



Welcome to CS 106S!

Introduction to CS for Social Good, our
map for the quarter, and JavaScript!

cs106s.stanford.edu, Autumn 2024



Welcome to the First Day of Class!

Benjamin Yan, CS106S 2024

Stanford | ENGINEERING
Computer Science



Hi! I'm Ben

- Stanford MS CS coterms '25
- Stanford grad, BS double major in CS & Math '24
- Minored in Creative Writing ❤️; took 11 English classes in undergrad + Oxford CW abroad
- Prev. SWE @ NVIDIA; now more teaching and research oriented

- Taught CS 106S last year Aut. & Spr.; also head TA of CS 106AX this Aut.; TA MATH 51 this Win.
- Interests: fiction/novel writing, anime (*JJK, Demon Slayer, My Hero Academia, Blue Period*), outdoors, boygenius music, bubble tea 





Commercial Break!

If you're planning to take an intro CS course this autumn, consider **CS106AX!**
(accelerated CS106A which satisfies any requirement that 106A does)

The screenshot shows the homepage of the CS106AX course. At the top, there's a header with a tablet icon, the course name "CS106AX: Programming Methodologies in JavaScript and Python", the semester "Autumn 2024", and the Stanford University logo. Below the header is a decorative banner featuring a collage of colorful 3D cubes. The main content area is organized into three columns: "Course Material" with links to "Lectures", "Handouts", and "Set Up PyCharm"; "Resources" with links to "Style Guide", "JSGraphics Docs", and "JavaScript Reader"; and "Getting Help" with links to "Office Hours Calendar" and "Ed Q&A Forum".

cs106ax.stanford.edu (taught by the wonderful Jerry Cain)

I promise I wasn't paid to say this.

Benjamin Yan, CS106S 2024

Stanford | ENGINEERING
Computer Science

Intros!

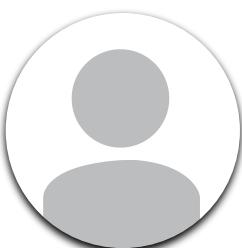
- Name & pronouns if you're comfortable sharing!
- What you're studying / thinking about studying
- Year
- Fun fact 😲 or **any one of the following!**
 - What are you looking forward this autumn / year? 🎄
 - Something you did over the summer 🎈
 - Music / book recommendations? 🎵📚
 - Anything else you'd like to share! :)

Course Staff

Teaching Team



Ben Yan



Sarah Chen



Cooper de
Nicola



Aditya
Saligrama

Faculty Sponsor



Prof. Jerry Cain



Contact: cs106s-aut2425-staff@lists.stanford.edu
or bbyan@stanford.edu



The Map For Today

- ◆ 1 syllabus & logistics
- ◆ 2 getting set up for the class
- ◆ 3 HTML/CSS/javascript basics
- ◆ 4 caesar ciphers!



Course Logistics

- 1 unit, S/NC
- Attendance (8/9*)**
 - Relaxed, workshop-style environment
 - Brief check-off forms**
- Canvas for announcements
- Questions welcome!

***Please do reach out to us if difficult circumstances arise! We understand life can be very stressful and challenging, and will always create a path for you to pass 106S.**



Course Website!

cs106s.stanford.edu



Contact Email

cs106s-aut2425-staff@lists.stanford.edu;
bbyan@stanford.edu is equally fine!



Place & Time

Lathrop 190

Thurs, 4:30 - 6:20 PM; usually try to keep class to 90 minutes-ish

Course Schedule

Week 1 Sep 26 Intro, JavaScript, Ciphers

Week 2 Oct 3 Sentiment Analysis & Refugee Tweets

Week 3 Oct 10 CS for Climate Change

Week 4 Oct 17 KNN for Cancer Detection

Week 5 Oct 24 Cybersecurity and Ethical Hacking

Week 6 Oct 31 Web Deployment & Open Source

Tuesday, Nov 5, Election Day – Go Vote! 

Week 7 Nov 7 Mental Health & Chatbots

Week 8 Nov 14 Trust & Safety

Week 9 Nov 21 What's Next – Beyond 106S, End-Term Boba Party 

Nov 28 **Thanksgiving Recess**

Week 10 Dec 5 **No class; good luck on your finals!** 

**Subject to change – please let me know if you
have any feedback or suggestions at any point!**

Overview of Classes!

Coding for Social Good

Benjamin Yan, CS106S 2024

Stanford | ENGINEERING
Computer Science

Overview of Classes!

What **technologies** (machine learning, sentiment analysis, etc.) can be used to **positively impact the world?**

Coding for Social Good

How can we use **JavaScript** to materialize ideas into **real-world applications?**

Overview of Classes!

What **technologies** (machine learning, sentiment analysis, etc.) can be used to **positively impact the world?**

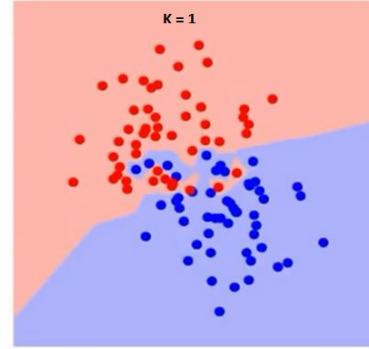
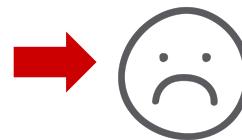
Coding for Social Good

How can we use **JavaScript** to materialize ideas into **real-world applications**?

In **what areas & industries** can we use technology + CS for positive impact?

For what current problems is programming **NOT the answer?**

Overview of Classes!



