CS50's Web Programming with Python and JavaScript

OpenCourseWare

Brian Yu
brian@cs.harvard.edu

David J. Malan
malan@harvard.edu

CS50 Certificate

Ed Discussion for Q&A

Quick Start Guide

---

edX
iTunes U
YouTube

Discord
Facebook Group
Facebook Page

Git

Git and GitHub

- Git can keep track of changes made to code, synchronize code between different people, test changes to code without losing the original, and revert back to old versions of code.
- GitHub is a website that stores Git repositories on the internet to facilitate the collaboration that Git allows for. A repository is simply a place to keep track of code and all the changes to code.
- Git commands:
  - `git clone <url>` : take a repository stored on a server (like GitHub) and downloads it
  - `git add <filename(s)>` : add files to the staging area to be included in the next commit
  - `git commit -m "message"` : take a snapshot of the repository and save it with a message about the changes

    - `git commit -am <filename(s)> "message"` : add files and commit changes all in one
    - `git status` : print what is currently going on with the repository
    - `git push` : push any local changes (commits) to a remote server
    - `git pull` : pull any remote changes from a remote server to a local computer
    - `git log` : print a history of all the commits that have been made
    - `git reflog` : print a list of all the different references to commits
    - `git reset --hard <commit>` : reset the repository to a given commit
    - `git reset --hard origin/master` : reset the repository to its original state (e.g. the version cloned from GitHub)
- When combining different versions of code, e.g. using `git pull`, a merge conflict can occur if the different versions have different data in the same location. Git will try to take care of merging automatically, but if two users edit, for example, the same line, a merge conflict will have to be manually resolved.
    - To resolve a merge conflict, simply locally remove all lines and code that are not wanted and push the results.

HTML

- HTML (HyperText Markup Language) is used to lay out the structure of a webpage.
- `<!DOCTYPE html>` is placed at the start of an HTML file to indicate to the browser that HTML5 is being used.
- HTML is made up of tags. Tags generally come in pairs, with data being in between the tags. Generally, tags are indented to help visualize their hierarchy, but any indentation is purely stylistic. Tags can also have attributes, which are data fields, sometimes required and sometimes optional, that provide additional information to the browser about how to render the data.
- Some common HTML tags:
    - `<html></html>` : contents of website
    - `<head></head>` : metadata about the page that is useful for the browser when displaying the page
    - `<title></title>` : title of the page
    - `<body></body>` : body of the page
    - `<h1></h1>` : header (h1 is the largest header, h6 is the smallest header)
    - `<ul></ul>` : unordered list
    - `<ol></ol>` : ordered list
        - `<li></li>` : list item (must be inside either `<ul></ul>` or `<ol></ol>`)
    - `<img src="path/to/img.jpg" height="200" width="300">` : image stored at `src` attribute, whcih can also be a URL

        - note that this is a single tag without an end tag
        - both `height` and `width` are optional (if one is omitted, the browser will auto-size the image), and can also take a percentage: `height=50%` to automatically scale the image to a certain portion of the page
    - `<table></table>` : table
        - `<th></th>` : table header
        - `<tr></tr>` : table row
        - `<td></td>` : table data (cell)
    - `<form></form>` : form that can be filled out and submitted by the user
        - `<input type="text" placeholder="Full Name" name="name">` : input field
            - `type` indicates the type of data
            - `placeholder` is the greyed-out text shown before the field is filled
            - `name` is an identifier for the input field
        - `<button></button>` : button used to submit form
- The Document Object Model is a way to conceptualize webpages by representing them as a interconnected hierarchy of nodes. In HTML, the nodes of the DOM would be the different tags and their contained data, with the `<html></html>` tag being at the very top of the tree. While this might seem a little unintuitive at first, this model will become very useful in the future.

CSS

- CSS (Cascading Style Sheets) is a language used to interact with and style HTML, changing the way it looks according to a series of rules. CSS is what makes websites look nice.
- CSS can be applied to HTML in a variety of ways:
    - The `style` attribute: `<h5 style="color:blue;text-align:center;"></h5>`
        - The semicolon-separated CSS properties passed to `style` will be applied to whatever the tag containts.
    - The `<style></style>` tags: ```html

    ``` * This is a useful paradigm to use when reusing the same styling many times throughout a page. The properties listed will apply to all of the tags that are listed.

    - A separate `.css` file: add `<link rel="stylesheet" href="path/to/styles.css">` to the header, and move the CSS code (same format as used inside the `<style></style>` tags.
        - This is often an even better paradigm because it separates two distinctly different things: structure (HTML) and style (CSS), while also being easier to manage and read.
- Some common CSS properties (those that take arguments in pixels often can take a percentage or simply `auto`
    ```

`auto`):

- `color: blue`, `color: #0c8e05` : can be 1 of ~140 named colors, or a hexadecimal value that represents an RGB value
- `text-align: left` : aligns text to `left`; other possible arguments are `center`, `right`, or `justify`
- `background-color: teal` : sets the background to a color, which is the same format as the `color` property
- `height: 150px` : sets the height of an area
- `width: 150px` : sets the width of an area
- `margin: 30px` : sets the margin around all four sides of an area
  - can also be broken up into `margin-left`, `margin-right`, `margin-top`, and `margin-bottom`
- `padding: 20px` : sets the padding around text inside an area
  - can be broken up the same way as `margin`
- `font-family: Arial, sans-serif` : sets the font family to be used
  - a comma-separated list provides alternatives in case a browser doesn't support a specific font
  - generic families such as `sans-serif` will use browser defaults
- `font-size: 28px` : sets the font size
- `font-weight: bold`: sets the font weight to quality, a relative measure (`lighter`), or a number (`200`)
- `border: 3px solid blue`: sets a border around an area

- There are lots of CSS properties that can be used in a lot of different ways. Check out the wonderfully extensive documentation for more information.

Sectioning with HTML and CSS

- Two special tags allow us to break up a our webpage into sections:
  - `<div></div>` : vertical division of a webpage
  - `<span></span>` : section of a webpage inside, for example, text
- Both `<div></div>` and `<span></span>` don't really do much by themselves, but they allow for the labelling of sections of a webpage..
- Different sections of a webpage can be referenced with the `id` and `class` attributes. `id`s uniquely identify elements, but there can be an arbitary number of elements with a given `class`.
- `id`s can be referenced in CSS with `#id` and `class`es can be referenced with `.class`. `id`s and `class`es, therefore, can be used to style the same types of areas (3 different `<div></div>`, for example) in different ways.

GitHub Pages

- GitHub Pages is a feature of GitHub which allows for a repository to be deployed to the internet.
- Simply scroll to **GitHub Pages** under **Settings**, select the **master branch**, and click **save**.
- By default, the repository will be deployed to username.github.io/repository.
- GitHub Pages is automatically updated when the repository is updated.