Welcome to CS 106S!

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

Ben Yan, Winter 2025 ☃️

cs106s.stanford.edu

Stanford | ENGINEERING
Computer Science

Happy New Year 🎉

Welcome to the First Day of Class!

Thank you for heading all the way to McMurtry – not to mention all the way up to the third floor!!

Hi I’m Ben!

- **Stanford CS MS ‘26**, Jerry is my advisor
- Just finished my undergrad here ‘24! Double-majored in CS & Math, Minored in Creative Writing (Prose)
- Studied abroad at Oxford my senior winter – tutorial in Creative Writing

- Prev. SWE @ NVIDIA (CUDA Systems), did CS research for 2ish years – life rn is more teaching-oriented
- Have taught CS 106S 3x before (🍁Autumn, 🌱Spring last year, and 🍁 this autumn), and **I love teaching it!**
- Was also 🍁CS106AX head TA last quarter, ☃️ TAing CS107 this winter & very excited! ~~not heap alligator though~~

🎨 Interests, what brings me joy

- **Anime & manga** (my all-time favs *Blue Box, Jujutsu Kaisen*, *Demon Slayer*), Spider-Verse, Gravity Falls, been getting into K-dramas very recently
- **Poetry & novel writing** – my one "claim to fame" is once writing 2 novels (50k words each) in one month for NaNoWriMo 2023 / English 190E
- I haven't written much since :(
- **Music:** Olivia Rodrigo, Phoebe Bridgers, YOASOBI, Blackpink, SEVENTEEN, BTS, Charli XCX, Clairo
- Not my Spotify wrapped though – it's embarrassing
- **Books:** Anything by Ocean Vuong, Elif Batuman
- Always looking for recs!

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 winter / year? 🌲
  - ❏ Something you did over the winter break ☃️
  - ❏ Music / book / show recommendations? 🎶
  - ❏ Anything else you'd like to share! :)

The Map For Today

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

Course Mechanics

- ❑ 1 unit, S/NC
- ❑ **Attendance (7/9*)**
  - ❑ 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

bbyan@stanford.edu

🧭 Place & Time

🎨 McMurtry Art Building, Room 350
Tues, 4:30 - 6:20 PM; usually try to keep class to 90 minutes ish

👋 Office Hours

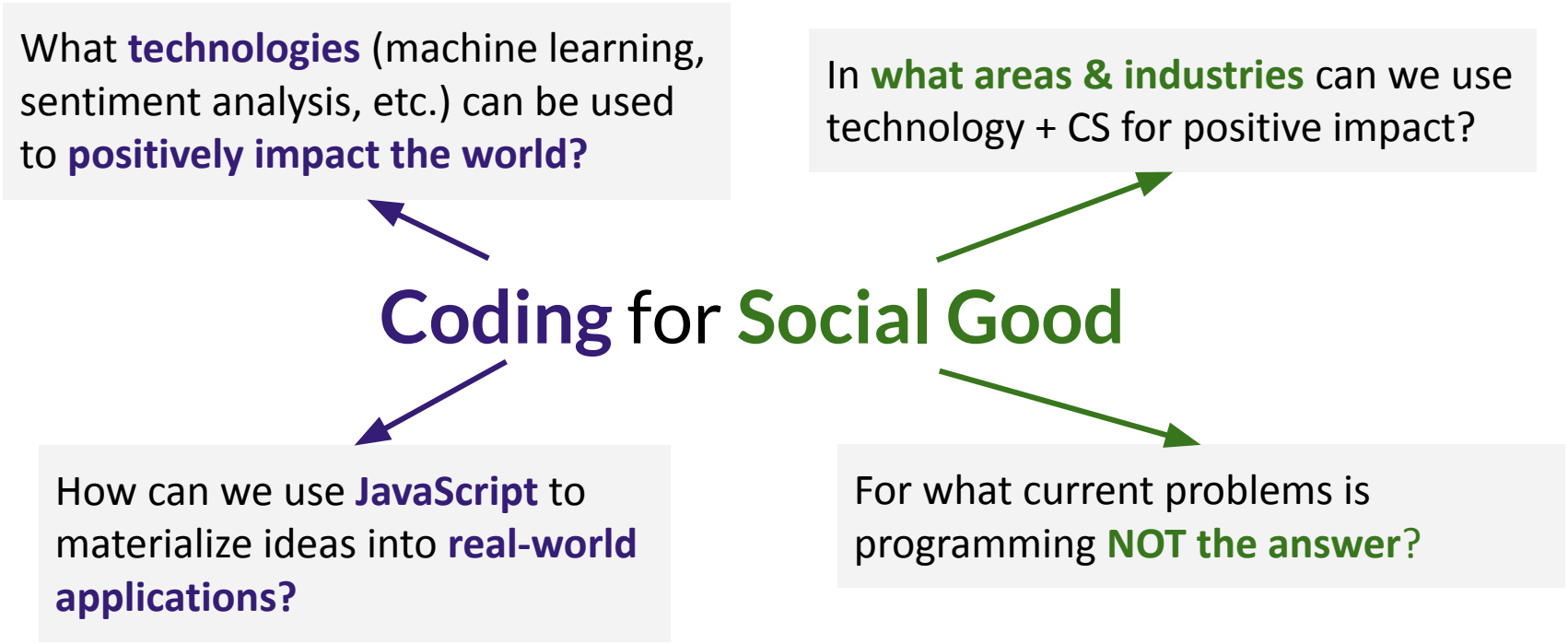
After class, or email/Slack me!
No assignments in 106S, but happy to chat about material, CS, Stanford, anything!

