Introduction to CS106AX

Jerry Cain
CS106AX
September 27, 2023
slides courtesy of Eric Roberts

Course Description

CS106AX: Programming Methodologies in JavaScript and Python

Introduction to the engineering of computer and web applications emphasizing modern software engineering principles: object-oriented design, decomposition, encapsulation, abstraction, and testing. This course targets an audience with prior programming experience, and that prior experience is leveraged so material can be covered in greater depth.

Terms: Aut | Units: 3-5 | UG Reqs: WAY-FR | Grading: Letter or CR/NC

CS106AX satisfies the same WAYS requirements as any other CS106A offering.

Why JavaScript?

- When Stanford CS106A adopted Java about 20 years ago, we expected—along with its designers—that it would become the "language of the web". That didn't happen. 😢
- Today, the "language of the web" is JavaScript, which has become the most widely used language in industry.
- Along with JavaScript expert Douglas Crockford, we believe that, provided you avoid some of its most frequently abused features, JavaScript is "a beautiful, elegant, highly expressive language" that is ideal for a first course in programming.
  - It is considerably easier to learn than Java.
  - There are far fewer details to memorize.
  - It offers cleaner implementations of modern language features.
  - It is universally supported on the web.

Why Python?

- Stanford's CS106A course adopted Python as its language of instruction about six years ago.
- The name of the language does not come from the snake that graces the cover of the O'Reilly book but is instead a tribute to *Monty Python's Flying Circus*.

- While JavaScript is required of any nontrivial application written for the browser, it's not quite as popular for traditional, computationally intense programs.
- Python is more general purpose, provides a large set of libraries with no equivalents in standard JavaScript, and more easily integrates with C, C++, and Java libraries so that Python programs can be executed more quickly.

