Introduction to CS106AX

Jerry Cain
CS106AX
September 26, 2022
slides courtesy of Eric Roberts
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 a little over 15 years ago, we expected—along with its designers—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 five 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 a language 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: \textit{HTML} (Hypertext Markup Language), \textit{CSS} (Cascading Style Sheets), and \textit{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?

---

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.

CS106AX Course Staff

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

Jonathan Kula
dkula@stanford.edu
Office Hours (Durand 303):
  TBD
<table>
<thead>
<tr>
<th>September 26</th>
<th>28</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introductions and syllabus</td>
<td>Basic Functions</td>
<td>CS106AX Graphics</td>
</tr>
<tr>
<td>Why two languages?</td>
<td>Control Idioms</td>
<td><em>GObject</em> hierarchy</td>
</tr>
<tr>
<td>Simple JavaScript programs</td>
<td>Decomposition</td>
<td></td>
</tr>
<tr>
<td><strong>Textbook: Chapters 2 - 5</strong></td>
<td><strong>Assign1 Out</strong></td>
<td><strong>Textbook: Chapters 2 – 5</strong></td>
</tr>
</tbody>
</table>
# Syllabus—Week 2

<table>
<thead>
<tr>
<th>October 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event-driven programming</td>
</tr>
<tr>
<td>Functions as data</td>
</tr>
<tr>
<td>Closures</td>
</tr>
<tr>
<td>Mouse events</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event-driven programming</td>
</tr>
<tr>
<td>Advanced closures</td>
</tr>
<tr>
<td>Images</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-shot functions</td>
</tr>
<tr>
<td>Timer functions</td>
</tr>
<tr>
<td>Animations</td>
</tr>
</tbody>
</table>

Textbook: Sections 6.1 – 6.4  
Textbook: Sections 6.1 – 6.4  
Assign1 In, Assign2 Out  
Textbook: Sections 6.5 - 6.6
# Syllabus—Week 3

<table>
<thead>
<tr>
<th>10</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple strings</td>
<td>Simple arrays</td>
<td>Simple aggregate objects</td>
</tr>
<tr>
<td>String methods</td>
<td>Array methods</td>
<td>Object construction</td>
</tr>
<tr>
<td>String algorithms</td>
<td>Array algorithms</td>
<td>JSON and eval</td>
</tr>
</tbody>
</table>

Textbook: Sections 7.1 – 7.5

Textbook: Sections 8.1 – 8.6

Assign2 In, Assign3 Out

Textbook: Sections 9.1 - 9.2
<table>
<thead>
<tr>
<th>17</th>
<th>19</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary representation</td>
<td>Cryptography</td>
<td>JavaScript wrap</td>
</tr>
<tr>
<td>ASCII and Unicode</td>
<td>Encryption techniques</td>
<td></td>
</tr>
<tr>
<td>Rep-sensitive algorithms</td>
<td>The Enigma Machine</td>
<td></td>
</tr>
</tbody>
</table>

Textbook: Sections 7.1 – 7.5

Textbook: Section 7.6

Assign3 In, Assign4 Out
<table>
<thead>
<tr>
<th>October 24</th>
<th>26</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Python programs</td>
<td>Python Strings</td>
<td>Python lists</td>
</tr>
<tr>
<td>Control idioms</td>
<td>Slices</td>
<td>List methods</td>
</tr>
<tr>
<td>Modules and import</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reader: Chapters 1 - 4

Reader: Chapter 6

Assign4 In Textbook: Chapter 7
## Syllabus—Week 6

<table>
<thead>
<tr>
<th>October 31</th>
<th>November 2</th>
<th>November 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python dictionaries</td>
<td>Introduction to OO Design</td>
<td>Advanced OO design</td>
</tr>
<tr>
<td>Large data sets</td>
<td>Designing simple objects</td>
<td>Constructors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internal representations</td>
</tr>
</tbody>
</table>

**Midterm Exam:**
Tuesday, November 1
3:30 P.M. or 7:00 P.M

**Assign5 Out**
Reader: Sections 9.1 – 9.3

**Reader:**
Section 11.1

**Reader:**
Sections 12.1 – 12.3
# Syllabus—Week 7

<table>
<thead>
<tr>
<th>November 7</th>
<th>9</th>
<th>11</th>
</tr>
</thead>
</table>
| Data-driven design  
The Teaching Machine | Overview of Adventure! | Generators  
Lambdas |

Reader: Section 12.4

Assign5 In, Assign6 Out
# Syllabus—Week 8

<table>
<thead>
<tr>
<th>November 14</th>
<th>16</th>
<th>18</th>
</tr>
</thead>
</table>
| 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

<table>
<thead>
<tr>
<th>November 28</th>
<th>30</th>
<th>December 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload Types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accessing the DOM in JavaScript</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>document</code> and <code>window</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web Programming wrap</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assign7 In, Assign8 Out
# Syllabus—Week 10 and Beyond

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 5</td>
<td>Guest Speaker: TBD Digital Identity</td>
</tr>
<tr>
<td>7</td>
<td>Guest Speaker: TBD Computing and Ethics Has the Internet Failed Us?</td>
</tr>
<tr>
<td>9</td>
<td>Life After CS106AX</td>
</tr>
</tbody>
</table>

**Assign8 In**

- Review Session: Sunday, December 11 12:00 P.M.
- Final Exam (tentatively): Monday, December 12 8:30 A.M.
Assignments in CS106AX

- Assignments in CS106AX are due at 5:00 PM. Assignments that come in after 5:00 PM 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 Wednesday and you turn it in on Friday, that counts as one late day.

- Extensions can only be approved by the TA, Jonathan Kula.

- 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.

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 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.

```html
<!DOCTYPE html>
<html>
<head>
  <title>title of the page</title>
  One or more script tags to load JavaScript code.
</head>
<body onunload="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:

```javascript
/*
 * File: HelloWorld.js
 * -------------------
 * This program displays "hello, world" on the console. It
 * is inspired by the first program in Brian Kernighan and
 */

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:

```html
<!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 advise.
Creating the HTML File (Version 2)

• The output from the console log appears in different places in different browsers and usually requires the user to make an effort to discover 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.

```html
<!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