Cancerous

NOT cancerous

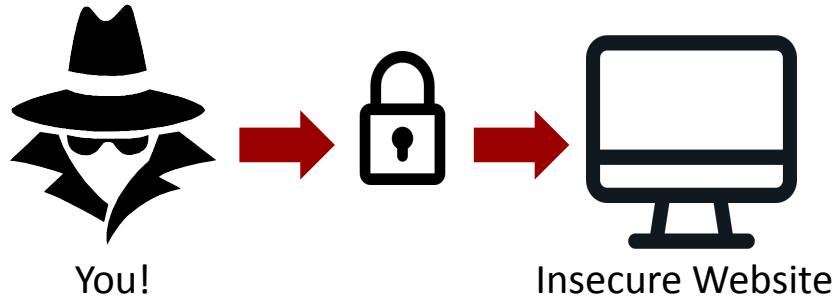
**Sentiment Analysis for Detecting
Hate Speech on Twitter**

**Cancer Diagnosis with
K-Nearest Neighbors**

Overview of Classes!



**Mapping + Quantifying the
Impacts of Climate Change with
Google Earth Engine**



**Ethical Cybersecurity: Hacking an
Insecure Website—to Discover
Vulnerabilities to Patch**

And more!

Benjamin Yan, CS106S 2024

Stanford | ENGINEERING
Computer Science

Let's Dive In!

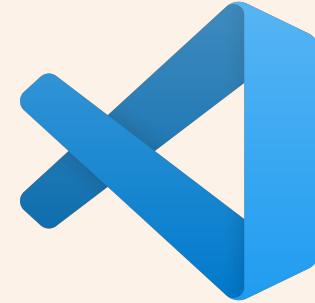
Benjamin Yan, CS106S 2024

Stanford | ENGINEERING
Computer Science



install Chrome

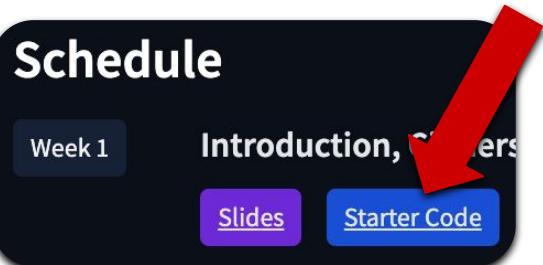
Getting Set Up



install VS Code

(or an editor of your choice)

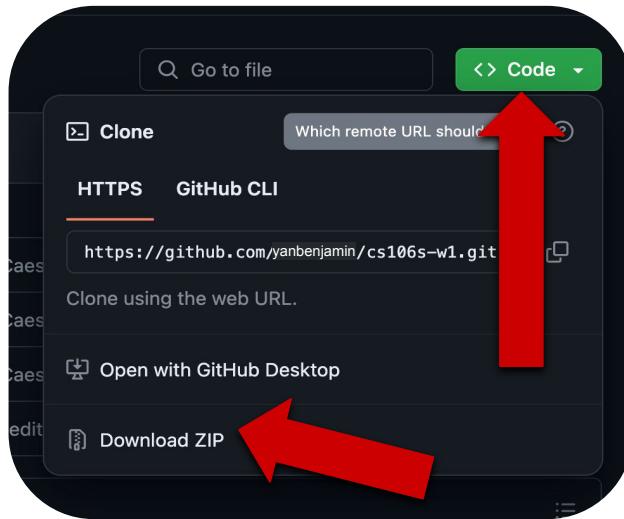
Opening the Starter Code



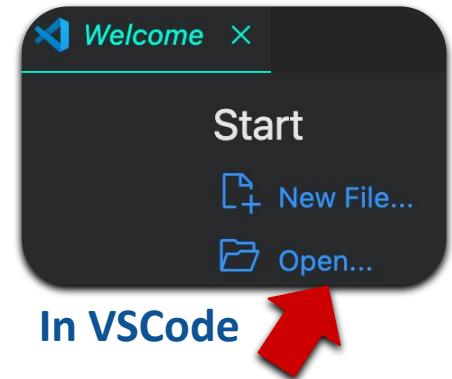
- 1 Navigate to Week 1 of the Schedule section of cs106s.stanford.edu

Also, at this link:

<https://github.com/yanbenjamin/cs106s-w1>



- 2 Click the bright “Code” button, then click “Download ZIP”



- 3 Unzip the download (clicking .zip file should do the trick) and open the folder / files in your editor

HTML, CSS, JS Overview

.html Hypertext Markup Language

.css Cascading Style Sheets

.js JavaScript

- HTML for defining the **webpage content and basic structure**
- CSS for **regulating style and formatting**
- JavaScript for **enabling the HTML/CSS page to be interactive**
 - “Language of the Web”
 - 99% of websites use JavaScript on the client side, making it essential for building browser applications

HTML/CSS

Note: CS 106S isn't a dedicated web development course — but I think it's helpful to at least cover the basics.

index.html

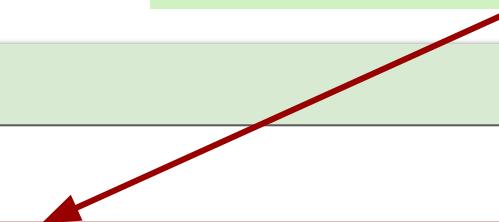
```
<!doctype html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
    <img src = "obiwan.jpg" width = 400>
    <p>Hello there. Open the JavaScript console to continue onward!</p>
  </body>
</html>
```

HTML/CSS

The `<html>` and `</html>` tags enclose all the content.

index.html

```
<!doctype html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
    <img src = "obiwan.jpg" width = 400>
    <p>Hello there. Open the JavaScript console to continue
onward!</p>
  </body>
</html>
```



HTML/CSS

index.html

```
<!doctype html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
        <img src = "obiwan.jpg" width = 400>
        <p>Hello there. Open the JavaScript console to continue
onward!</p>
    </body>
</html>
```

HEAD contains info not displayed on webpage (e.g., browser title, any JavaScript or CSS style files to load)

HTML/CSS

index.html

```
<!doctype html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
    <img src = "obiwan.jpg" width = 400>
    <p>Hello there. Open the JavaScript console to continue
onward!</p>
  </body>
</html>
```

BODY contains everything displayed on the webpage (e.g., text, images, GIFs, etc)

HTML/CSS

index.html

```
<!doctype html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
    <img src = "obiwan.jpg" width = 400>
    <p>Hello there. Open the JavaScript console to continue
onward!</p>
  </body>
</html>
```