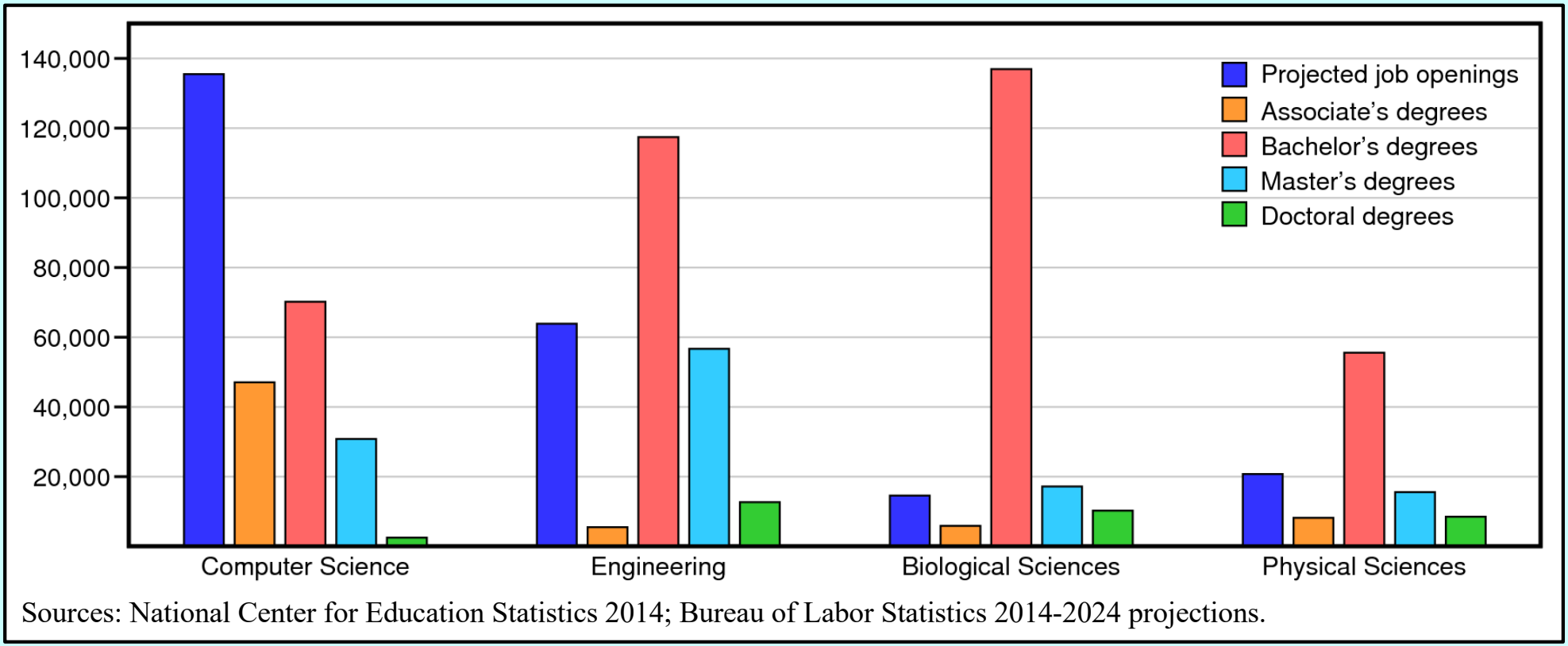
The Web's Client-Server Model

1. The user launches a web browser.
2. The user requests a web page.
3. The browser sends a request for a document.
4. The server sends back a text file containing the payload.
5. The browser interprets the document and renders the page.

Why Both JavaScript and Python

- Modern web pages depend on three related technologies: *HTML* (Hypertext Markup Language), *CSS* (Cascading Style Sheets), and *JavaScript*.
- These tools are used to control different aspects of the page:
  - HTML is used to specify content, structure, and data hierarchy.
  - CSS is used to control appearance and formatting.
  - JavaScript elevates the page to be truly interactive and operate like a traditional desktop application incidentally running in a browser.
- Many web properties—e.g., Pinterest, Netflix, and Spotify—programmatically generate user-specific HTML using Python.
- By learning both languages, you'll understand how web apps—both the client and server endpoints—are implemented.

Why Study Computer Science?



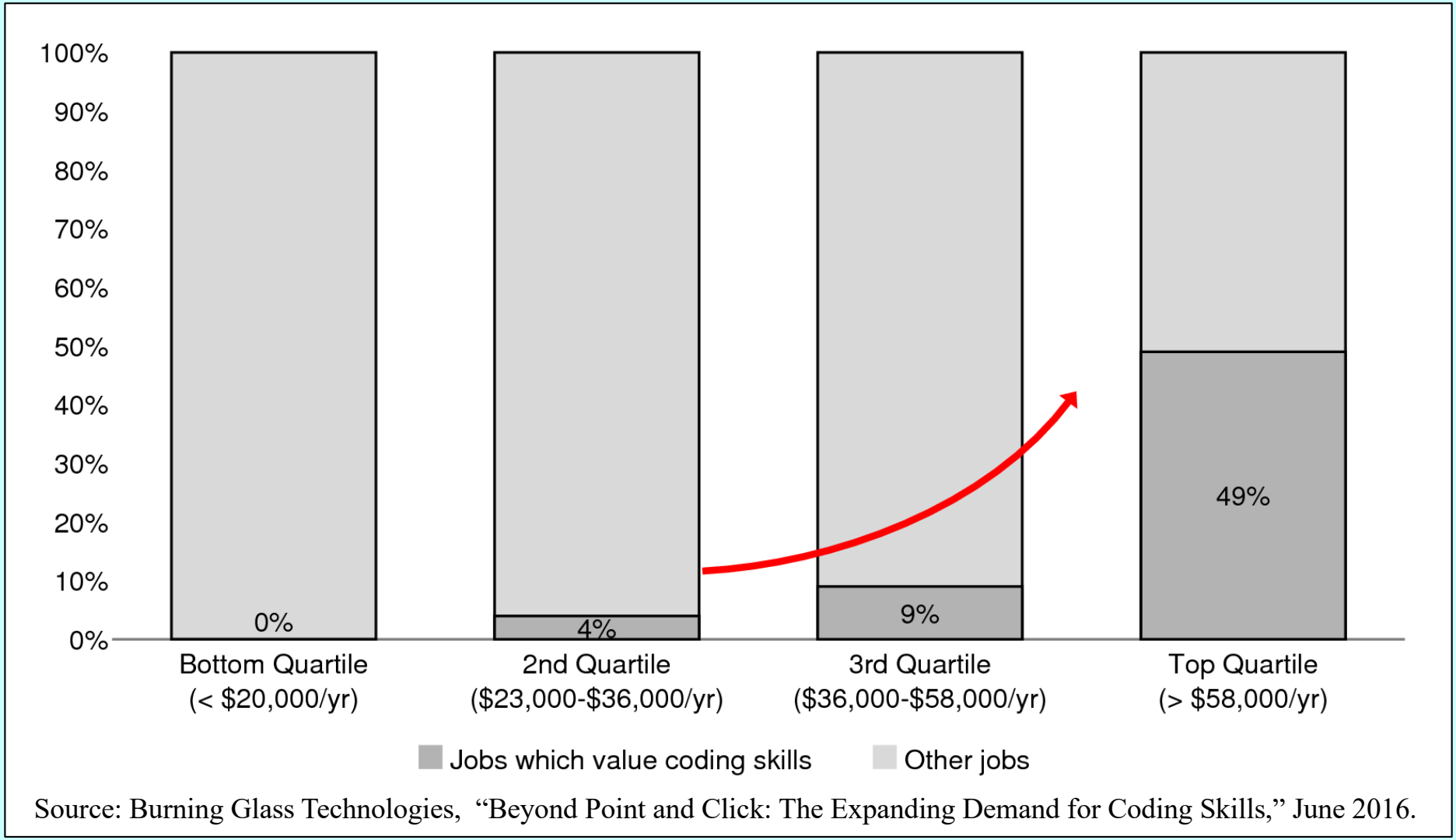
Sources: National Center for Education Statistics 2014; Bureau of Labor Statistics 2014-2024 projections.