Course Schedule

Week 1	Jan 7	Intro, JavaScript, Ciphers
Week 2	Jan 14	Sentiment Analysis & Refugee Tweets
Week 3	Jan 21	CS for Climate Change
Week 4	Jan 28	KNN for Cancer Detection
Week 5	Feb 4	Cybersecurity and Ethical Hacking
Week 6	Feb 11	Open Source & Web Software
Week 7	Feb 18	Trust & Safety
Week 8	Feb 25	Mental Health & Chatbots
Week 9	Mar 4	What's Next – Beyond 106S, End-Term Boba Party 🧋
Week 10	Mar 11	**No class; good luck on your finals!** 🍀

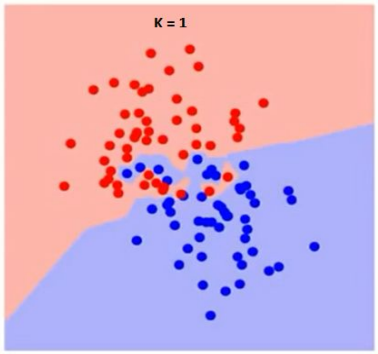
Subject to change – please let me know if you have any feedback or suggestions at any point! Happy to run this class in the way it'd be most helpful to you

Overview of Classes!



Overview of Classes!

Sentiment Analysis for Detecting Hate Speech on Twitter

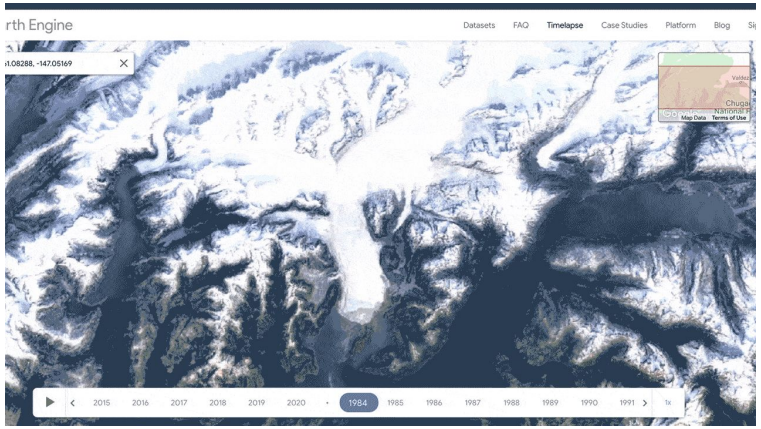


Cancerous

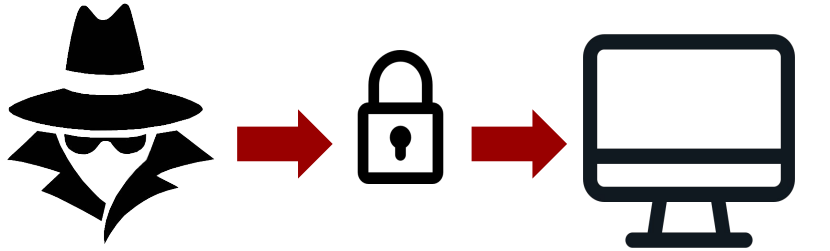
NOT cancerous

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!

Let’s Dive In!

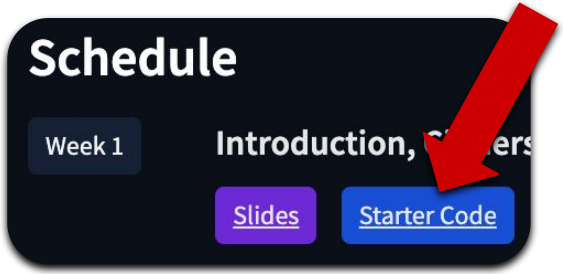
Getting Set Up

install Chrome

install VS Code

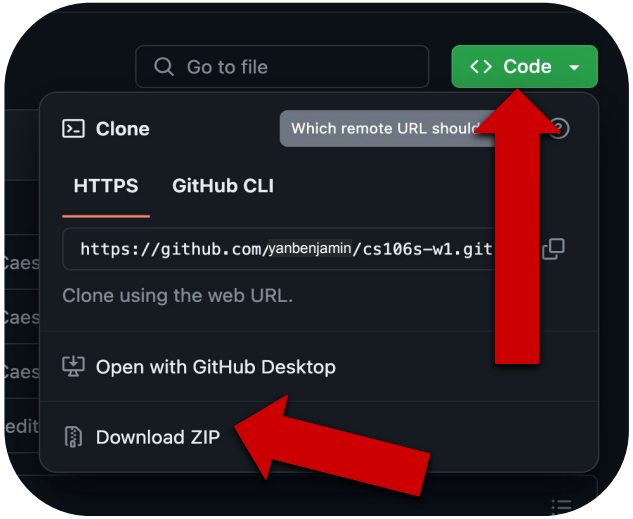
(or an editor of your choice, Sublime Text is also great)

