

DGMD E-28

Single Page Web Applications

Spring 2024

1

Introduction

- This is a programming course
- BUT...
- It is also a course in features of WordPress
- You should already have
 - Basic website knowledge
 - Basic programming skills
- Section meetings will provide extra help with basic skills
- The pre-course survey will assist you in identifying any skills you need

2

Overarching goals of this course

- Take web development skills to the next level
- Take software development skills to the next level
- Learn to build engaging, useful single page web applications



3

A little about myself

- Lifelong programmer (since I was 14)
- Experience in business and marketing
- Own/operate Fembot Creative
- University level instructor for over 20 years.
- I am currently at Harvard Extension School, Harvard Summer School and Tufts University teaching C++, Website Planning, WordPress Programming, Web Programming

4

Teaching Assistant: Michael Jacobs

- High school Math and economics teacher turned web developer
- Grad certificate in Programming from Harvard Extension
- Currently working as a web developer

5

Course logistics

- **Lecture** for demos and theory
- **Quizzes** for demonstration of theoretical knowledge
- **Weekly assignments** to practice individual concepts
- **Projects** to help understand the bigger picture
- **Section / Office Hours** (optional) for Q&A and special topics.
Time: Tuesday at 8:30pm, Wednesday at 7pm
(the schedule will be adjusted if needed)
Additional Office Hours by request.
- *All times are EST*

6

Course logistics

- **Canvas** – course materials: assignments, “handouts”, code, quizzes, syllabus. All links and info will be in each module (in Canvas, navigate to “Modules”)
- **Ed Discussion** – course communications: discussion forum, Q&A, help.
- **If you get lost-** This course covers a lot of ground and students are at many different levels. If you don’t understand something, please just ask.

7

Class Policies

- **Attendance:** Attendance will not be taken at lectures. However, it is strongly suggested that you watch lectures in “real time”, or within 24 hours of the class time..
- **Late Assignments:**
 - 1 hour “grace period” past the due date
 - Up to 24 hours, 8 point deduction
 - Up to 72 hours, 15 point deduction
 - After 72 hours, not accepted
 - Requests for a 24 hour extension will be considered with *advance* notice.
- **Quizzes:** Quizzes are online and can be taken any time within a 48 hour period.

8

Course Roadmap

- Refresher: HTML/CSS
- JavaScript- basics
- JavaScript- advanced including new features
- jQuery
- JSON
- AJAX / Asynchronous access to the server
- API's
- React.js
- New React JavaScript features
- React components
- JSX
- State Management
- Dynamic binding
- React Hooks
- React Tools
- Deploying a React project

9

Website vs Web App

- Website
- Web App

10

The HTTP Request

11

Anatomy of a URL

- Uniform Resource Locator
- <https://now.feedme.com/snacks.php?flavor=salty>
- Protocol
- Domain (TLD + domain name)
- Subfolder/subdomain
- File (default file)
- Query string

12

The Domain is Mapped to the IP Address of the Server

Domain name ~ human friendly \Leftrightarrow IP address ~ computer friendly

- IP Address is 32 bits (4 sets of numbers, 0 – 255)
 - 81.138.15.252
- Every connection point to the internet has an IP Address
- Domains are registered with the Internet Corporation for Assigned Names and Numbers (ICANN) via a domain registrar company
- The DNS records (DNS = Domain Name System) for the domain “point to” the hosting by setting the A record to the IP address of the host.
- Tools:
 - whatsmydns.net
 - whoishostingthis.com
 - bustaname.com
 - whois.com

13

Back At the server ...