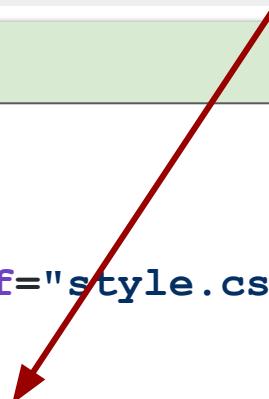
Tags such as `<h2>` enclose each of the HTML elements. Typically have end tag (`</h2>`), but not always (``)

HTML/CSS

Question: How can we stylize each of these webpage elements embedded in tags?

index.html

```
<!doctype html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
        <img src = "obiwan.jpg" width = 400>
        <p>Hello there. Open the JavaScript console to continue
onward!</p>
    </body>
</html>
```



HTML/CSS

Strategy: We use a separate CSS file to specify stylization, colors, etc.

index.html

```
<!doctype html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h2>CS 106S Week 1: JavaScript and Cryptography</h2>
    <img src = "obiwan.jpg" width = 400>
    <p>Hello there. Open the JavaScript console to continue
onward!</p>
  </body>
</html>
```



HTML/CSS

style.css

```
* {  
    font-family: 'Courier New', monospace !important;  
} /* sets everything (*) on page to Courier New font */  
  
h2 {  
    color: darkred;  
} /* sets section heading to a dark red color */
```

HTML/CSS – Browser Rendering

Resulting webpage from **index.html** and **style.css**



HTML/CSS – Browser Rendering

Resulting webpage from **index.html** and **style.css**



Heading **<h2>** tag,
with dark red color
from CSS

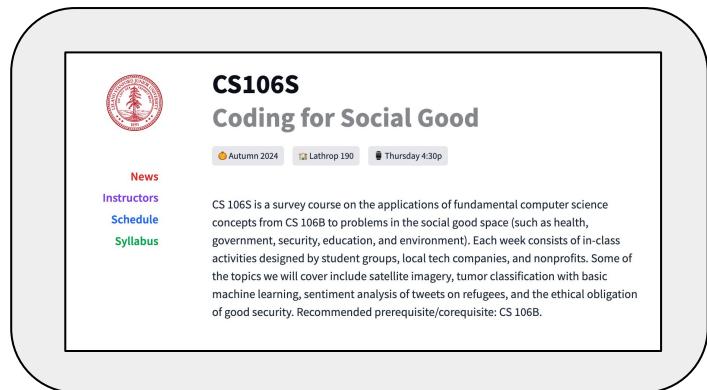
**** tag, loading in
image [obiwan.jpg](#)

Text in paragraph
tag **<p>**

Any questions so far?

What is index.html?

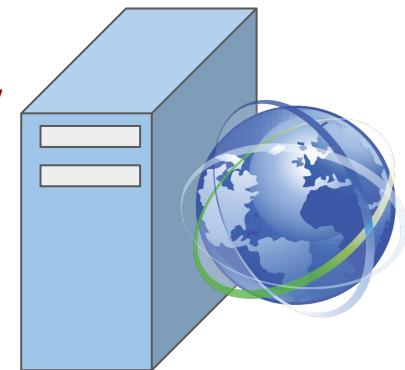
- In the starter code, you'll find a file named **index.html**; using Finder or your OS equivalent, **open it in Google Chrome**
- This is the **homepage** of a website.



Client Browser

HTTPS Request for
https://cs106s.stanford.edu/

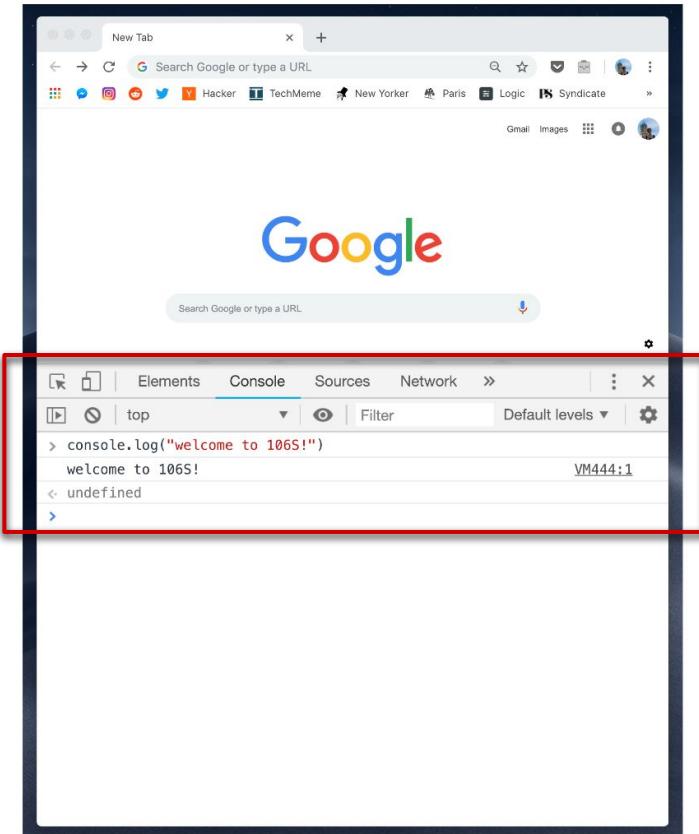
HTTPS Response of
index.html



Web Server

Stanford | ENGINEERING
Computer Science

JavaScript in Chrome



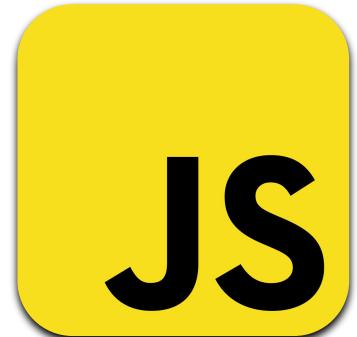
1. Open **index.html** in Chrome
2. On Mac: Press **cmd**—**option**—**j**
On Windows: Press **ctrl**—**shift**—**j**

Don't let go of the previous key while pressing the next.