We are very happy with the students that we get from this university. . . . We just wish we could hire two to three times as many of them.

— Bill Gates at Stanford, February 19, 2008

Everyone Benefits from Programming

Half of all jobs in the top income quartile value coding skills.



Source: Burning Glass Technologies, "Beyond Point and Click: The Expanding Demand for Coding Skills," June 2016.

CS106AX Course Staff

Jerry Cain
jerry@cs.stanford.edu
Office Hours:
Mondays (Durand 319): 2:00 – 4:00 P.M.
Tuesdays (online): 2:00 – 4:00 P.M.

Avi Gupta
agupta07@stanford.edu
Office Hours (Durand 303):
TBD

Syllabus—Week 1

September 25	27	29
observing Yom Kipper *no lecture*	Introductions and syllabus Why two languages? Simple JavaScript programs	Basic Functions Control Idioms Decomposition **Textbook: Chapters 2 - 5**

Syllabus—Week 2

October 2	4	6
CS106AX Graphics `GObject` hierarchy	Event-driven programming Functions as data Closures Mouse events	Event-driven programming Advanced closures Images
Assign1 Out **Textbook: Chapters 2 – 5**	**Textbook: Sections 6.1 – 6.4**	**Textbook: Sections 6.1 – 6.4**

Syllabus—Week 3

9	11	13
One-shot functions Timer functions Animations	Simple strings String methods String algorithms	Simple arrays Array methods Array algorithms
Assign1 In, Assign2 Out **Textbook: Sections 6.5 - 6.6**	**Textbook: Sections 7.1 – 7.5**	**Textbook: Sections 8.1 – 8.6**

Syllabus—Week 4

16	18	20
Simple aggregate objects Object construction JSON and **`eval`**	Binary representation ASCII and Unicode Rep-sensitive algorithms	Cryptography Encryption techniques The Enigma Machine
Assign2 In, Assign3 Out **Textbook: Sections 9.1 - 9.2**	**Textbook: Sections 7.1 – 7.5**	**Textbook: Section 7.6**

Syllabus—Week 5

October 23	25	27
JavaScript wrap	Simple Python programs Control idioms Modules and **`import`**	Python Strings Slices
Assign 3 In, Assign 4 Out	**Reader: Chapters 1 - 4**	**Reader: Chapter 6**

Syllabus—Week 6

October 30	November 1	3
Python lists List methods	Python dictionaries Large data sets Midterm Exam: Wednesday, November 1 3:45 P.M. or 7:00 P.M	Introduction to OO Design Designing simple objects
Assign4 In **Textbook: Chapter 7**	**Reader: Section 11.1**	**Assign5 Out** **Reader: Sections 9.1 – 9.3**

Syllabus—Week 7

November 6	8	10
Advanced OO design Constructors Internal representations	Data-driven design The Teaching Machine	Overview of Adventure!
Reader: Sections 12.1 – 12.3	**Reader: Section 12.4**	**Assign5 In, Assign6 Out**

Syllabus—Week 8

November 13	15	17
HTML and the DOM Interactors Native events	Introduction to CSS Selectors, Classes, Rules Inline Styling	Client-Server Paradigm Life of an HTML page Basic HTTP
Textbook: Sections 12.1 – 12.2	**Textbook: Sections 12.3 – 12.4**	**Assign6 In, Assign7 Out**

Syllabus—Week 9

November 27	29	December 1
Async programming Payload Types APIs	Accessing the DOM in JavaScript **`document`** and **`window`**	Web Programming Wrap **Assign7 In, Assign8 Out**

Syllabus—Week 10 and Beyond

December 4 Guest Speaker: TBD Digital Identity	6 Guest Speaker: TBD Computing and Ethics Has the Internet failed Us?	8 Life After CS106AX **Assign8 In**

Review Session (tentatively):
Sunday, December 10
12:00 P.M.

