CSCI S-33a (Web50)
Section 4

Ref: Lectures 5 (JavaScript)

Vlad Popil

Jul 15, 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
- JS / Ajax
- Project 3
- Grading criteria (not exhaustive)
- Chrome Debugging - Javascript
- IDEs
- cURL/Postman
- `pycodestyle` , `pylint` (recap)
- `jshint`
- 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 2 - Due Sunday, Jul 19 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:

- *UI vs backend*
- *console.log() - better to cleanup*
- *fetch() where it starts, do after fetch is done*
- *Load existing emails when switching view Archive/Unarchive*
- *On Reply, add formatting, look better.*

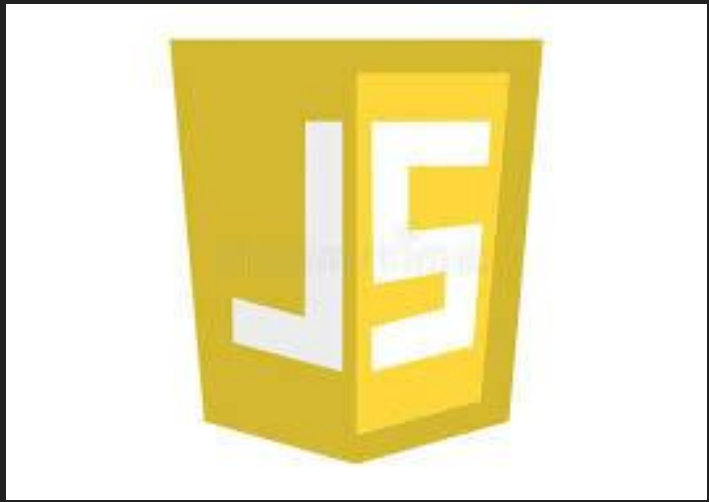
Lecture Recap

5-10 min

JavaScript

JavaScript

- Programming Language used for running client-side code.
- Useful for manipulating the Document Object Model (DOM)
- Employs Event-Driven Programming



Include JavaScript in page

- Write code between <script> tags in HTML

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hello</title>
        <script>
            alert('Hello, world!');
        </script>
    </head>
    <body>
        <h1>Hello!</h1>
    </body>
</html>
```

- Link to separate JavaScript file (preferred)

```
<script src="counter.js"></script>
```

Functions

- Regular notation
- Arrow Notation
- Anonymous Functions

```
1   // Traditional function notation
2   function add(a, b) {
3       return a + b;
4   }
5
6   // Arrow notation
7   add = (a, b) => {
8       return a + b;
9   }
10
11  // Anonymous function
12  // (We've given it a name here, but no need to)
13  var anon = function(a, b) {
14      return a + b;
15  }
```

Variable Declaration

- var: Declares a function-scoped variable
- let: Declares a limited-scope variable
- const: Declares a constant value

```
// Declaring a Constant
const PI = 3.14;

// Declaring a function-scoped variable
var some_var = "Hello!";

// Declaring a limited scope variable
let i = 1;
```

Selecting DOM Elements

- document.querySelector(query) selects the first element matching the query
- document.querySelectorAll(query) selects all elements matching the query

Examples:

Query	Result
`...Selector('#header1');`	Selects element with ID header1
`...SelectorAll('.big');`	Selects all elements with class big
`...SelectorAll('image');`	Selects all images
`...Selector('form');`	Selects first form on the page

Manipulating DOM Elements

Attribute/Command	Meaning	Example
`element.innerHTML`	All HTML inside element	header.innerHTML = “Hi!”;
`element.style.property`	The CSS styling of an element	header.style.color = ‘blue’;
`document.createElement(type)`	Creates new DOM element of specified type	var item = document. createElement(“li”);
`element.appendChild(child_name)`	Adds child element nested within parent	listy.appendChild(item);

Conditionals

- Use `===` for strict equality (must be same type) (preferred)
- Use `==` for normal equality (can be different type, must be same value)
- Example:
  - 3 == ‘3’ is True
  - 3 === ‘3’ is False