Here, **in the console that pops up**, we can input and run JavaScript code!

Onto the JavaScript Tutorial!

To follow along, inspect the file **tut.js** in your code editor; we'll be running the JavaScript commands inside on the Chrome console!



JavaScript – Hello World

- Unlike Python, **do not use print()** for outputting to console; it will try printing ... to a physical printer lol

tut.js

```
console.log("Hello World!");
```



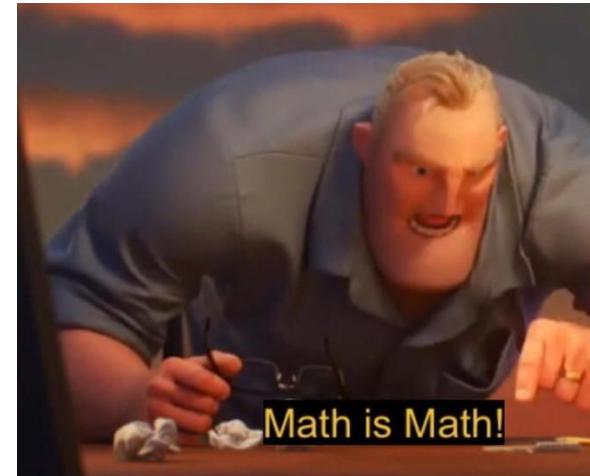
JavaScript – Math Operations

- Works similarly to Python; note the (optional) semi-colon

tut.js

```
1 + 1; // => 2
10 - 4; // => 6
2 * 7; // => 14
3 / 2; // => 1.5

/* mod: remainder function */
4 % 2; // => 0
5 % 2; // => 1
6 % 2; // => 0
10 % 26; // => 10
```



JavaScript – Variables

- Use **let** keyword to define **variables of any type** (int, string, array, etc.), and **const** instead for variables with fixed values

```
let variableName = expression;
```

tut.js

```
let num_classes = 4;
num_classes += 1; // modified
num_classes = num_classes - 2;

const CS106S_UNITS = 1;
var total_units = 17;
```

Note: **var** (in the place of **let**) is often seen in older JS code; as a general principle, **avoid using it**. tut.js has an explanation of the key difference (var scoping)

JavaScript – Functions & Calls

```
// general structure
function functionName(arg list){
    statements in function body
}
```

tut.js

```
function add(x,y) {
    let answer = x+y;
    return answer;
    // Just like Python!
}
```

```
// calling function
functionName(args)
```

● ○ ● JS console

```
> add(3,5)
< 8

> add(add(1,2), 3)
> 6
```

JavaScript – Conditionals (if, else if, else)

tut.js

```
function getMax(x,y,z){  
    if (x >= y && x >= z){  
        return x;  
    }  
    //either y or z is max  
    else if (y >= z){  
        return y;  
    }  
    else{  
        return z;  
    }  
}
```

== Equality **==!** Non-equality
&& Logical AND **||** Logical OR

< **<=** **>** **>=**

Operate as mathematically expected

Note: A **return** statement exits out of the function immediately i.e. the following lines are not run.

JavaScript – Objects

JS objects are akin to Python **dictionaries**
i.e. key-value pairs enclosed in {}. Entries
can be of different type!

tut.js

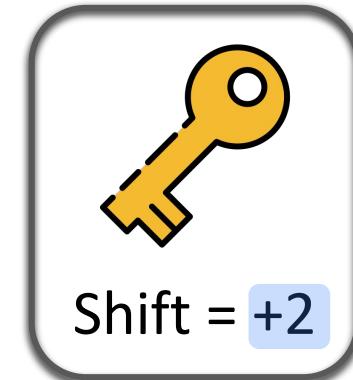
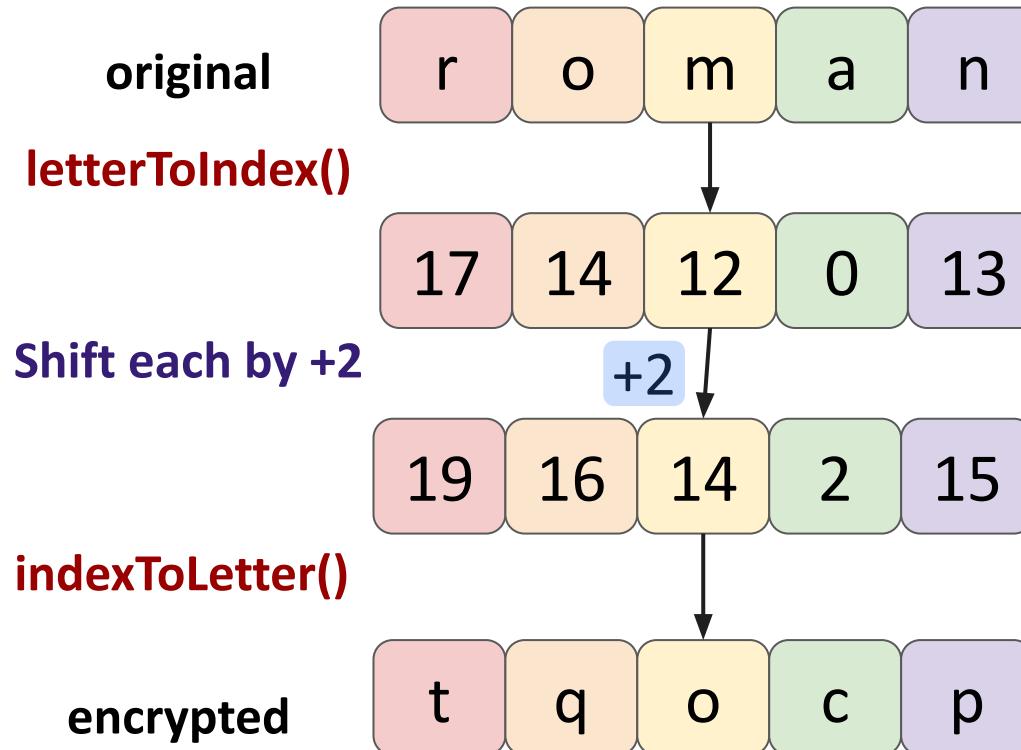
```
let menuPrices = {  
    "hotdog": 5,  
    "soda": 3,  
    "pretzel": 4.50,  
    "funnelcake": 9,  
}
```

Use [] bracket notation to
get values!

