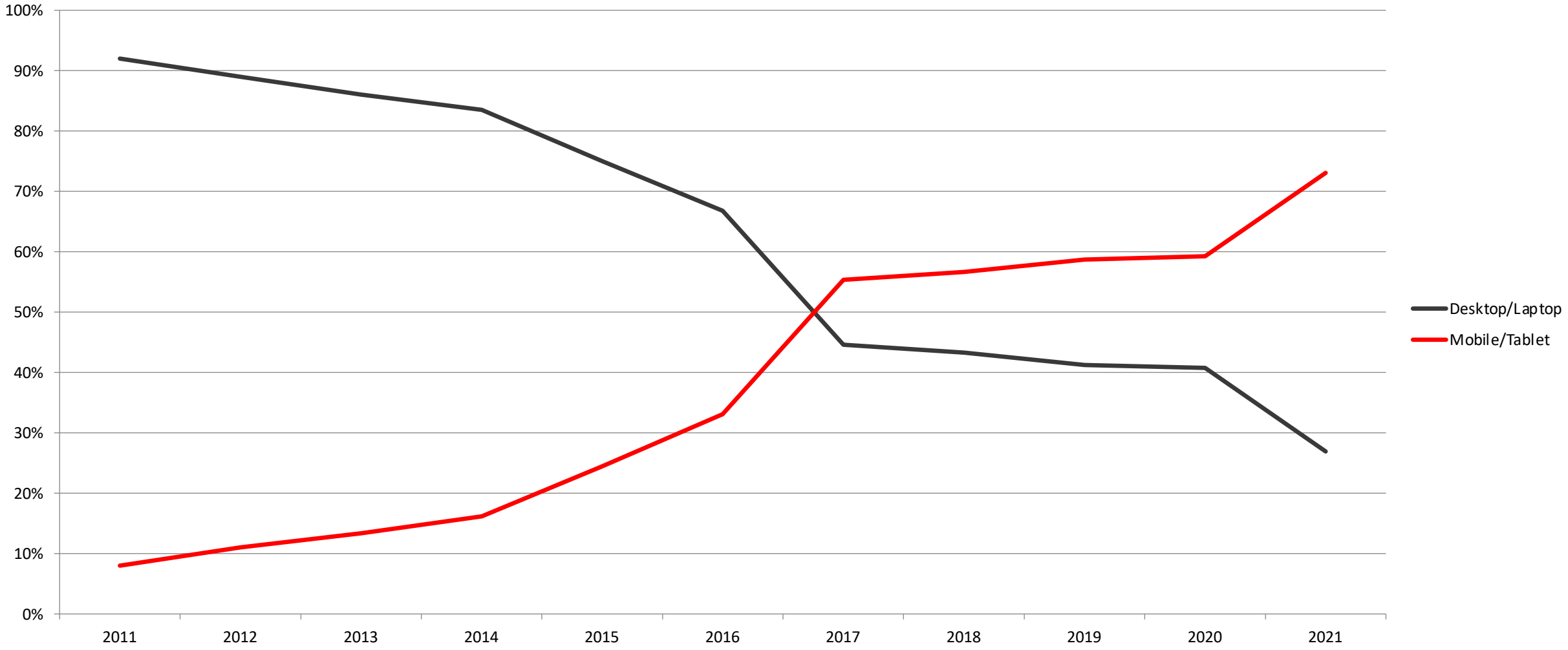
Agenda

- ✓ Course Admin
- ✓ Cross-site Scripting (XSS)
- ✓ Cross-site Request Forgery
- ✓ Homework #3 Discussion
- → Responsive Web Design
- Quiz

Responsive (Web) Design

- Constructing web pages that scale appropriately on different devices
- Primary motivation is to make pages that scale appropriately for:
  - Mobile
  - Tablet
  - Laptop/Desktop

Browser Market Share (Desktop/Laptop vs Mobile/Tablet)



