Programming web-based educational media

Matthew X. Curinga

Antonios Saravanos

Robby Lucia

Table of Contents

**EDT 603 *Programming web-based educational media*, Spring 2022**

<!DOCTYPE html>  
<html>  
 <body>  
 <blockquote>  
 Anyone who has lost track of time when  
 using a computer knows the propensity  
 to dream, the urge to make dreams come  
 true and the tendency to miss  
 lunch.<br>  
 <strong>Tim Berners-Lee,</strong>  
 <em>inventor of the world wide web</em>  
 </blockquote>  
 </body>  
</html>

**Description:** In this course students learn techniques of web programming to develop interactive, educational media. Using the Javascript (React) programming language and related web development technologies (HTML5, CSS, SVG) for interactive front-end programming, and Python (Flask) for backend server design, students gain practice in the programming and design of interactive software.

**Key words:** computer science, web development, mobile web, interaction design, html, html5, css, javascript, OOP, mobile first, React, python, Flask, REST

**Course website:** <https://canvas.instructure.com/courses/1519530>

# Office Hours

**Matt Curinga**

* Monday, 11-1:00PM
* Wednesday, 2:30-4:30PM
* Thursday, 3-5PM
* *Dr. Curinga’s in person office hours are in the MIXI offices, room 274 of the Addelphi Manhattan Center.* *Office hours by appointment.*

# Course Communications

Participants in this course must actively participate in our suite of online communications tools, including Slack (<https://auedtech.slack.com>), Adelphi email, and the course website.

You *must* check your Adelphi email and the #code channel on Slack at least once a day. It is highly recommended that you install the Slack mobile client and an email client on your mobile phone so that you receive “push notifications” of course announcements. You *must* install the Slack desktop client to facilitate sharing code and screenshots from your development environment.

The best place to post general course questions and any content-related questions is the #code Slack channel. The instructor as well as other students and alums monitor this channel and often provide immediate support.

You are encouraged to contact the instructor at any time via email ([mcuringa@adelphi.edu](mailto:mcuringa@adelphi.edu)) or direct message on Slack to @mxc.

# Goals and Objectives

This course builds on CSC 602 to move beyond basic programming concepts; students will gain expertise in building more complex computer programs, over several iterations. At the end of the course, students will be able to design educationally sound web-based learning media, solve moderately complex problems using object oriented and functional programming paradigms, and collaborate on team programming projects. This course focuses on the design of multi-tier, networked software applications.

Specific teaching and learning goals include:

* designing web-based interactions and multimedia to support learning
* coding effective user interfaces for learning
* implementing Universal Design goals for accessible web sites
* identifying effective methods for teaching more advanced programming concepts and web design skills

Specific software development goals include:

* modeling real world problems with software
* iterative software development
* testing and debugging
* Object oriented programming concepts:
  + Abstraction
  + Encapsulation
  + Objects & Classes
  + Composition
  + Inheritance
  + Polymorphism

# Required Software

* [Atom text editor](https://atom.io/)
* [Slack](http://slack.com) (recommend desktop and mobile clients)
  + [AU Ed Tech #code](https://auedtech.slack.com/signup)
* [Firefox web browser](https://www.mozilla.org/en-US/firefox/new/)
* Chrome or [Chromium](https://www.chromium.org/Home) web browser
* [Python 3](https://www.python.org/downloads/)
* [Jupyter Notebook](https://jupyter.org/install.html)
* [Github Client](https://desktop.github.com/) [also create a [github](https://github.com) account]
* [Postman API Client](https://www.postman.com/downloads/) [optional]

# Online Documentation

### Javascript & HTML

* [Mozilla Developer Network Javascript Docs](https://developer.mozilla.org/en-US/docs/Web/javascript)
* [Mozilla Developer Network HTML Docs](https://developer.mozilla.org/en-US/docs/Web/HTML)
* [Mozilla Developer Network CSS Docs](https://developer.mozilla.org/en-US/docs/Web/CSS)
* [React Framework](https://reactjs.org/docs/getting-started.html)
* [BULMA (CSS framework)](https://bulma.io/documentation/)