```
1   // Setting up variable:
2   let x = 4;
3
4   if (x < 0) {
5       console.log("negative");
6   } else if (x === 0) {
7       console.log("zero");
8   } else {
9       console.log("positive");
10  }
```

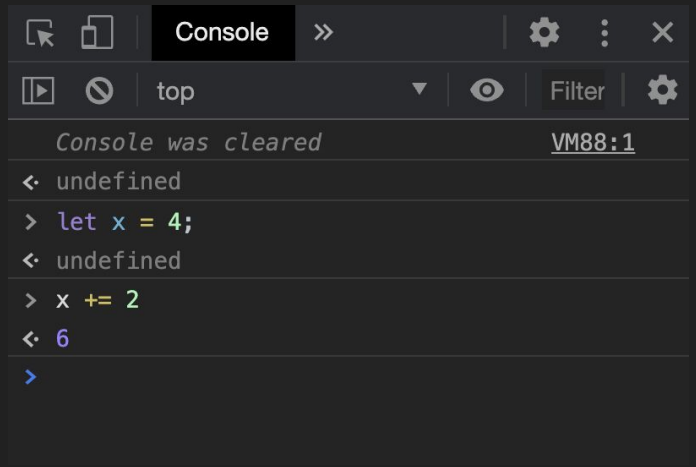
JavaScript Objects

- Stores Key-Value pairs similar to a Python Dictionary

ages

JavaScript Console

- Command-line like tool on web browsers where we can run JavaScript
- Use `console.log(x)` to print x to the console
- In Chrome: Right-Click -> Inspect ->



Timing

- `setInterval(function_name, milliseconds);`
- Ex: `setInterval(count, 1000);`

Local Storage

- Storage within the user’s web browser
- Get item using:
  `localStorage.getItem(“key_name”);`
- Set item using:
  `localStorage.setItem(“key_name”, value);`

Application Programming Interfaces (APIs)

- A set of functions and values that clients can access through requests.
- Data typically transferred as a JSON (JavaScript Object Notation) object
- fetch function makes a web request and returns an HTTP response:

JS api.js > getQuestion

```
1  getQuestion = () => {
2      const url = "https://jservice.io/api/random";
3      fetch(url).then(response => {
4          return response.json();
5      }).then(data => {
6          console.log(data);
7      })
8  }
```

Handle Form Submission

- Use the “submit” event to add a function to be run on submission of a form: `object.addEventListener(“submit”, event => myFunction(event));`
- Use the “value” attribute of each HTML element to access user input: `let f_name = document.querySelector(“#first”).value;`
- Use the `event.preventDefault` function in your function to prevent the default action of the form. (Works with any other event as well if using `addEventListener` function). `event.preventDefault();`
- Note: `return false;` was used in lecture in place of `event.preventDefault`, but that can be problematic at times.

Style

Major Style/Design Issues

- Keep indentation consistent
- camelCase typically used instead of snake_case
- Keep spacing consistent
- Don’t repeat code!!!
- Use separate functions when it improves readability

Questions?

Demo

Javascript

Demo

AJAX

Demo

Project

Project 3

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

Project 3

- Refactor functions to:
  - add email to the mailbox (render email: create element and append)
  - archive current email (and go back to inbox)
  - compose reply email -> compose email and plug in some values
  - view single email
  - send email
- Styling the inbox entry:
  - CSS classes
  - for timestamp
  - CSS child selector

Project 3

- Reading mailbox (inbox/sent/archive)
  - on POST email add ***event.preventDefault();***
  - Style your ‘read’ emails, e.g.:

```
if (email.read) {
    row.classList.add('email-read');
}
```

  - Add event listeners to each email:

```
email.addEventListener('click', function() {
    view_email(email.id);
});
```

  - When loading mailbox make sure to hide the single view as well

Project 3

- Showing single email:
  - You can display button depending on which mailbox you are observing (or by fetching info)
  - Changing visibility → myArchiveElement.*style.display = if globMailbox === ‘inbox’ ? 'inline-block' : 'none';*
  - When showing single email, flush all values before fetching, in case no data or partial data returned
  - Make sure to mark email as read when landing on the page
  - Breaking `\n` with
:
    - email.body.split('\n').forEach(line => {
      1) insert line
      2) insert
      }

Project 3

- Showing single email HTML:
  - separate entries for : From/To/Subject/ Timestamp
  - buttons: reply/(archive|unarchive)
  - Also the div for body

Project 3

- Archiving
  - Simple method that calls backend:
    - you can keep global value of the current email to be always available, so all to pass is True/False
    - No global value you need to pass T/F and email id
- Reply:
  - If subject.slice(0, 4) !== 'Re: ':
    subject += 'Re: '
  - ……element.value= `\n\n\n<hr>On ${email.timestamp} ${email.sender} wrote:\n${curEmail.body}`;

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

Chrome Developer Tools (Network)

In Chrome:

1. Right click
2. Inspect
3. → Demo

Extremely powerful! Let’s try...

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: **ORANGE**

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 4

Ref: Lectures 5 (JavaScript)

Vlad Popil

Jul 15, 2020