Source: NetMarketShare
https://netmarketshare.com

Responsive CSS

- You can use sizes relative to the size of the current font or of the viewport
  - Example: set font size to be 10% of browser width `{ font-size: 10vw; }`
  - Example: set element width to be twice font size `{ width: 2em; }`
  - Google “CSS Units” for a list
- You can specify different CSS depending on the viewport type and size
  - Example: specify CSS to be used for screen heights less than or equal to 500 pixels

    ```
    @media screen and (max-height: 500px) { … }
    ```

  - Google “CSS media query” for details
- Let’s take a look at https://www.cmu-webapps.org and AutoGrader

Cellphone Viewports

- When browsers first appeared on phones…
  - Pages were not designed for small screens
  - Mobile browsers render pages on a large “virtual” screen
  - Users can pan and zoom to view the page and find what they need

- Add the `<meta name="viewport">` tag to:
  - Use actual screen size
  - Specify the initial zoom-level

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

CSS Grid

- Classically, we use the <table> tag to layout rows and columns
- The CSS { display: grid } allows you to make a tag layout like a table
- Specify the number and sizes of columns and rows:

```
#container {
    margin: 0.2em;
    display: grid;
    grid-template-columns: 2em 2em 2em 2em;
    grid-template-rows: repeat(6, 2em);
    grid-column-gap: 0.4em;
    grid-row-gap: 0.4em;
    font-size: 10vw;
}
```

Let’s checkout the HW

- I have made a few variations
  - Unresponsive design (without viewport)
  - Unresponsive with viewport
  - Responsive Web Design version
  - Responsive Web Design version (without viewport)

UI Frameworks

- Takes the hassle out of specifying all that CSS
- Responsive web design
- Provide a consistent look and feel
  - Made many design decisions for you
- Addressed a lot of the issues related to different device sizes
  - Example: switching navigation display depending on viewport size
- Popular Frameworks
  - Bootstrap (https://getbootstrap.com)
  - Materialize (https://materializecss.com)
  - Angular (https://angular.io)
  - React (https://reactjs.org)

Bootstrap

- Very popular UI Framework
  - Uses JavaScript, CSS, & images
- An internal project at Twitter
  - Released as open source on GitHub in 2011
- Supports current versions of HTML & CSS
- Supports all major browsers
- My example is Bootstrap 3
- Bootstrap 4 and 5 are out, but I haven't had a chance to upgrade
- W3Schools documentation (as well as Bootstrap's docs)
- I built this one just to check it out
  https://eppinger-homepage.appspot.com

Bootstrap Code

- Open Source on GitHub
  - https://github.com/twbs
- You can download it and include it with your static files
  - http://getbootstrap.com
- Easiest is to reference the files from a CDN

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Bootstrap Responsive Design

- Adjusts to the device's screen size
  - Looks good on phones & tablets
  - On desktop, adjusts to window size

- Include tag to set this up:

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

Bootstrap Grid

- Divided screen into 12 units across
- Allows a good deal of flexibility

E.g., https://getbootstrap.com/docs/3.4/examples/grid/

Bootstrap Typography

- Set the style for HTML5 Tags
- The tags already existed, but Bootstrap provides CSS to make it have a consistent style
- Examples: https://getbootstrap.com/docs/3.4/css/

Bootstrap Components

- Input
  - Buttons
  - Dropdowns
- Tabs
- Pagination
- Progress Bars
- Alerts

See https://getbootstrap.com/docs/3.4/components/

Customization

- You can customize many, many components of Bootstrap
- Generate new files to include in your site

See: https://getbootstrap.com/docs/3.4/customize/

You can do it yourself

- HTML
- CSS
- JavaScript

- Let’s checkout https://www.jeffeppinger.com

Agenda

- ✓ Course Admin
- ✓ Homework #3 Discussion
- ✓ Responsive Web Design
- → Quiz

Lecture #6 Quiz

- There's link on Canvas which takes you to:
  https://www.cmu-webapps.org/quiz6.html
- Notice the at the bottom of the page does not scale
  …when the width of the page changes
  - Nothing on this page scales
- Make the span most of the page width (80% – 90%)
- Change the <style> for the so that it responsively scales
  - Both the width and font-size of the must responsively scale
- If you like, see if you can get the rest to responsively scale
- Push your solution as described