Opening the Starter Code

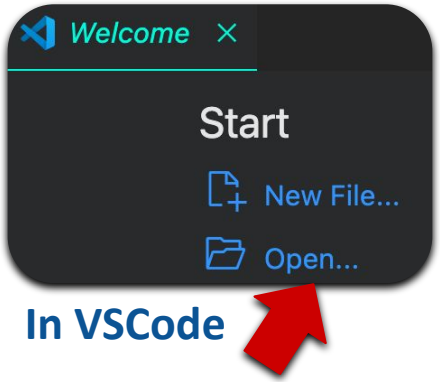


1 Navigate to Week 1 of the Schedule section of **cs106s.stanford.edu**

Also, at this link: **https://github.com/yanbenjamin/cs106s-w1**

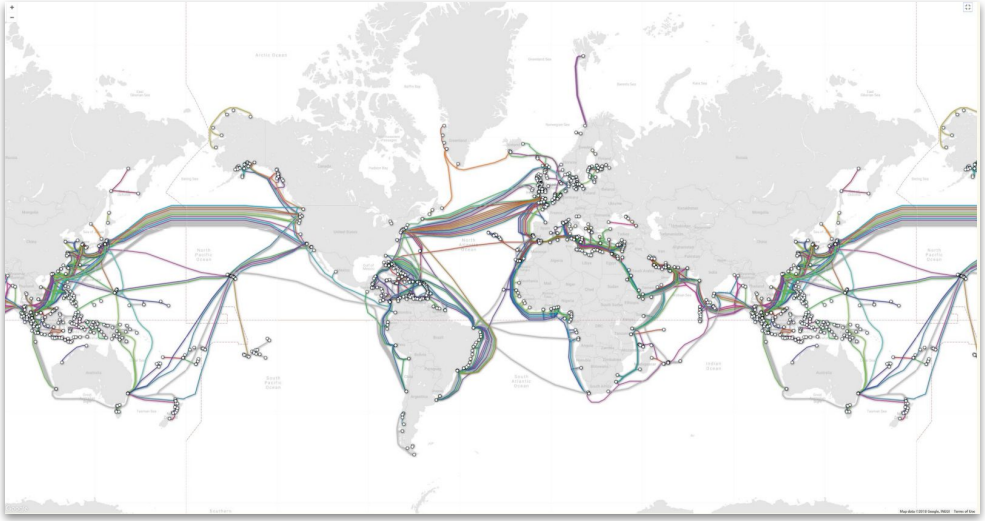


2 Click the bright **“Code”** button, then click “Download ZIP”



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

HTML, CSS, JS Overview



HTML, 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 components on the webpage to be interactive**
  - 🌐 “Language of the Web”
  - 🔗 99% of websites use JavaScript on the client side, making it essential for building browser applications

HTML Layout

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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

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

hello_cat2.jpg

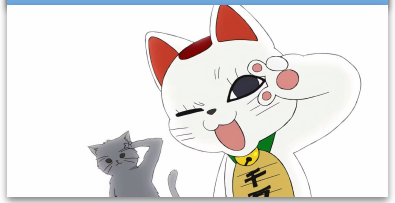
hello_cat3.jpg

hello_cat4.jpg

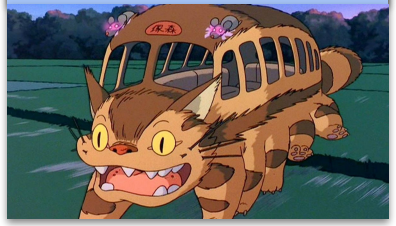
hello_cat.jpg

hello_cat5.jpg

hello_cat6.jpg



hello_cat7.jpg



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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

🌐 **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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

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

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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

🌐 **BODY contains everything displayed on the webpage (e.g., text, section headings, images, GIFs, 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 = "hello_cat.jpg" width = 400>
        <p>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 (<img>,
 – line break)**

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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

🌐 **Question: How can we stylize each of these webpage elements embedded in tags? (e.g., use different colors, fonts, spacing)**

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 = "hello_cat.jpg" width = 400>
        <p>Open the JavaScript console to continue onward!</p>
    </body>
</html>
```

🌐 **Strategy: We use a separate CSS file to specify stylization, layout, font, colors, etc. (HTML → content, CSS → style)**

style.css

```
img{
    border-style: dashed;
    border-width: 5px;
} /* adds dashed border with 5 pixel width to images */
h2{
    color: darkblue;
} /* sets section heading (<h2>) to a dark blue color */
