

CS106AX Midterm Review

Autumn 2026

Eugene, Sabrina, and Ben!

October 25th, 2026

PDF of this presentation





Welcome!!





Today is all about midterm review

Are you ready? :)





FEAR NOT

LE JJERRYCAIN

Inspired by the Le Sserafim motto :)





You're going to do wonderful!

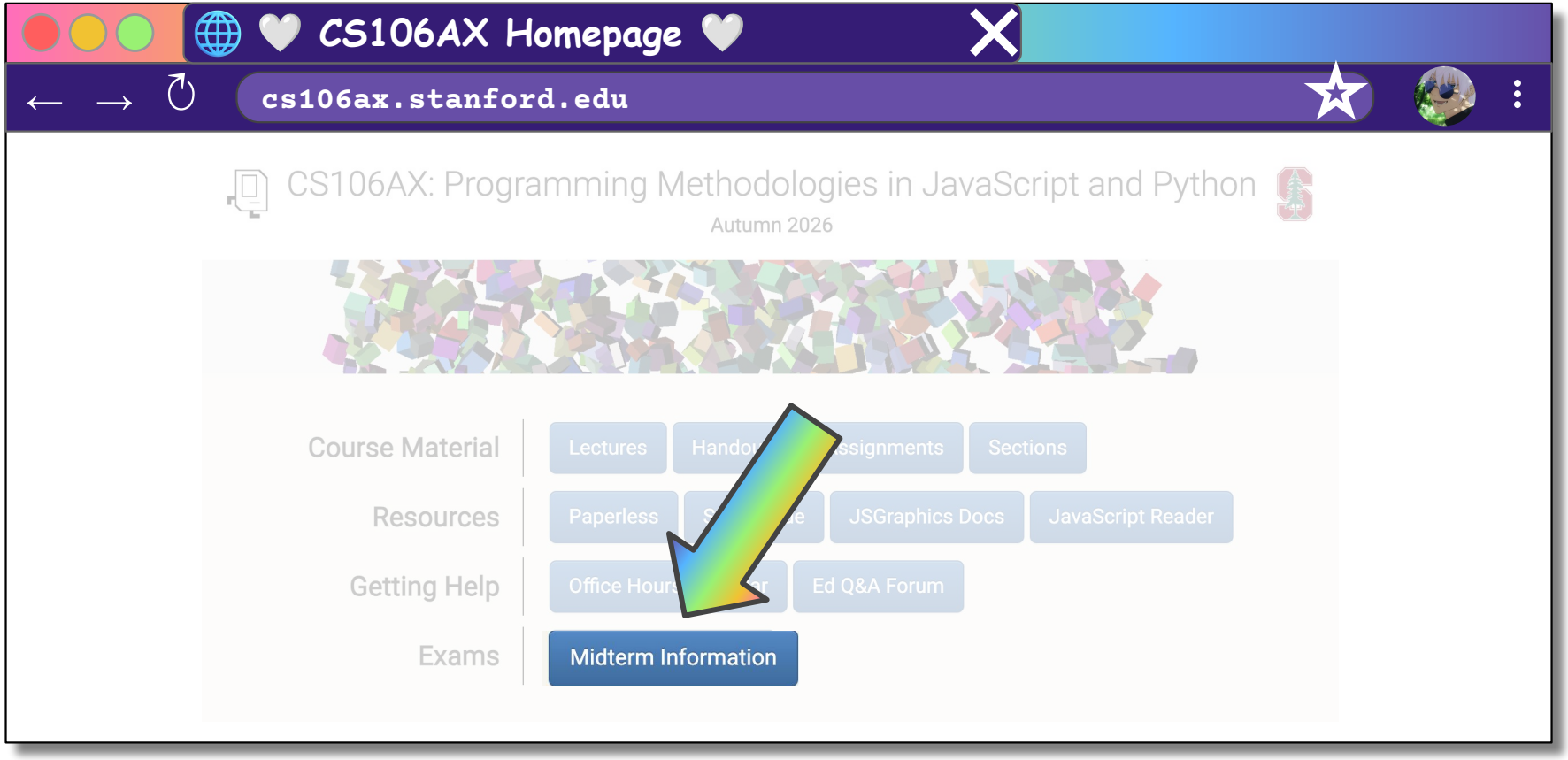
Keep up all the amazing work you've done all quarter!

Practice makes not perfect, but better!

🎯 Midterm Structure

Identically architected to Practice Midterms, as a gameboard of five problems. Like Breakout!, take down **one brick at a time**!

