CSCI S-33a (Web50)
Section 5

Ref: Lectures 6 (User Interfaces)

Vlad Popil

Jul 22, 2020

My Info

About me:

Vlad Popil
Master's (ALM) in Software Engineering
Principal Data Analyst at Capital One

Email: volodymyr.popil@gmail.com

Sections: Wed 8:00-9:30 pm EDT **+** 1st week only on Thu 8:00-9:30 pm
Office Hours: Sat 2:00-3:30pm EDT

Agenda

- Logistics
- Lecture review
- Animation
- Project 4
- Grading criteria (reminders)
- cURL/Postman
- `pycodestyle` , `pylint`, `jshint`(recap)
- Tips
- Q&A

Logistics

Reminders

- Zoom:
  - Use zoom features like raise hand, chat and other
  - Video presence is STRONGLY encouraged
  - Mute your line when not speaking (enable temporary unmute)
- Projects:
  - Start early (or even better RIGHT AWAY!!!)
  - Post questions on Ed platform
  - Remember: bugs can take time to fix
  - Grade -> 3 × correctness + 2 × design + 1 × style
  - Lateness policy - 0.01 per minute x 60 x 24 x 7 days ~ 100
  - Set a reminder to submit the form for each project
  - Online search, Ed platform, etc.
  - Documentation
  - Project 4 - Due Sunday, Jul 26 at 11:59pm EDT (**4 DAYS LEFT**)

Reminders

- Sections/Office Hours:
  - Sections are recorded, office hours are not
  - Real-time attendance encouraged
  - Video and participation encouraged even more
- Section prep:
  - Watch lecture
  - Review project requirements
- Office hours prep:
  - Write down your questions as you go, TODO, etc.
  - Come with particular questions

10,000 foot overview

- *Section 0 (HTML, CSS)* - Chrome Dev Tools (Inspector), Grading aspects, Overviews
- *Section 1 (Git + Python)* - Python Installation, IDEs, CDT (Network), Project 0
- *Section 2 (Django)* - Markdown, *RegEx*, IDEs extra, pycodestyle, Debugging, Project 1
- *Section 3 (SQL, Models, Migrations)* - IDE's, linting, DB modeling, Project 2
- *Section 4 (JavaScript)* - cURL/Postman, jshint, CDT + IDE's Debugging, Project 3
- *Section 5 (User Interfaces)* - Animations, DB modeling, Pagination, Project 4
- *Section 6 (Testing, CI/CD)* - Test Driven Development, DevOps, Final Project
- *Section 7 (Scalability and Security)* - Cryptography, CAs, Attacks, App Deployment

Most sections: material review, logistics, project criteria review, reminders, hints, etc.

Burning Questions?

Please ask questions, or topics to cover today!

Topics:

- *DB model*
- *CSRF Token - keep enabled we'll show how to plug in from JS*
- *Elegant way to enable "Follow/Unfollow" switching*
- *No React needed*

Postman Follow Up From Last Week

Demo...