- There is a mapping at the server for each domain name to the root folder where the corresponding files are located
- If the end of the URL is a file – then that file is served. Otherwise, there is an attempt to locate the default file (index.html, index.php, default.php - dependent on the server settings)
- Action depends on the file type
 - .html => send file to the client
 - Server-side code (node.js, php, aspx, etc) => executes on a server
 - The query string is passed to the server-side code
 - The server sends a browser readable file to the client
- Now it's the browser's turn. The browser will
 - display content formatted via **HTML** markup
 - execute JavaScript or other **Client Side** code
- The browser may
 - request other resources from the server - images, include files
 - access the server asynchronously (this is common for an SPA)

14

HTTP Request Visualized

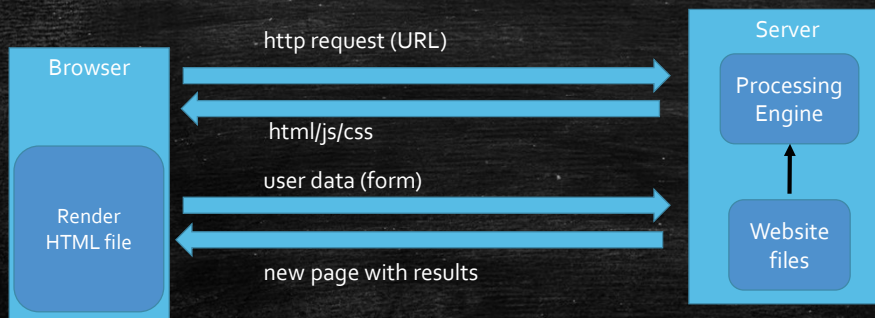


15

What is an SPA?

16

Traditional Web App



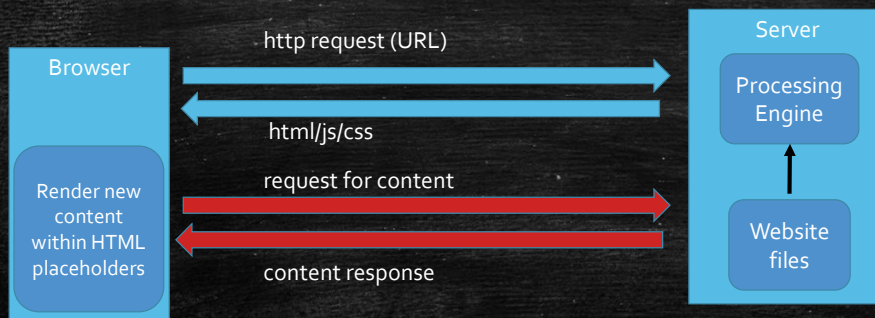
17

How does an SPA work?

- Single Page Web Application
- Consists of HTML with placeholders plus JavaScript.
- The placeholders are dynamically populated with content from the server or other source via JavaScript.

18

Single Page Web App



19

Features of an SPA

- Feels like a client-server app, but has much of the speed and performance of client only.
- Interactive
- Rich User Interface
- Page does not need to be reloaded
- Resources are loaded as needed often in the background
- Some examples: Facebook, Gmail, Trello, and Twitter
- Various open-source frameworks based on JavaScript have adapted to the paradigm of an SPA including:
 - AngularJS
 - React.js
 - Vue.js
- In this course we will explore React.js

20

Should Every Web App be an SPA?

- SPA advantages
 - Normally faster loading time
 - No reload needed
 - App look and feel
- SPA disadvantages
 - Poorer metric for rankings with search engines (that think more pages is better)
 - Needs state of the art browser capabilities
 - Needs JavaScript
 - More security issues such as exposing data

21

Why React.js

- Fast
- Integrates with HTML – leverage familiarity with JavaScript
- Build apps quickly
- Used by some big players including Facebook and Instagram

22

Challenges to learning React.js

- New JavaScript constructs
- Learning JSX
- Concept of dynamic binding
- Concept of components to extend functionality and modularize your app
- Concept of routes to create a multi-view app
- Concept of asynchronous programming to grab data from the server
- Building a React app
- Deploying a React app

23

Some inspiration!