```

In a CSS file, style rules are written in the form below:

Which HTML element(s) the styles should apply to, e.g., all <img> tags, the id of a specific element

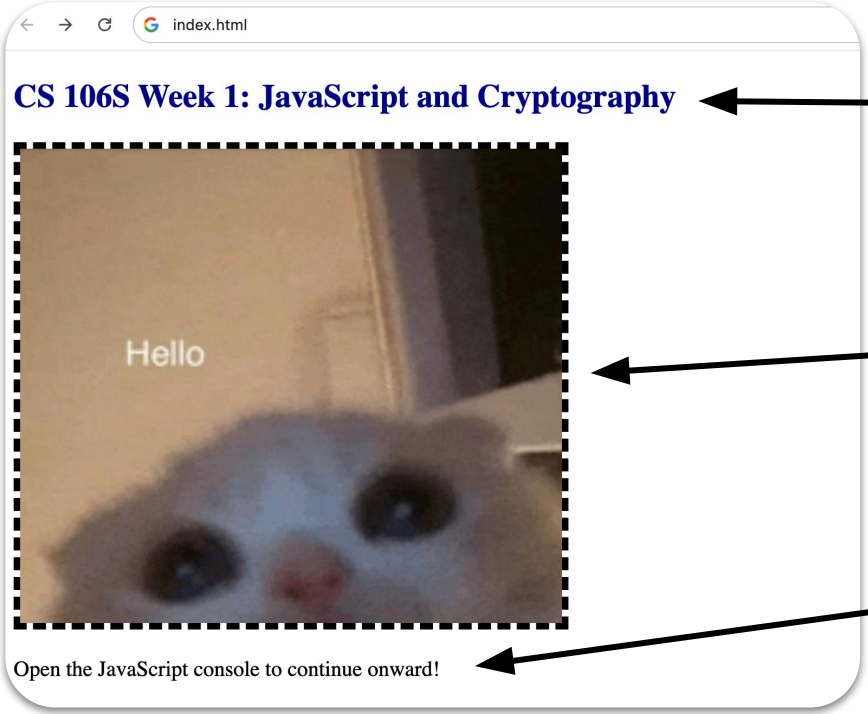
```
selector{
    property-name: property-value
    ... more pairs of CSS properties & values
}
```

One of several CSS keywords specifying a style detail, e.g., *font-family, font-size, color, background-color, border-width,* etc.

Possible **values** depend on property name, e.g., *color* (**red, green**), *font-family* (**serif, sans-serif, cursive, fantasy**), etc.

HTML/CSS – Browser Rendering

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



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

<img> tag, loading in image `hello_cat.jpg`, stylized with dotted 5-pixel-wide frame

Text in paragraph tag **<p>**

What JavaScript does

- JavaScript is a **powerful 'interface' with an HTML webpage,** enabling HTML elements to be programmatically accessed and modified.
  - Empowers **dynamic web apps** that respond to user interactions.
- To get a high-level sense of what JavaScript can do (before we dive into a starter tutorial on its features & syntax), it can:
  - Append, remove, or modify any HTML nodes i.e. tags/elements

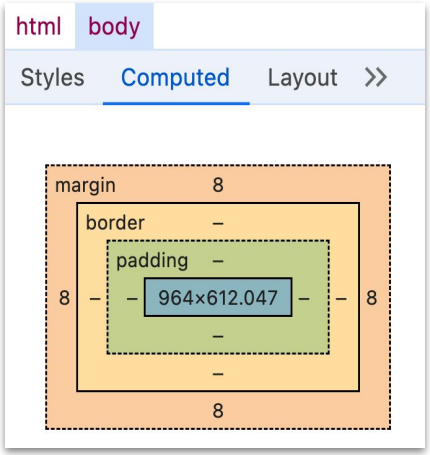
    Open pop-up window | Load new images | Toggle display to night mode
  - React to user events/actions with parts of the webpage, e.g., buttons

    Button clicks | Hovering mouse over image | Keyboard key is pressed
- JS is now quite **general-purpose**! Has evolved beyond its web roots.

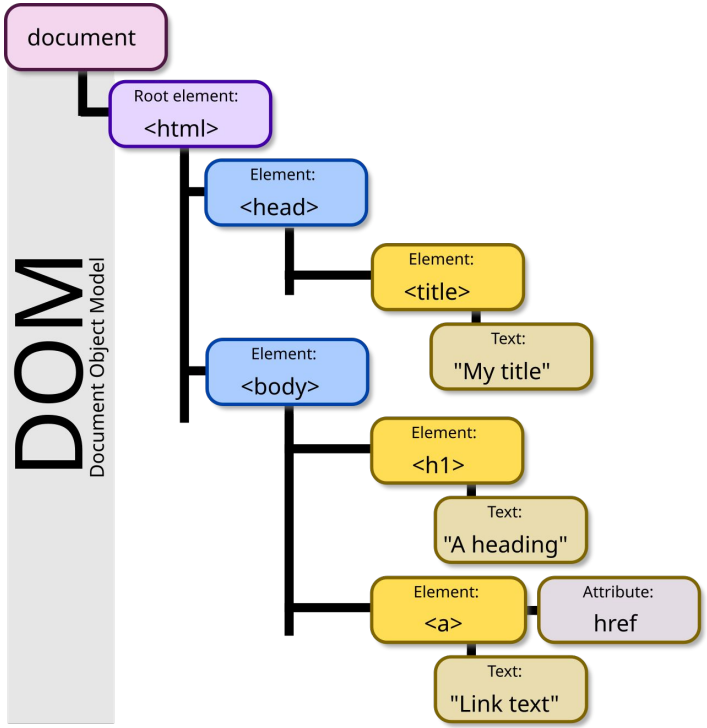
(Extra) HTML Rendering

How does the browser read the HTML page to have it displayed on screen? (How does the text magically turn into a visual, tactile experience?)

- A browser engine parses the HTML into a **Document Object Model (DOM),** an internal **tree-based** representation of the webpage.
- The DOM tree is a **hierarchy of data objects** called ***elements,*** which usually correspond to paired set of tags in the HTML.
- Using a rendered tree, the browser calculates the *layout*, dimension sizes, and locations of each element (setting aside empty boxes on the screen); then, it very efficiently *paints* each element as pixels into those boxes.
- Visualization of DOM tree on the next slide!



(Extra) HTML: DOM Tree Model