### Python

* [Python Standard Library](https://docs.python.org/3/library/index.html)
* [PANDAS](https://pandas.pydata.org/docs/getting_started/index.html#getting-started)
* [Jupyter](https://jupyter.org/documentation)
* [Flask](https://flask.palletsprojects.com/en/2.0.x/)

# Recommended Books

Curinga, M. Peter Wentworth, P., Elkner, J., Downey, A, and Meyers, C. (2018). [Think Javascript](https://mcuringa.github.io/think-js/). [free open textbook]

Downey, A. B. (2016). [*Think Python: How to Think Like a Computer Scientist, Version 2.4.0*](http://greenteapress.com/thinkpython2/html/index.html). Green Tea Press.

Duckett, J. T. (2011). [*Html & css: design and build websites*](http://htmlandcssbook.com/). Indianapolis, IN: Wiley Pubishing, Inc.

# Assignments & Grading

There are 4 graded assignments for this course. From week to week there may be other required, non-graded assignments posted on the course website.

|  |  |
| --- | --- |
| Assignment | Points |
| Multimedia Resume | 25 |
| School Data Analysis | 30 |
| Web Application | 35 |
| Self Evaluation | 10 |

### Self-evaluation (10 points)

At the beginning of week 9, you will complete a self-evaluation. During your one-on-one meeting with the instructor, you will discuss your self-evaluation in order to make sure you get the most out of the remainder of the class.

### Software Project Evaluation Rubric

*This marking guide will be used to evaluate the three software development assignments required for this course.*

1. **React/Javascript**
   * poor: the code runs with errors, is incomplete, or a very close copy of the example project
   * satisfactory: code is organized into functions and uses parameters, code is well organized and well styled, can be improved by writing more general/reusable functions and parameters, being more flexible, or using Javascript idioms efficiently and correctly
   * excellent: code is well thought out and reusable functions create core parts of the site, functions are organized so that changes and new features can be easily implemented, code meets our style guides and clear/accurate names are given to all identifiers (variables, functions). It is clear that the program goes beyond the example project.
2. **Data modeling**
   * poor: most content is hard-coded in the Javascript code, model is an exact copy of example, and/or model does not support the goals of the site
   * satisfactory: data model supports the goals of the specific site, but may not be flexible enough for a different presentation or to handle new data
   * excellent: data model supports the site, and can support other uses without modification to the model, new content can be easily accommodated
3. **HTML**
   * poor: content is presented with little structure, or different HTML tags would better describe the content
   * satisfactory: there is a good fit between the tags used and the data they contain
   * excellent: there is a good fit between the tags used and the data they contain *and* the code correctly and appropriately uses tags that were not part of the example code
4. **CSS**
   * poor: little or no styles, styles are defined that are not used or do not take effect because of errors
   * satisfactory: code demonstrates understanding of box-model, color, typography, and/or images
   * excellent: advanced layout are used to interesting effect (flex, grids, etc), code exceeds example projects
5. **User Experience**
   * poor: site is hard to read, understand, and use; labels and text are not well edited, items are not clearly organized, essential information is missing, and/or the design significantly impedes the usability of the site
   * satisfactory: site is easy to use and information is well organized, presentation is clear and clean
   * excellent: code, layout, and css combine to create an interesting effect, site has a high quality, professional feel
6. **Risk Taking**
   * poor: the project narrowly adapts worked examples and sample code; no evidence of attempts to incorporate ideas not strictly covered in class
   * satisfactory: the project attempts either a novel idea or to integrate sophisticated programming approaches beyond what is covered in class; it’s evident that student is able to read technical documentation and adapt ideas for new purposes. some aspects of the program may not be fully functional or integrated with the rest of the program.
   * excellent: incorporates novel ideas and techniques, as in “satisfactory,” but demonstrates a high level of success with the code and integration.