<https://codesandbox.io/p/sandbox/react-js-simple-calculator-pefmr>

24

HTML & CSS Refresher

25

Anatomy of an HTML web page



26

HTML:
Hello World

```
<html>
<head>
  <title>Cool Page Here</title>
</head>
<body>
  <h1>Hello World!</h1>
  This is my cool hello world page
</body>
</html>
```

27

HTML Components

- Tags / Container tags
- Attributes
- Values
- Entities: describe a character
 - Start with & End with ;
 - Ex: ©

Tags that wrap content are container tags
<h1> MyTitle </h1>
Add a slash to a standalone tag to let it serve as a
beginning and end tag (best practice)

<h1 style='text-align: center;'>This is my page title</h1>



28

Basics

- HTML ignores whitespace.
 - Paragraph `<p>` (double line break)
 - Break `
` (single line break - not a container tag)
 - Div `<div>` (single line break – a container tag)
 - Non-breaking space ` ` (a space)
- Headings `<h1>`, `<h2>`, ... `<h6>`
 - `H1` is *most important*
- Horizontal rule `<hr>` (not a container tag)

- Formatting Tags

- Can also be done with CSS
- `` bold (also ``)
- `` italic (also `<i>`)
- `<SUP>`, `<SUB>` superscript, subscript
- Tags can be nested

```
<strong>text1<em>text2</em>
text3</strong>text4
text1 will be bold
text2 will be bold and italic
text3 will be bold
text 4 will be normal
```

29

Deploying a Web Page

Option 1: View the page locally

- Create the file. Save with .html extension
- Open a browser and put the full path to the file in the url bar (the protocol is <file:///>)
- Or easier, drag/drop the file from file explorer into the browser
- This is for LOCAL VIEWING ONLY
- Any additional resources must be in the same folder or referenced via relative path

Option 2: View the page on the web

- Upload the file to a server on the web (in a hosting account)
 - note the web address
- Open a browser and enter the web address
- Add the file name to the url
- Example: www.mysite.com/help.html
- Any additional resources must be in the same folder or referenced via absolute URL or relative path

30

Deploying on Github

- Github implements "git" which provides version control and ease of collaboration
- We will use github.com in this course as a deployment engine and host

1. Establish account on github
2. Add a new repository (repo)
3. Add a file to the repo
4. Enter the code and Commit to save
5. Go to Settings -> Pages
Deploy from master branch
6. Wait while it is created
7. You will be able to see the URL at the top of Pages

31

Hyperlinks <a>

- Link `text to be linked`
- Use a complete URL for external sites
- Click to link email and phone
 - ``
 - ``
- To open a link in a new window or tab: `target="_blank"`
ex: ``

32

images

- ``
- `img` is *not* a container tag
- Will auto-size but may deter performance
- Attributes for accessibility: `alt`, `title`
- Formats: `jpg`, `png`, `WebP`
- Tradeoff – quality of image vs load time
- Finding images
 - Custom photography (best)
 - [unsplash.com](#)
 - Stock photos
 - Creative commons
 - Be mindful of attribution



Photo by Unknown Author is licensed under [CC BY-SA](#)

33

HTML Lists

- Unordered (bulleted) ``
- Ordered (numbered) ``
- List item ``
- Attributes: `start`, `type`
- You can create an indent level by creating a list inside of an ``
- CSS styling to shut off bullets
`<ul style="list-style-type: none">`
- Lists are often used to group items for custom styling such as navigation

Ordered List

1. first item
2. second item
3. third item

Unordered List

- first item
- second item
- third item

```
<h3>Ordered List</h3>  
<ol>
```

```
  <li>first item</li>  
  <li>second item</li>  
  <li>third item</li>
```

```
</ol>  
<h3>Unordered List</h3>  
<ul>
```

```
  <li>first item</li>  
  <li>second item</li>  
  <li>third item</li>
```

```
</ul>
```

34

HTML Tables

- Create a table: `<table>`
- Tables are made of rows: `<tr>`
- Rows are made of elements (table data) `<td>`
- "Header" rows are bold and centered `<th>`
- Rows are auto-sized to the tallest item in the row
- Columns are auto-sized to the widest element in the column
- Use ` ` for an empty cell