```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="www.google.com">
   Link text</a>
  </body>
</html>
```

Every HTML element on the page is a node/member of the DOM tree!

(Extra) JS Applications on DOM Tree

- This DOM tree operates as a **convenient interface between the HTML document and JavaScript**, enabling the webpage to be programmatically accessed and changed upon events (e.g., if the user clicks a button)
  - Empowers **dynamic**, real-time generated **web pages!**
- To get a high-level sense of what JavaScript can do (before we dive into a starter tutorial on it!), by surgically exploring the DOM tree, it can:
  - Append, remove, or modify any nodes i.e. HTML elements/attributes

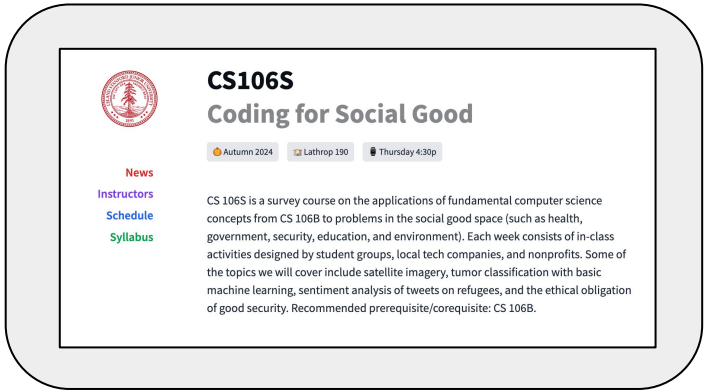
    Open pop-up window | Load new images | Toggle display to night mode
  - React to user events/actions with parts of the webpage, e.g., buttons

    Button clicks | Hovering mouse over image | Keyboard key is pressed
- JS is now quite general-purpose! Has evolved beyond its web roots.

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.



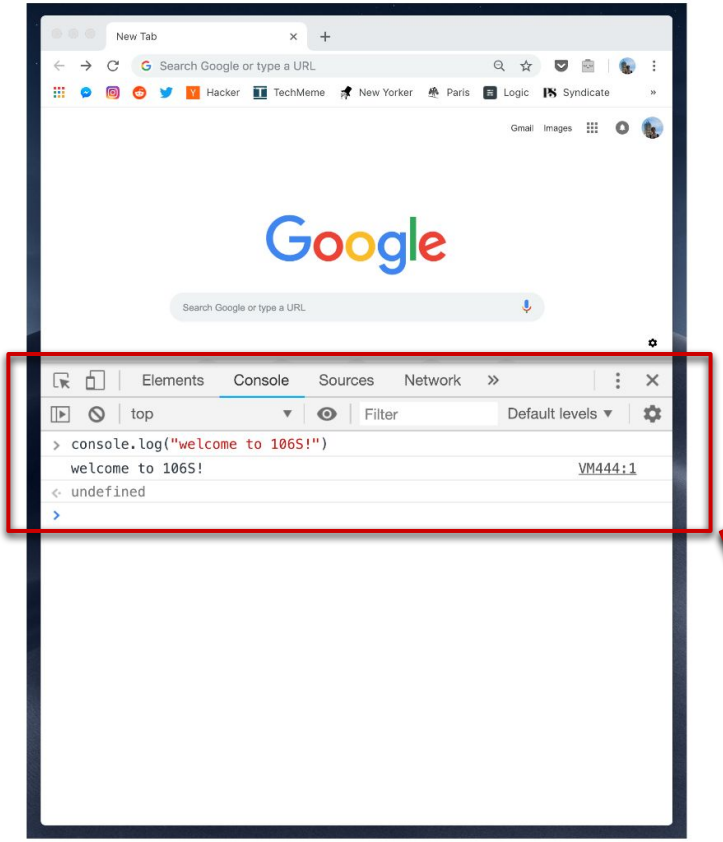
Client Browser

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

HTTPS Response of
index.html

Web Server

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!

Inspect the file `tutorial.js` in your code editor.

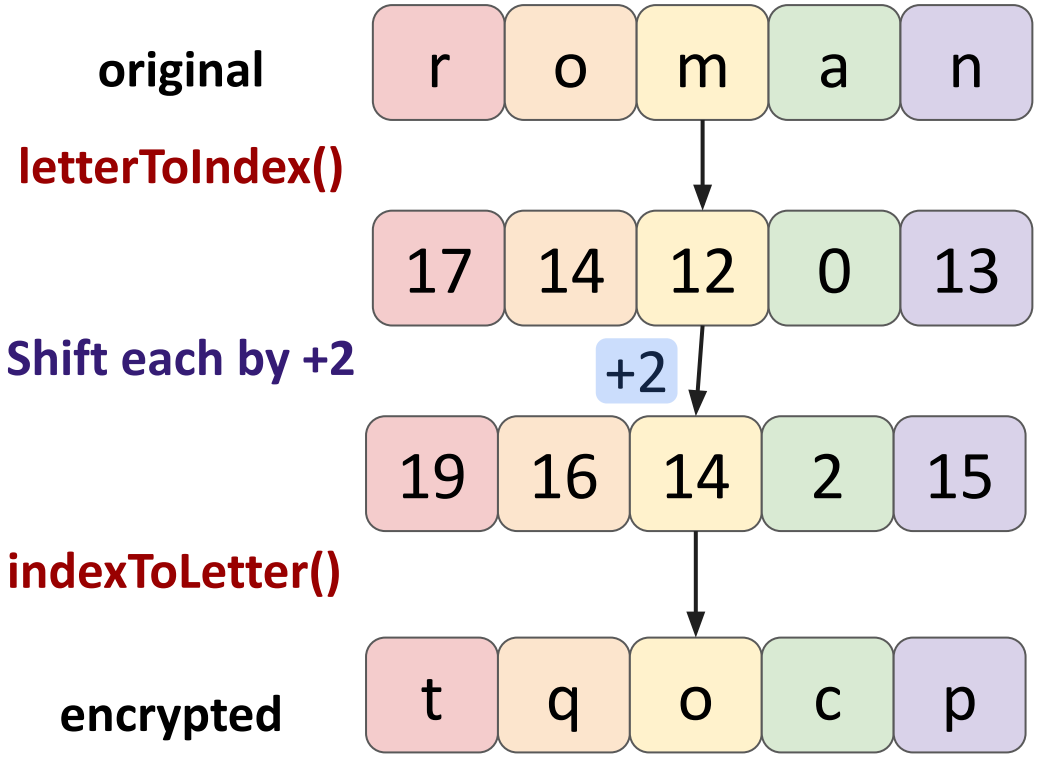
To experiment around, copy & paste, and tinker with / run the JavaScript commands in the Chrome console!