### Multimedia Resume (25 points)

This first assignment is designed to get you up and running with the key technologies we will use this semester. You will create a data-driven web page as your own online resume or portfolio.

For this project, you will:

1. Keep all of the *data* for your resume in .json data files. These files will contain the lists of your work experience, education background, technical skills, and other relevant information.
2. Code Javascript source files using the React framework and HTML5 to structure your web page. Use React component functions to create a program that is modular and easy to change. Your final project must have (at least) two different “looks” which can be swapped live. Your resume program must be separated into multiple source files (.js or .jsx) which are imported into App.js to compose the site.
3. Create or find any media assets (images, audio, video, etc) necessary for your resume. Your site *must* contain some images.
4. Code CSS files to achieve your desired aesthetic and usability goals.

*The resume is an individual project*

### Mini-app (30 points)

The mini app is a data driven web application. It is “mini” because it has a constrained, singular focus. This application stands alone – it does not need any networked resources. It goes beyond the resume project, though, in that it’s *interactive*. Buttons, text boxes, and other form elements enable the user to alter data and the way the app functions. Data created is either stored locally (in the user’s client) or only exists during the session.

Successful projects will use React *state* and React *life cycle functions* to achieve their results.

*This mini-app is an individual project*

### School Data Analysis (35 points)

For the school data analysis you will become expert in one of the open data sets that are available to us regarding school data. These data sets include school performance, demographics, location, and other interesting data. This live data is often not “cleaned” – meaning that some data is missing, some of it is inconsistent with expected data, and some values may not be understood intuitively. Further, the documentation accompanying this data is often sparse or non-existent. Raw test scores will be included without any details of how tests are scored, the range of values, etc. Lastly, the data is most useful when it can be connected and compared to *other* data sets. However “keys” can be used inconsistently (school names, address fields, identifiers).

For this project, you will write a program that *cleans* the data and outputs it in useful formats. You will study the data *and* use your research skills to understand what the data means. For example, students are identified as current\_ELL, ever\_ELL or never\_ELL to indicated if they were still learning English while in the school system. Your job would be to provide some qualitative information about how and when students are placed in and out of these categories. Lastly, you will explore the data and build a series of graphical reports with descriptive statistics that will make it easier for other researchers and programmers to work with these data sets. For example, you will want to provide ranges, and averages (mode, mean, median) for numerical data. For categorical data, you will want to enumerate and describe each of the categories. If you had to make assumptions in “cleaning” your data, you will detail the process and describe any potential problems. You will consider all of the unique identifiers in your data and describe how they might be used to connect your data set to other sets of data.

Once you have a strong understanding of your data, you will create a sample report where you analyze one aspect of the data and present it as a web page that contains both quantitative output as well as written analysis.

This project will use Python, PANDAS, and Jupyter Notebooks to clean and analyze the data. Jupyter will output any necessary files (.json, SVG images, etc) that can be displayed on the web. You will use React/Javascript to build a mini website that describes your data and hosts your report.

*This is a “team” project that you will work on with one other member of the class.*

### Web Application

The web application will build on the work of the Data Analysis project. In this project we will build interactive tools where users can click buttons, fill out forms, drag-and-drop, and interact with our program in various ways. The “front-end” will be programmed in Javascript with React, and the live backend will be written with Python running in the Flask server. We will test and prototype the Python code in Jupyter Notebook, but it will run live from Flask.

I will encourage everyone to continue working with open school data, so that we have a host of data interactions by the end of the term. However, if you have your own interesting projects, you will have the opportunity to pursue them instead.