JS console

```
> menuPrices ["soda"]  
< 3  
  
> menuPrices ["pretzel"]  
> 4.50
```

Today's Coding – Caesar Ciphers



Checkpoint #1

- Traverse over to **assignment.js** on your code editor.



Task

Implement the function `letterToIndex()`

Input: A lowercase letter (a-z)

Output: Index in alphabet (a=0,b=1,c=2,...,z=25)

Tip - You may find the key-value object `mapping` in the file useful.

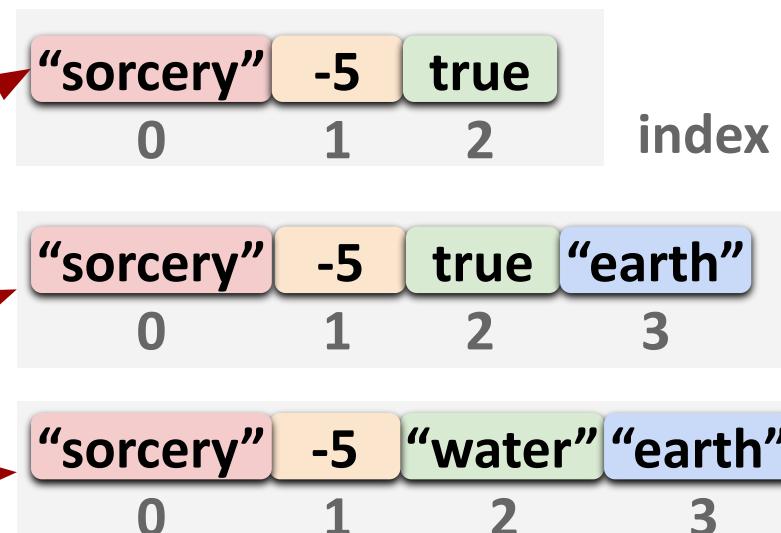
Note: After editing the JS file, make sure to click **File -> Save in VSCode**, and **refresh the Chrome page**, for the edits to manifest in the console.

JavaScript – Arrays

- Ordered lists of any / heterogeneous data type, 0-indexed in JS.
- Mutable and of variable length.

tut.js

```
let myArray = ["sorcery", -5,  
true];  
  
myArray[0]; // => "sorcery"  
myArray[1]; // => -5  
  
myArray.push("earth");  
myArray.length; // => 4  
myArray[2] = "water";
```



Checkpoint #2



Task in assignment.js

Implement the function `indexToLetter()`

Input: Non-negative index of a letter, can be > 25

Output: Corresponding lowercase letter; numbers above 25 wrap around i.e. $0=a, 1=b, \dots, 25=z, 26=a, 27=b, \dots$

Tip - The array `alphabet=['a','b',...,'z']` may come in handy. For dealing with overflow, take any letter, say 'a'; how are all its possible indices related?

Checkpoint #3



Task in assignment.js

Implement the function
shiftLetter()

Inputs: **original** (letter to shift), **shift** (length to transpose letters by)

Output: shifted letter

Tip - Use **letterToIndex()** and **indexToLetter()**!

Example Functionality



JS Console

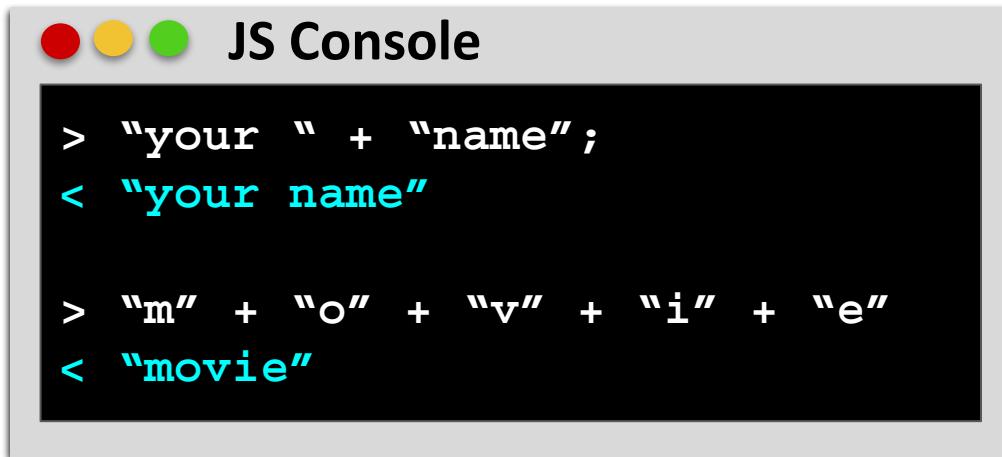
```
> shiftLetter('a',1)
< 'b'

> shiftLetter('a',4)
< 'e'

> shiftLetter('z',3)
< 'c'
```

JavaScript – Strings

- **Text or sequence of characters**, wrapped in quotation marks.
- Like Python/C++, strings can be concatenated using + operator



The image shows a screenshot of a JavaScript console window titled "JS Console". It features three colored circular icons (red, yellow, green) at the top left. The console output is as follows:

```
> "your " + "name";
< "your name"

> "m" + "o" + "v" + "i" + "e"
< "movie"
```