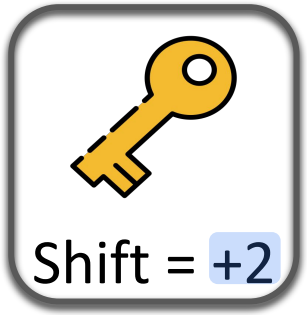
More fleshed-out / lecture-notes-style version at the link below!

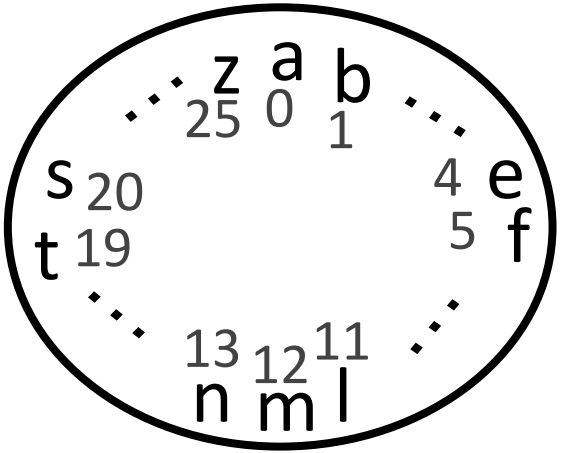
https://web.stanford.edu/class/cs106s/handouts/js-tutorial.html

First time debut of this, so pls lemme know if there's any typos!

Today's Coding – Caesar Ciphers

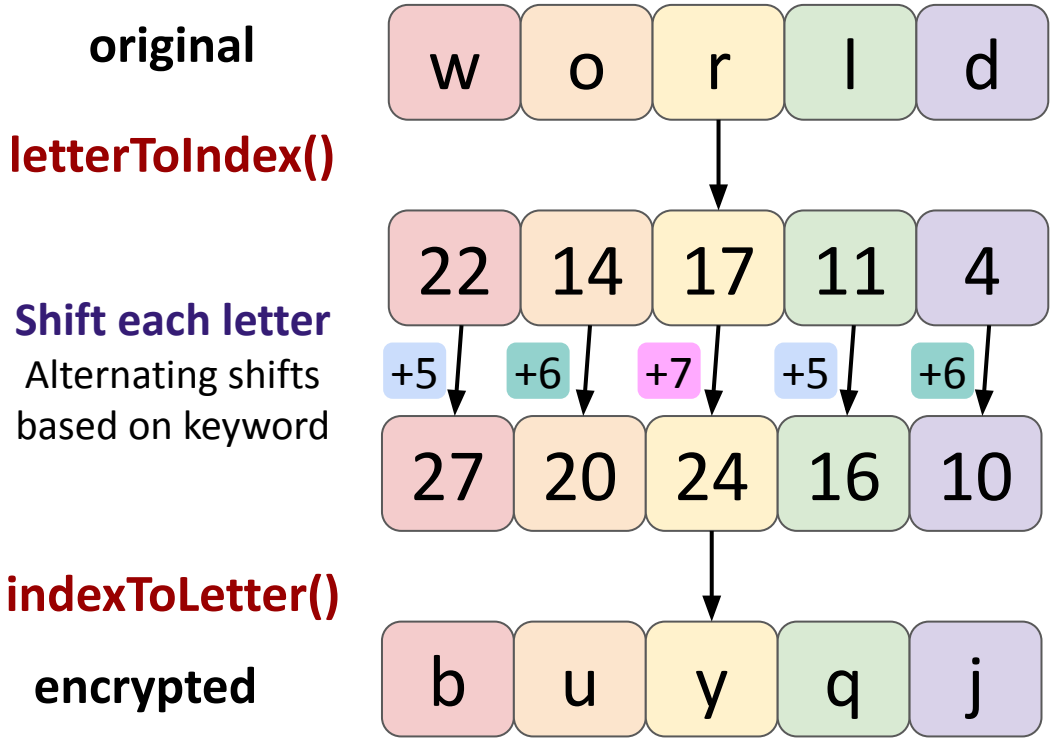


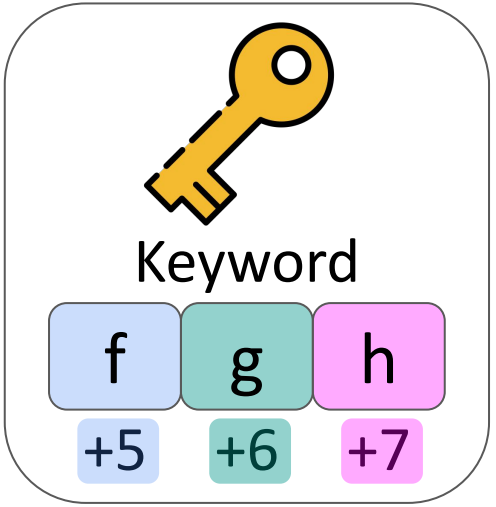




(Optional) Vigenère Ciphers

Feel free to try out this extension if you have time!





Remark: It's like having multiple Caesar ciphers in one encryption! A rotating set of keys.

Checkpoint #1 assignment.js

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.

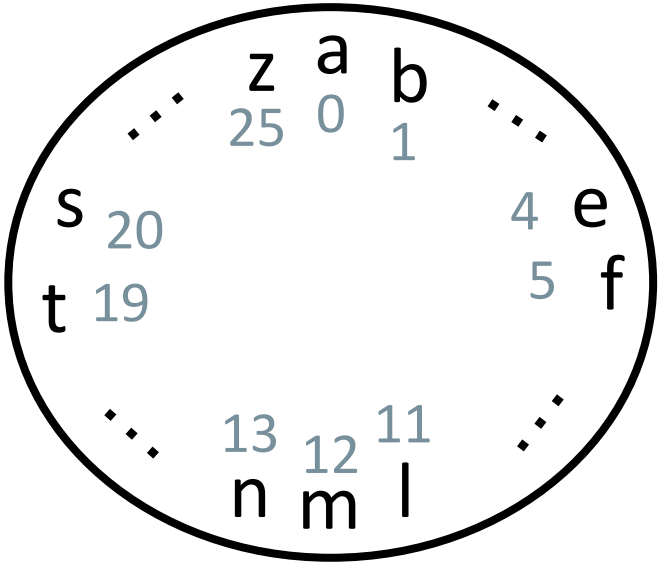
Note: After editing the JS file, make sure to click **File -> Save in VSCode,** and **refresh the Chrome page**, for the edits to manifest in the console!

Checkpoint #2 — assignment.js

Implement the function **indexToLetter()**

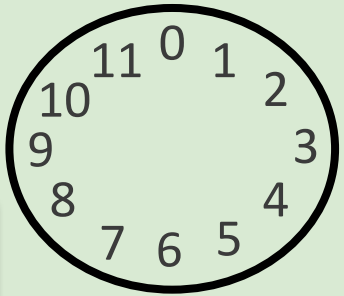
Input: Non-negative index of a letter, can be 26 and greater
Output: Corresponding lowercase letter; numbers above 25 wrap around
i.e. 0=a,1=b,..,25=z,26=a,27=b,...

Tip – The array **alphabet=['a','b',..,'z']** may come in handy.



For handling the wrap-around, consider the remainder operator (%) and this clock example. If it's 8:00 right now, then 7 hours later, it'll be 3:00, as times *wrap around* the 12-hour clock. This can be computed with the following JavaScript:

```
(8 + 7) % 12 // 8+7 => 15 o'clock 🤔
  // clock only has 12 hours (0-11), so 15%12 => 3:00
```



Checkpoint #3

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'
```

Full Encryption Pipeline

Final Checkpoint

Implement `encryptCaesar()`

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!

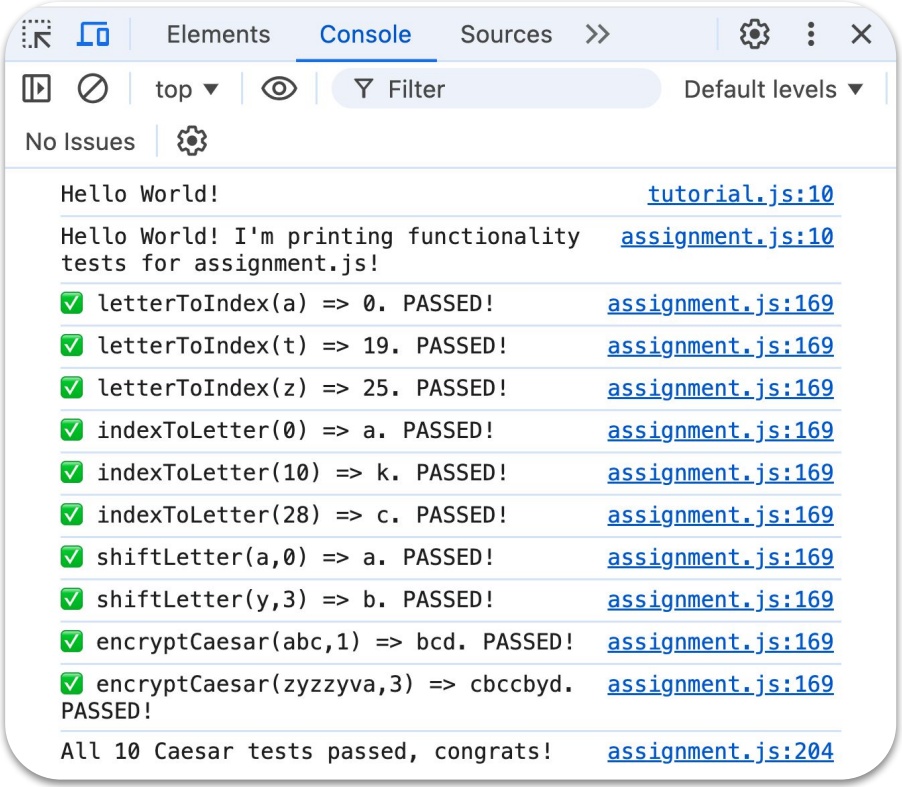
Example Functionality

JS Console

```
encryptCaesar('abc',1)
'bcd'

encryptCeasar('zyzzyva',3)
'cbccbyd'
```

Sanity Testing



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

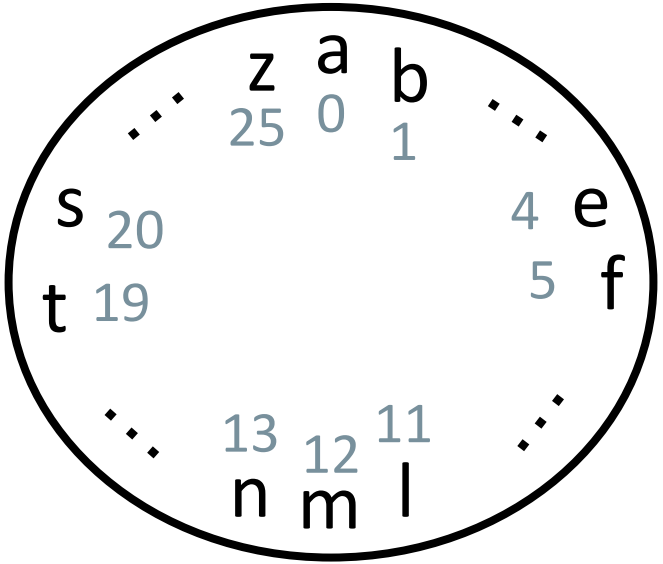
Solution code available on website right after class :)

To test Vigenère (optional!), go to the bottom of the **assignment.js** file, and uncomment the line that reads as:

```
// testVigenere() // uncomment to test Vigenere cipher (optional)
```

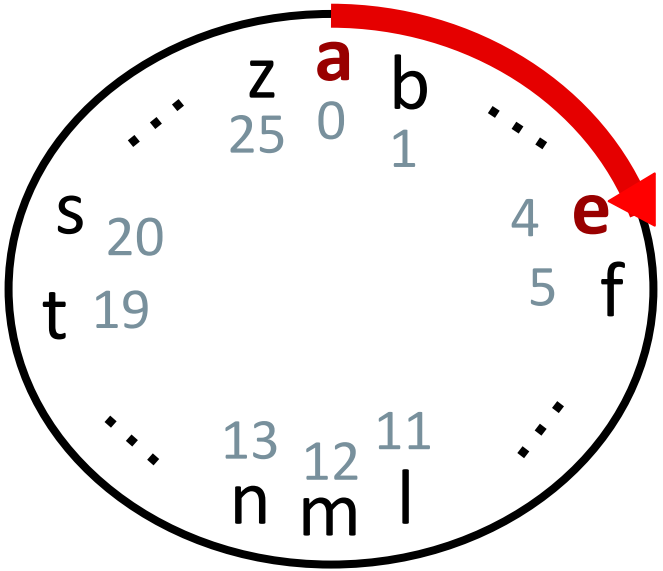
(Optional) Caesar Decryption

- 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?**

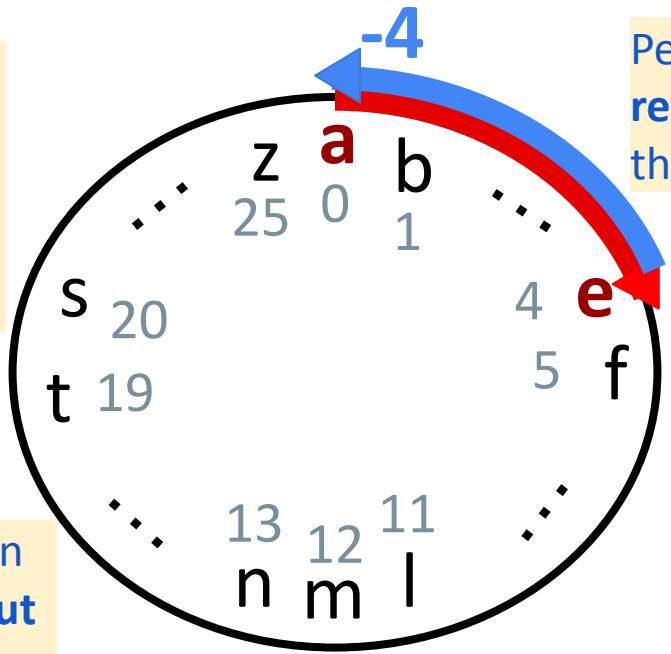


Suppose our cipher has a shift of **+4**, e.g., a => e. **How might we go back to letter ‘a’**?

(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 => a.