Effect	Description	Results or Outcomes	Other Interesting Facts

35

HTML forms

- `<form>` tag
- `<input>` tag for text, radio, checkbox, submit, reset
- Radio buttons in the same group all have the same name
- `<select>` for drop-down/list
 - `<option>` tag for each item
- `<textarea>` for multi-line text
 - container tag
 - rows, cols attributes specify number of rows/columns
- Use CSS to align elements
- Use id, name to identify elements for use in script

Pet Information Form

Name:

Create a password:

Where did you hear about us?

☐ Friend
☐ Internet
☐ Other

Type of pet: ☒ Dog ☐ Cat ☐ Hamster

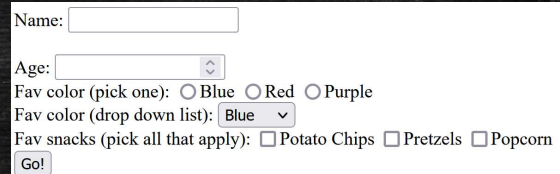
Alternate type of pet:

Describe the problem:

36

Form Example

```
<form action="#" method='post'>
<label>Name:</label> <input type="text" name='name' id='the_name'><br />
<br />
Age: <input type = "number">
<br />
Fav color (pick one):
    <input type="radio" name="color" value="blue">Blue
    <input type="radio" name="color" value="red">Red
    <input type="radio" name="color" value="purple">Purple
<br />
Fav color (drop down list):
    <select name="favColor">
    <option value="blue">Blue</option>
    <option value="red">Red</option>
    <option value="purple">Purple</option></select>
<br />
Fav snacks (pick all that apply):
    <input type="checkbox" name="chkPotChip" value="potato_chip">Potato Chips
    <input type="checkbox" name="chkPretzel" value="pretzels">Pretzels
    <input type="checkbox" name="chkPopcorn" value="popcorn">Popcorn
<br />
<input type = "button" value = "Go!">
```



37

CSS (Cascading Style Sheets)

- Allows formatting to be separate from content
- Define style instructions through style rules
- Each rule has a selector and a set of property/value pairs

```
selector { style-property1:value; style-property2:value }
```
- The *selector* indicates what the rule applies to.
- The *properties* are the style characteristics that are being modified.
- The *value* is the new value for that property.
- Multiple property/value pairs are separated by ;

38

Example

```
h1 {text-align:center; color: #ff0000;}
```

- Selector: h1
- Property: text-align, Value: center
- Property: color, Value: #ff0000
- This **rule** states that all h1 tags should be centered on the page and colored red.
- **!important**
at the end of any property-value pair, !important indicates priority
 - text-align:center !important;

39

Style rule placement

- **Inline**
 - "at the tag"
 - <p style="color:#ff0000">
- **Internal: within <style> tag**
 - Less "important" than inline
 - The *closer* a rule is to the selected element, the *stronger* the precedence

```
<style type= 'text/css'>
  h1 {text-align:center;
      color: #ff0000;}
</style>
```
- **External: In a separate file**
 - By convention, style.css
 - No style tag
 - Include on a page using:
<link rel="stylesheet" href="styles.css">

40

Selectors

Type of selector	Example
An HTML tag Applies to any instances of that tag on the page	h1
A style class (starts with a '.') Applies for: class="x" <tag class="x"my-class
An id (starts with a '#') Applies for: id="x" <div id="x" ...	#the-id
A pseudo-element (starts with a ':') modifies another selector	li:first-child

41

Selector combinations

- a:hover rule applies when mouse is over the link
- a,b rule applies to all (note: no space after comma)
- a b rule applies when b is *contained* within a
ex: "h1 em" applies for
"<h1> here is text</h1>"
- tag [attr=value] rule applies to specified tag only when the attribute is set to the given value
- tag.x rule applies for tag when its class="x"
<h1 class="x" ...

42