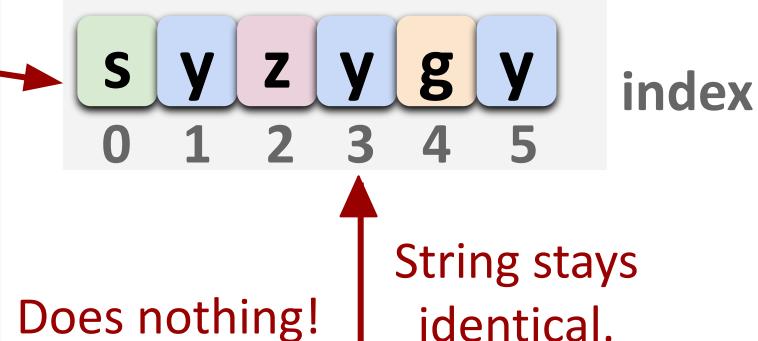
JavaScript – Strings

- Indexing similar to arrays, though unlike arrays, strings are immutable
i.e. **its contents cannot be changed once declared.**

tut.js

```
let myString = "syzygy";
myString[1]; // => y
myString[2]; // => z
myString.length; // => 6

//attempt to change a letter
myString[0] = "a";
```



JavaScript – String Methods

- However, strings can be replaced.
- String methods (like `+`) actually create an **entirely new string**, then re-assign the variable name to that string!



tut.js

```
let myString = "syzygy";  
myString += " !";  
myString = myString + " !";
```

JavaScript – Loops!

- ‘For’ loops for executing a block of code a fixed number of times; JS **syntax is similar to C++/Java**

tut.js

```
function sayHelloThereNTimes(N) {  
    //loop runs N times  
    //(i = 0,1,2,...,N-1)  
    for (let i = 0; i < N; i++) {  
        console.log("Hello There!");  
    }  
}
```

JavaScript – Loops!

- ‘For’ loops for executing a block of code a fixed number of times; JS **syntax is similar to C++/Java**

tut.js

```
function sayHelloThereNTimes(N) {  
    //loop runs N times  
    //(i = 0,1,2,...,N-1)  
    for (let i = 0; i < N; i++) {  
        console.log("Hello There!");  
    }  
}
```

Each time this runs, the value of **i** is incremented by 1, starting from **i = 0**.

The block will be executed until condition **i < N** is broken, i.e. **i** reaches **N**, which occurs after **N runs**.

Please do not run `sayHelloThereNTimes(1000000)`;

JavaScript – Iterating Over String

tut.js

```
function printAllLetters(str){  
    for (let i = 0; i < str.length; i++) {  
        //get ith letter of string  
        let letter = str[i];  
        console.log(letter);  
    }  
}
```



JS Console

```
> printAllLetters("kind")  
k  
i  
n  
d
```



Final Checkpoint – The Full Pipeline



Task in assignment.js

Implement `encrpytCaesar()`

Inputs: `original` (string to encrypt), `shift` (how many places to move each letter down the alphabet)

Output: The encrypted string

Tip – Loops! And take advantage of functions you've already written!

Sample Functionality



JS Console

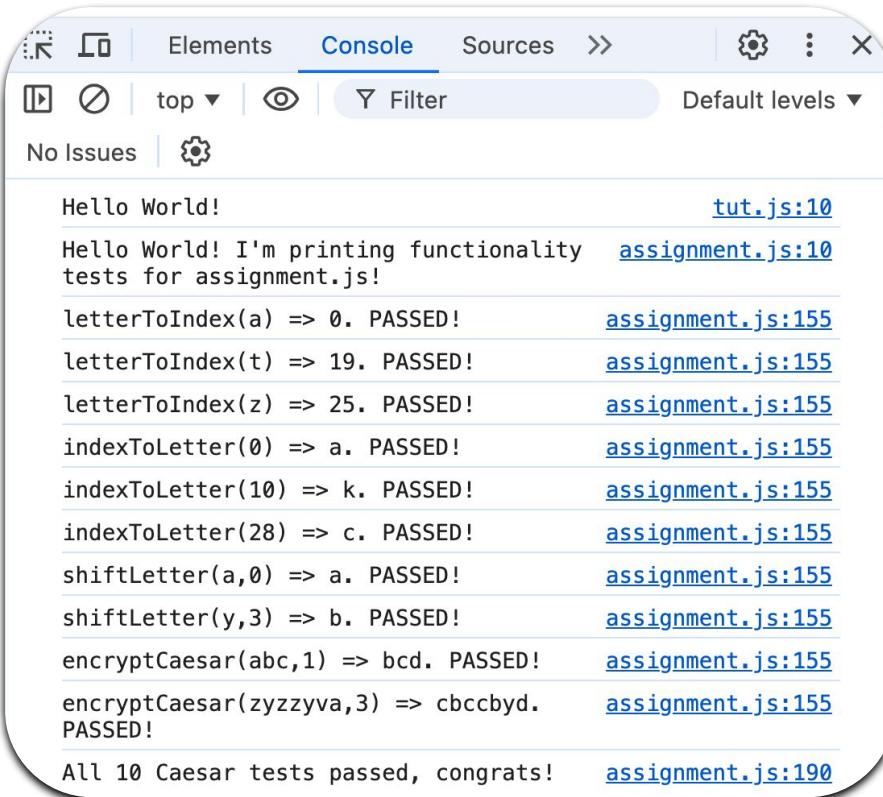
```
> encrpytCaesar  
(`abc`,1)
```

```
< 'bcd'
```

```
> encryptCeasar  
(`zyzzyva`,3)
```

```
< 'cbccbyd'
```

Sanity Testing



The screenshot shows a browser's developer tools console tab selected. The output area displays a series of test logs from a file named assignment.js. The logs indicate that all 10 Caesar cipher tests have passed. The logs are as follows:

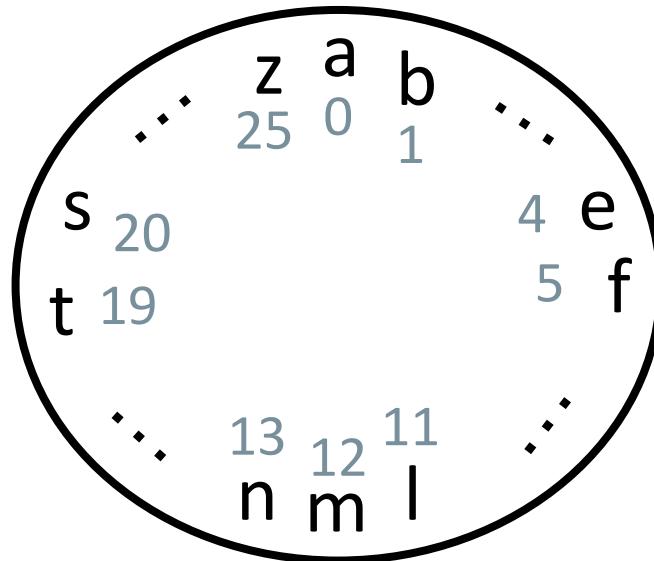
- Hello World! [tut.js:10](#)
- Hello World! I'm printing functionality tests for assignment.js! [assignment.js:10](#)
- letterToIndex(a) => 0. PASSED! [assignment.js:155](#)
- letterToIndex(t) => 19. PASSED! [assignment.js:155](#)
- letterToIndex(z) => 25. PASSED! [assignment.js:155](#)
- indexToLetter(0) => a. PASSED! [assignment.js:155](#)
- indexToLetter(10) => k. PASSED! [assignment.js:155](#)
- indexToLetter(28) => c. PASSED! [assignment.js:155](#)
- shiftLetter(a,0) => a. PASSED! [assignment.js:155](#)
- shiftLetter(y,3) => b. PASSED! [assignment.js:155](#)
- encryptCaesar(abc,1) => bcd. PASSED! [assignment.js:155](#)
- encryptCaesar(zyzzyva,3) => cbccbyd. PASSED! [assignment.js:155](#)
- All 10 Caesar tests passed, congrats! [assignment.js:190](#)

All tests should pass after **encryptCaesar()** is successfully implemented!

Solution code available on website right after class :)

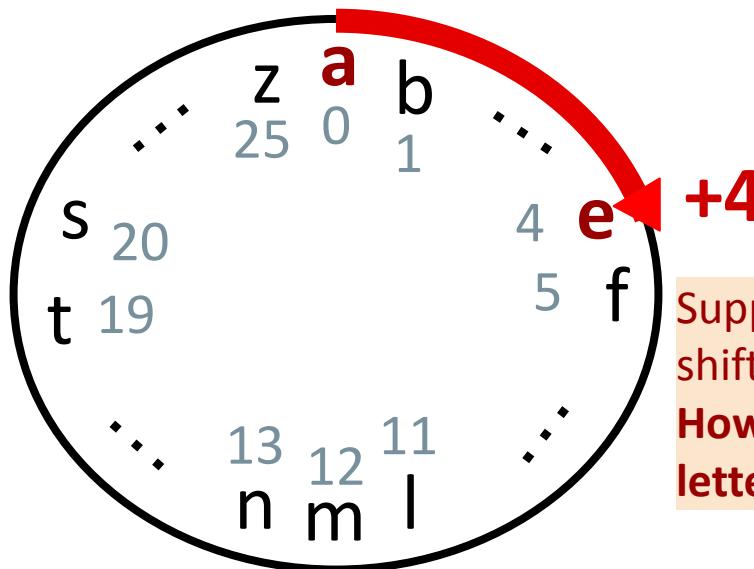
(Optional) Caesar Decryption

- Great, we now have encrypted strings! **How do we decode them?**



(Optional) Caesar Decryption

- Great, we now have encrypted strings! **How do we decode them?**

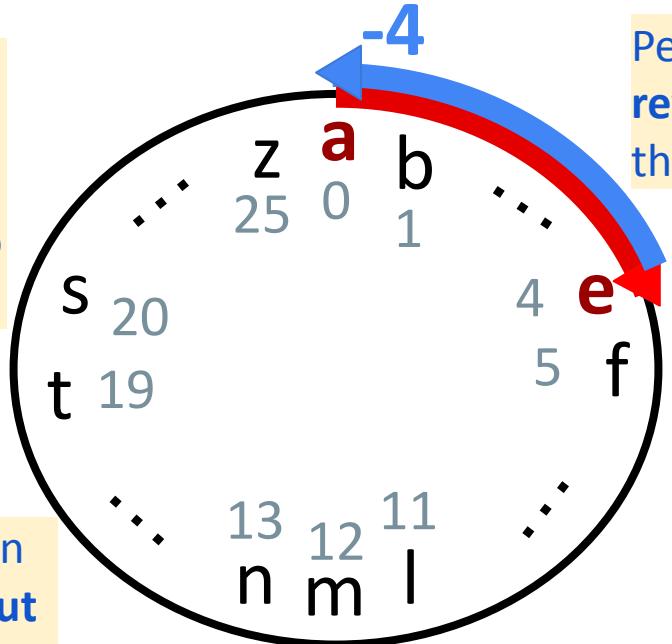


Suppose our cipher has a shift of +4, e.g., $a \Rightarrow e$.
How might we go back to letter 'a'?

(Optional) Caesar Decryption

- Great, we now have encrypted strings! **How do we decode them?**

This decodes 'e' **but has a problem.** If the negative shift is too large, the index will go negative.



Perhaps, we can do a reverse shift of -4! So that $e \Rightarrow a$.