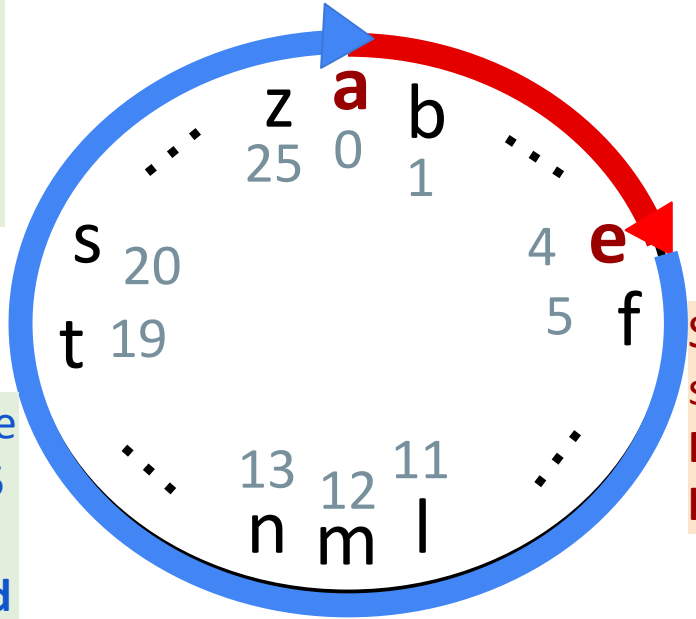
Suppose our cipher has a shift of **+4**, e.g., a => e. **How might we go back to letter ‘a’**?

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

(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.



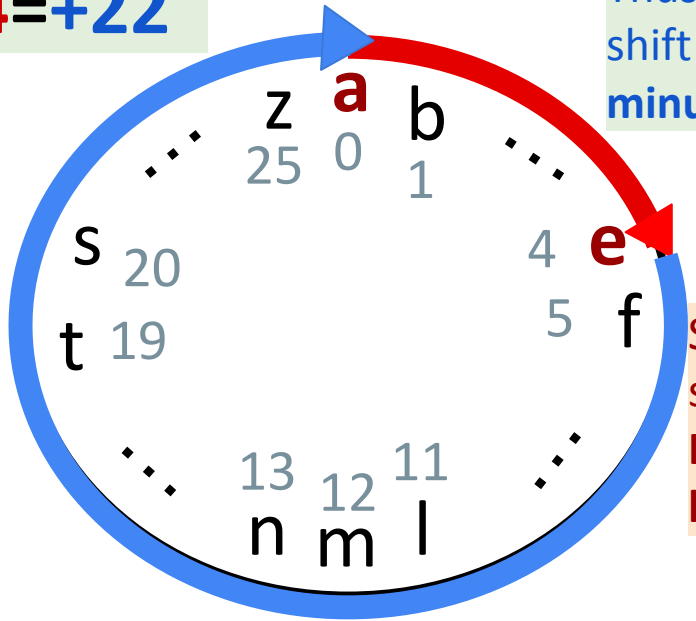
Suppose our cipher has a shift of **+4**, e.g., a => e. **How might we go back to letter ‘a’?**

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

(Optional) Caesar Decryption

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

26-4=+22



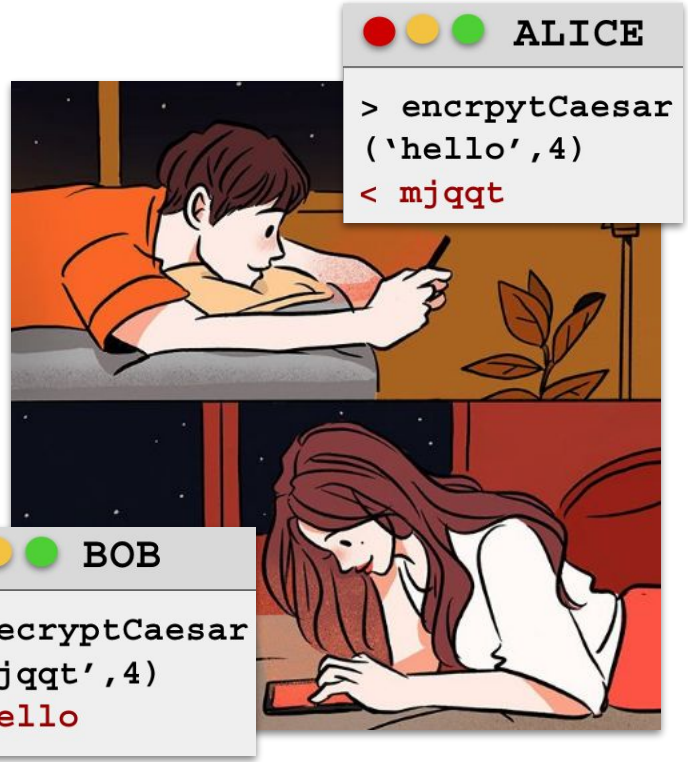
Thus, our decryption scheme will shift all letters by +22, or **26 minus the encryption shift.**

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

(Optional) Caesar Decryption

```
/* 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.



Check-Off Form!

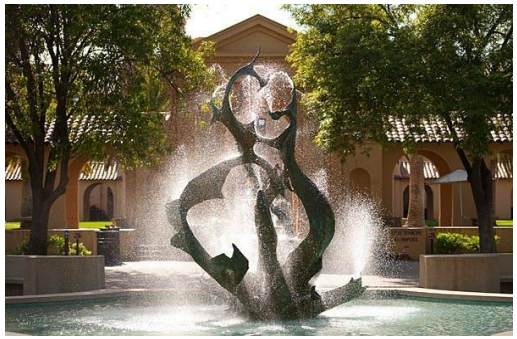
To get attendance credit each class, you'll fill out a **brief check-off form** (~2 – 5 min to complete).

For today, click the "Check-Off Form" link in the Week 1 section of **cs106s.stanford.edu!**

https://tinyurl.com/cs106s-win25-w1-checkoff (case sensitive!)

Looking Forward to this Winter 

Teaching this 1-unit wonder has been a truly wonderful joy and privilege for me, ~~because of all the free boba over the years~~; thank you for being here to learn with us, and I 👏 hope 👏 this 👏 will 👏 be 👏 fun 👏 for you!!



Have an awesome first week of classes! :)