*This web application is a group project. All team members will receive the same grade.*

# Books and online resources

### Documentation & Reference websites

* [Mozilla Developer Network](https://developer.mozilla.org/en-US/)
* [React](https://reactjs.org/docs/getting-started.html)
* [Bootstrap](https://getbootstrap.com/docs/5.1/getting-started/introduction/)
* [Python Official Docs](https://docs.python.org/3/index.html)
* [Flask](https://flask.palletsprojects.com/en/2.0.x/)
* [PANDAS](https://pandas.pydata.org/docs/)
* [World Wide Web Consortium](http://w3.org)
* [W3 Schools](http://www.w3schools.com/)
* [Regular Expressions](http://www.regexr.com/)
* [LinkedIn Learning *via eCampus*](https://portal.adelphi.edu/services/redirect.php?service_id=84)

### Books

* [HTML and CSS: Design and Build Websites](http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118008189.html), our textbook
* [JavaScript & jQuery: Interactive Front-End Web Development Hardcover](http://www.wiley.com/WileyCDA/WileyTitle/productCd-1118871650.html), also J. Duckett, same series
* [Dive into HTML 5](http://diveintohtml5.info/) [free online]
* [The Elements of Typographic Style Applied to the Web](http://webtypography.net/toc/) [free online]
* [Mastering Regular Expressions](http://shop.oreilly.com/product/9780596528126.do)

### Python, PANDAS, and Data science Tutorials

* [LinkedIn Learning:: Python Essential Training](https://www.linkedin.com/learning/python-essential-training-2018/welcome) [4h 37m]
* [LinkedIn Learning:: Pandas Essential Training](https://www.linkedin.com/learning/pandas-essential-training) [2h 14m]

### Javascript, HTML, and CSS

* [Khan Academy::Intro to JavaScript: Drawing & Animation](https://www.khanacademy.org/computing/computer-programming/programming)
* [Khan Academy::Intro to HTML/CSS: Making webpages](https://www.khanacademy.org/computing/computer-programming/html-css)
* [Code Academcy](http://www.codecademy.com/)
* [P2PU School of webcraft](https://p2pu.org/en/schools/school-of-webcraft/)
* [Treehouse](http://teamtreehouse.com/) [paid]
* [Thinkful](http://www.thinkful.com/)
* [GeekCamp::HTML5 Tutorial](http://www.geekchamp.com/html5-tutorials/1-html5-overview)
* [SkilledUp::Learn Web Design](http://www.skilledup.com/learn-web-design-guide/)

### Design, accessibility, UX

* [A List Apart](http://alistapart.com/topic/html)
* [Smashing Magazine](http://www.smashingmagazine.com/)
* [Adobe Kuler](https://color.adobe.com/create/color-wheel/)
* [Nielsen/Norman Group](http://www.nngroup.com/articles/)
* [United States Section 508](http://en.wikipedia.org/wiki/Section_508_Amendment_to_the_Rehabilitation_Act_of_1973)
  + <https://www.section508.gov/>
  + <http://webaim.org/standards/508/checklist>
* [Usability.gov](http://www.usability.gov/index.html)
* [Research-Based Web Design & Usability Guidelines](http://www.usability.gov/guidelines/guidelines_book.pdf)
* [hex/html color chart](http://www.december.com/html/spec/color.html)

### Online Tools

* [w3c HTML Validation Service](http://validator.w3.org/#validate_by_uri+with_options)
* [w3c CSS Validation Service](http://jigsaw.w3.org/css-validator/)

### Media Resources

* [Creative Commons Search](http://search.creativecommons.org/), for images, music, etc
* [Wikimedia Commons](http://commons.wikimedia.org/wiki/Main_Page), images and other media (including stuff from Wikipedia), curated
* [Open Clip Art](https://openclipart.org/), free vector graphics
* [Creative Commons Music](http://creativecommons.org/music-communities)
* [Fossil Bank](http://fossilbank.wikidot.com/)
* [Colour Lovers Palettes](http://www.colourlovers.com/)
* [Google Fonts](https://fonts.google.com/)