So how might we attain the same effect **without a negative shift?**

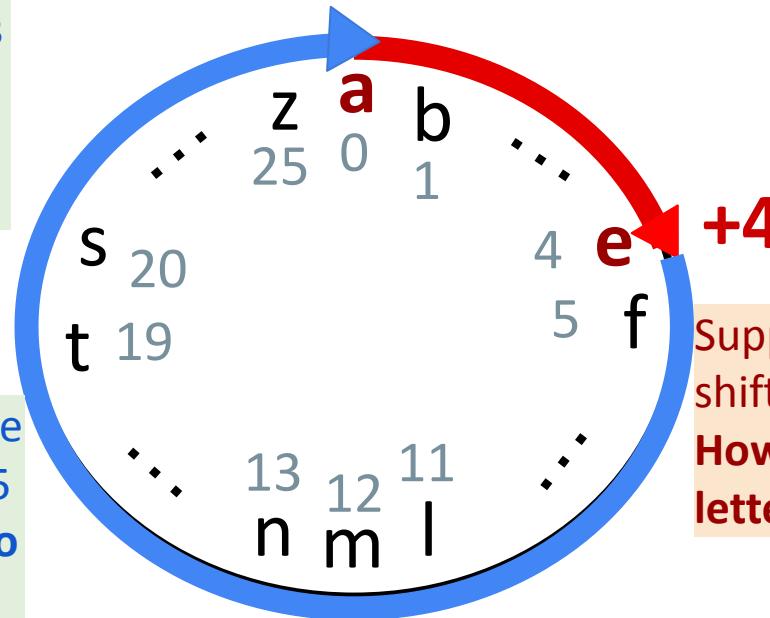
Suppose our cipher has a shift of +4, e.g., $a \Rightarrow e$. **How might we go back to letter 'a'?**

(Optional) Caesar Decryption

- Great, we now have encrypted strings! **How do we decode them?**

How large should this shift be? Keep in mind, alphabet / circle length is 26.

Positive shift! Since we can handle indices >25 (Checkpoint #2), we go this way around the circle instead.

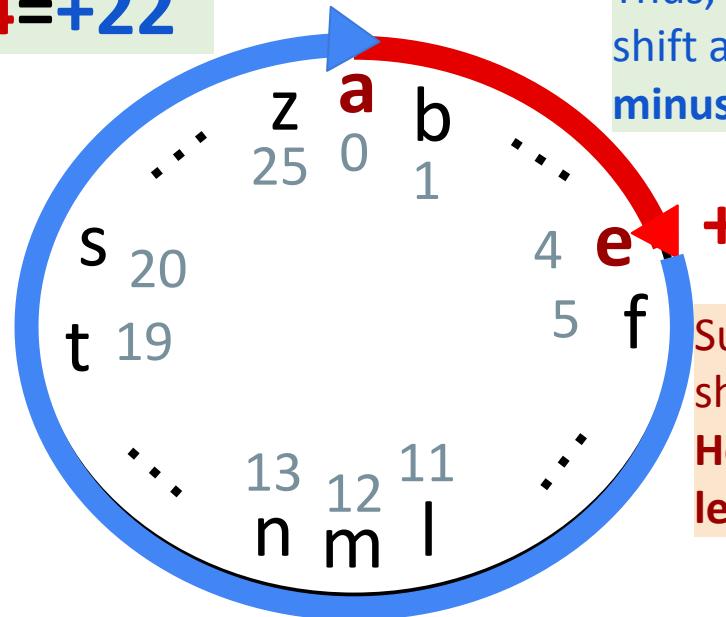


Suppose our cipher has a shift of +4, e.g., $a \Rightarrow e$.
How might we go back to letter 'a'?

(Optional) Caesar Decryption

- Great, we now have encrypted strings! **How do we decode them?**

$$26-4=+22$$



Thus, our decryption scheme will shift all letters by +22, or 26 minus the encryption shift.

Suppose our cipher has a shift of +4, e.g., a => e.

How might we go back to letter 'a'?

(Optional) Caesar Decryption

assignment.js

```
/* Decrypts the given string from Caesar cipher with a given shift length.*/
function decryptCaesar(ciphertext, shift) {
    let reverse_shift = 26 - shift;
    return encryptCaesar(
        ciphertext, reverse_shift);
}
```

We observe that decryption **reverses** each letter shift in the encryption—thus recovering the original message.

● ● ● Obi-Wan

```
> encryptCaesar
('hellothere', 4)
< mjqqtymjwj
```



● ● ● Gen. Grevious

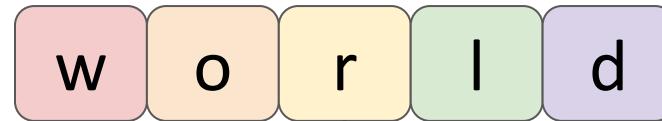
```
> decryptCaesar
('mjqqtymjwj', 4)
< hellothere
```



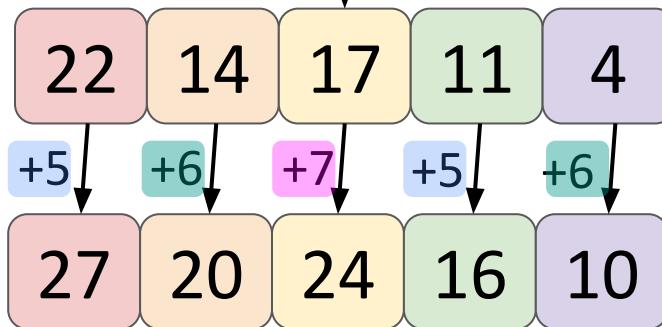
(Optional Extension) Vigenère Ciphers

We won't get to this in class, but feel free to try it out on your own!

original



letterToIndex()



Shift each letter

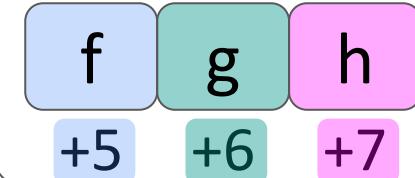
Alternating shifts
based on keyword

indexToLetter()

encrypted



Keyword



Remark: It's like having multiple Caesar ciphers in one encryption!

Stanford | ENGINEERING
Computer Science

Check-Off Form!

To get attendance credit each class, you'll fill out a **brief check-off form** (~2 – 5 min to complete).

For today, click the “Check-Off Form” link in the Week 1 section of [cs106s.stanford.edu!](https://tinyurl.com/cs106s-aut24-w1-checkoff)



<https://tinyurl.com/cs106s-aut24-w1-checkoff> (case sensitive!)

Looking Forward to this Autumn



Teaching this 1-unit wonder has been a truly wonderful privilege for me.

Thank you for being here to learn with us, and I hope this will be, for you, a fun, rewarding adventure.



Have an awesome first week of classes! :)



Stanford | ENGINEERING
Computer Science