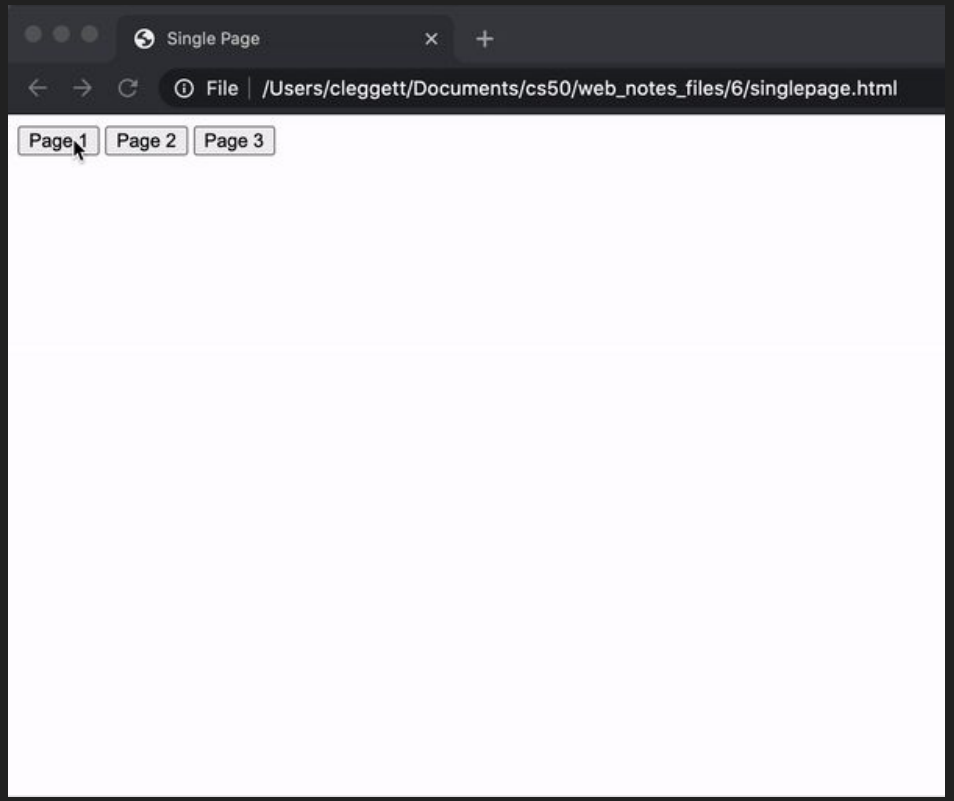
Lecture Recap

5-10 min

User Interfaces

- We want the User Experience to be as good as possible on our websites.
- There are many methods we can use to improve our interfaces:
  - Visually appealing pages (CSS)
  - Single-Page applications (Javascript)
    - React is one way of doing this
  - Animation (CSS)

Single-Page Applications



Single-Page Applications

Advantages:

- Only need to re-render the parts of the page that are changing.
- Often much faster than switching pages
- Debugging in Google Chrome

Disadvantages:

- A bit more difficult to manage the URL and history if you wish to do so.
- Have to be more careful about security.
- Initial Download could be slower

Using APIs

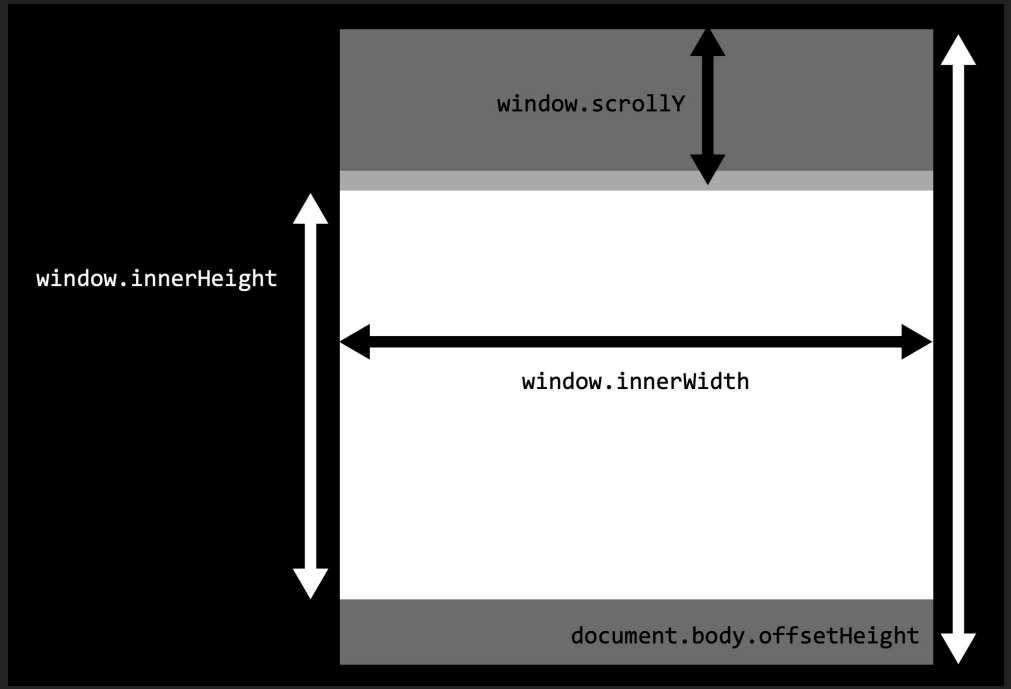
- We can use APIs to update the data associated with our site in JavaScript
- Sometimes, we'll have to write these APIs ourselves

Using History

- Use history.pushState function to add to the browser history, and then set the window.onpopstate attribute to change behavior when back arrow pressed.
- history.pushState(data, title, urlExtension);
  - data: A JavaScript object with any information you would like to associate with the current state
  - title: Title of the state, ignored by most web browsers
  - urlExtension: What should be displayed in the url
- window.onpopstate = myFunction(event);
  - Access data using event.state.parameterName
  - This function will be run whenever the back arrow is pressed

Infinite Scroll

- You can choose to load new data only when the user gets to a certain part of the page.