Final Exam:
Monday, December 11
8:30 A.M.

Assignments in CS106AX

- Assignments in CS106AX are due at 5:00P.M. Assignments that come in after 5:00P.M will be considered late.
- Everyone in CS106AX starts the quarter with three "late days" that you can use at any point you need some extra time. In my courses, late days correspond to class meetings, so that, if an assignment is due on Monday and you turn it in on Wednesday, that counts as ***one*** late day.
- Extensions can only be approved by the TA, Avi Gupta.
- Assignments are graded by your section leader, who discusses your work in an interactive, one-on-one grading session.
- Each assignment is given two grades: one for functionality and one for programming style. Style matters. Companies in Silicon Valley expect Stanford graduates to understand how to write code that other programmers can read and maintain.

The CS106AX Grading Scale

- Functionality and style grades for the assignments use the following scale:

++	A submission so good it "makes you weep".
+	Exceeds requirements in nontrivial ways.
✓+	Satisfies all requirements of the assignment.
✓	Meets most requirements, but with some problems.
✓−	Some more serious problems.
−	Even worse than that.
−−	Represents unsatisfactory work.

Contests

- CS106AX will have two contests as follows:
  - The Graphics Contest associated with Assignment #2
  - The Adventure Contest associated with Assignment #6

- The grand prize in the contest is a score of 100% on one of the graded components of the course (which in practice is almost always the final exam).
- As an additional incentive, entering any of the contests gives you a virtual ticket to win an additional grand prize in an end-of-quarter lottery. As does receiving a runner-up or honorable mention on a contest and finding errors and reporting in the textbooks.

Honor Code Rules

Rule 1: You must not look at solutions or program code that is not your own. This includes anything AI-generated.

Rule 2: You must not share your assignment solution code with other students.

Rule 3: You must indicate on your submission any assistance you received, no matter how small.

The Structure of an HTML File

- An HTML file consists of the text to be displayed on the page, interspersed with various commands enclosed in angle brackets, which are known as *tags*.
- HTML tags generally occur in pairs. The opening tag begins with the name of the tag. The corresponding closing tag has the same name preceded by a slash. The effect of the tag applies to everything between the opening and closing tags.
- The only HTML tags you will need before Week 7 of the course course appear in the template on the next page, which describes the structure of the HTML index file (and by convention is called **`index.html`**.)

Standard `index.html` Pattern

- The following components of **`index.html`** are standardized:
  - Every file begins with a **`<!DOCTYPE html>`** tag.
  - The entire content is enclosed in **`<html>`** and **`</html>`** tags.
  - The file begins with a **`<head>`** section that specifies the title and JavaScript files to load.
  - The file includes a **`<body>`** section that specifies the page.

```
<!DOCTYPE html>
<html>
  <head>
    <title>title of the page</title>
    One or more script tags to load JavaScript code.
  </head>
  <body onload="function()">
    Contents of the page, if any.
  </body>
</html>
```

Creating the JavaScript Program File

- The first step in running a JavaScript program is creating a file that contains the definitions of the functions, along with comments that give human readers a better understanding of what the program does.
- Here, for example, is the complete **`HelloWorld.js`** file:

```
/*
 * File: HelloWorld.js
 * -------------------
 * This program displays "hello, world" on the console.  It
 * is inspired by the first program in Brian Kernighan and
 * Dennis Ritchie's classic book, The C Programming Language.
 */

function HelloWorld() {
   console.log("hello, world");
}
```

Creating the HTML File (Version 1)

- A simple HTML file that loads the **`HelloWorld.js`** program looks like this:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hello World</title>
  <script type="text/javascript" src="HelloWorld.js"></script>
 </head>
 <body onload="HelloWorld()"></body>
</html>
```

- This file asks the browser to load the file **`HelloWorld.js`** and then call the function **`HelloWorld`** once the page is loaded.
- The problem with this strategy is that it is hard "to find out where your output went" as Kernighan and Ritchie mention.

Creating the HTML File (Version 2)

- The output from the console log appears in different places in different browsers and usually requires the user make an effort to search for and find the output.
- To make the console log easier to find, we provide a library called **`JSConsole.js`** that redirects the console log to a much more visible region of the web page.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Hello World</title>
   <script type="text/javascript" src="JSConsole.js"></script>
   <script type="text/javascript" src="HelloWorld.js"></script>
 </head>
 <body onload="HelloWorld()"></body>
</html>
```

The End