Animation

- We can use some simple CSS to animate our page!
- First use @keyframes to create an animation
- Then apply animation in CSS

```
@keyframes animation_name {
    0% {
        /* Some styling for the start */
    }

    75% {
        /* Some styling after 3/4 of animation */
    }

    100% {
        /* Some styling for the end */
    }
}
```

```
h1 {
    animation-name: grow;
    animation-duration: 2s;
    animation-fill-mode: forwards;
}
```

React

- Javascript Framework
- Uses Declarative Programming: stating the logic of what we wish to display without worrying about the precise content.
- Uses JSX, which combines HTML and JavaScript
- Some sites built with React:





Including React

- Include within HTML:
  - Link to React, React-DOM, and Babel in HTML page

```
<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

  - Write your components in a script tag.
  - Render component:

```
ReactDOM.render(<App />, document.querySelector("#app"));
```

  - A bit slow, as we need to load these libraries each time
- create-react-app
  - Created by Facebook for React Developers
  - Automatically generates baseline react app

Components

- Individual parts of a website like a nav-bar, a post, or even a page.
- Can be passed props (a bit like arguments) when rendered
- They can also store information in their state
    - When you wish to do this, you should include a constructor method to be run when the component is first created.
- Must include a render function that returns some HTML to render

Components: Syntax

```
class App extends React.Component {
    constructor(props) {
        super(props);
        // Initialize the state
        this.state = {key: value; ...}
    }
    render() {
        // You should only return one thing
        return ( <div> {this.state.key} </div> );
    }
}
```

Components: Changing State

```
class App extends React.Component {
     constructor(props) {
          super(props);
          this.state = {key: “value”;}
     }

     render() {
          return ( <input onChange={this.updateKey} value={this.state.key}> </input> );
     }

     updateKey = (event) => {
          this.setState({
               key: event.target.value
          });
     }
}
```

Using createreactapp

1. Install Node.js.
2. In your terminal, run `npx create-react-app [app_name]`
3. cd into new app name folder
4. Edit App.js to change what is displayed
5. Run `npm start` in the terminal to run your application

Pagination

- In addition to infinite scroll, we can use pagination to allow the user to view only a set number of posts at a time.
- Example: Google Search
- Django Documentation

Questions?

Demo

Animation quick review

A few examples...

Project

Project 4

Common suggestions:

- Start early!!! 4 Days left
- Google Form
- Make a checklist of requirement and check all before submission
- Make sure there’s no bugs
- Focus on functionality (NOT PRETTINESS)!!!

Project 4

- Models:
  - User model can point at self
  - Likes can be a standalone model, or a ManyToMany
  - Modeling exercise?
- All Posts vs Profile vs Following
  - Can be done as three HTML files or can reuse index for all three
  - If reusing index:
    - Can show the profile conditionally
    - Can show follow/unfollow buttons conditionally (when? If not your page)
      {% if request.user != profile_user %}
    - Can show New Post conditionally (when? Not under following, and not on else profile)
- New Post:
  - Conditionally if reusing index.html, otherwise do we need condition? Probably no (for separate page)

Project 4

- Token:
  - const token = document.querySelector('input[name="csrfmiddlewaretoken"]').value;
  - fetch(`posts/`, {json},
    headers: {
    "X-CSRFToken": token
    }...)
- Pagination
- Posts / dataset-
- Edit render

Design

What can be considered (not exclusively):

- Proper refactoring (copy-paste is usually a no-no)
- Use of constants:
  - 1. const
  - 2. let
  - 99. var
- Proper use of functions
- More reasonable solution
- Code/file structure

Design (continued)

What can be considered (not exclusively):

- Repetitive use of querySelector?
- Proper data structures
- == vs === ?
  - const x = 5
  - const y = ‘5’
  - x == y -> T
  - x === y -> F
- Code repetition

Style

What can be considered (not exclusively):

- *pycodestyle (indentations, line breaks, long lines)*
- COMMENTS!
- Naming for variable, function, files, etc.:
  - getemailbyid -> get_email_by_id (Python convention)
  - getEmailById (JS convention)
- Consistency is the key!

Style (continued)

What can be considered (not exclusively):

- ‘ vs “ consistency
- camelCase(c*, Javascript, Java) vs snake_case (Python)
- == vs ===

IDEs and Debugging

cURL / Postman

Allows to call API endpoints directly.

Demo...

PyCharm

- Debugging

pycodestyle (formerly pep8) - style check

- pip install pycodestyle
- pycodestyle views.py --max-line-length=120

pylint - quality, bugs + style

- pip install pylint
- pylint views.py

jshint

- UI:
  - https://jshint.com/

- CLI:
  - brew update
  - brew doctor
  - brew install node
  - npm install -g jshint
  - In ~/.jshintrc add:
    - {
    - "esversion": 6
    - }

Random Tips

- HTML beautifiers/prettify
- Windows licence (https://harvard.onthehub.com/)
- Video Speed Controller
- The Great Suspender
- GitHub Education Pack
- Spotify

Fruit of the day

<<< If you are watching this recorded >>

Please email the word: **WATERMELON**

To: volodymyr.popil@gmail.com

Thank you.

Q&A

Please ask any questions. Ideas:

- Anything discussed today
- Anything from lecture material
- About the project
- Logistics
- *Random*

Resources

- https://github.com/vpopil/e33a-sections-summer-2020

CSCI S-33a (Web50)
Section 5

Ref: Lectures 6 (User Interfaces)

Vlad Popil

Jul 22, 2020