Problem 1	**JavaScript** expressions, statements, methods (short answer questions, function **scoping and trace problems**)
Problem 2 🎨	Using **graphics and animation** (multi-step program, be familiar with mouse event handlers and timers)
Problem 3	**Strings** (write a `function` to solve a strings task)
Problem 4	**Arrays** (write a `function` to solve an arrays task)
Problem 5	Working with **data structures** (write a `function` that handles data stored in JSON / **JS Object Notation** form)



cs106ax.stanford.edu

CS106AX Midterm Exam Information

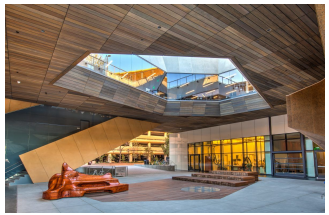
https://web.stanford.edu/class/cs106ax/exams/midterm.html

CS106AX Midterm Logistics

FIRST MIDTERM OFFERING: TUESDAY, OCTOBER 27TH FROM 3:30PM TO 5:30PM, MCMURTRY ART BUILDING 360

SECOND MIDTERM OFFERING: TUESDAY, OCTOBER 27TH FROM 7:00PM TO 9:00PM, BUILDING 370 ROOM 370



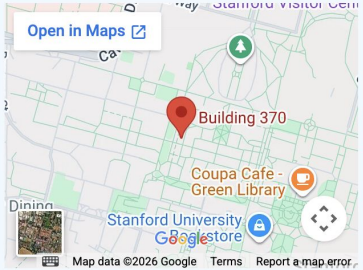


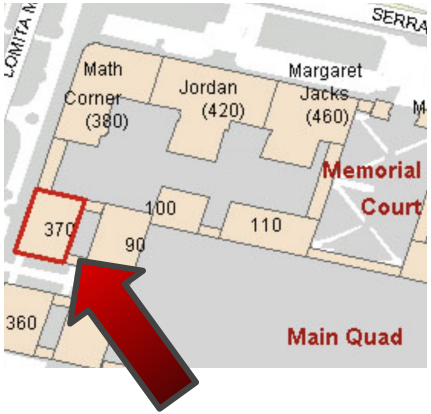
Midterm Exam Location

First Offering @ 3:30PM
McMurtry Art Building Room 360



Second Offering @ 7:00PM
Building 370, Room 370

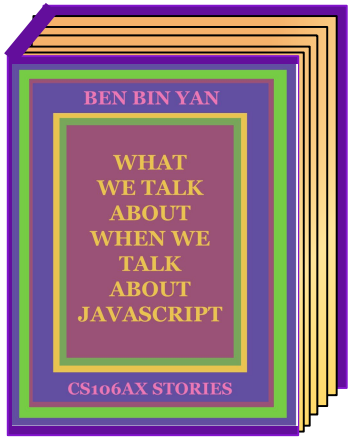




https://web.stanford.edu/class/cs106ax/exams/midterm.html

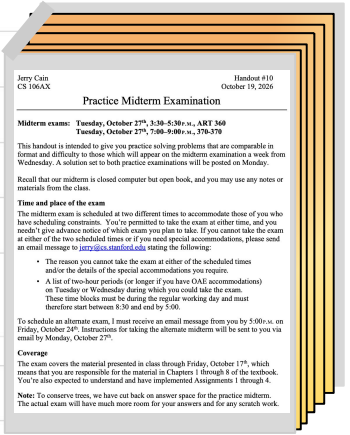
CS106AX Midterm Exam Information

https://web.stanford.edu/class/cs106ax/exams/midterm.html



✏️ Practice Exams and Problems	Download Links
Jerry's Practice Midterm	Download
Jerry's Practice Midterm Solutions	Download
Ben's Practice Midterm	Download
Ben's Practice Midterm Solutions	Download
Extra Midterm Practice	Download
Extra Midterm Practice Solutions	Download

📚 Study Resources	Download Links
Ben's JavaScript Course Reader	Download
JavaScript Library Documentation	JSGraphics RandomLib



Jerry Cain
CS 106AX

Handout #10
October 19, 2026

Practice Midterm Examination

Midterm exams: Tuesday, October 27th, 3:30–5:30P.M., ART 360
Tuesday, October 27th, 7:00–9:00P.M., 370-370

This handout is intended to give you practice solving problems that are comparable in format and difficulty to those which will appear on the midterm examination a week from Wednesday. A solution set to both practice examinations will be posted on Monday.

Recall that our midterm is closed computer but open book, and you may use any notes or materials from the class.

Time and place of the exam

The midterm exam is scheduled at two different times to accommodate those of you who have scheduling constraints. You're permitted to take the exam at either time, and you needn't give advance notice of which exam you plan to take. If you cannot take the exam at either of the two scheduled times or if you need special accommodations, please send an email message to jerry@cs.stanford.edu stating the following:

- The reason you cannot take the exam at either of the scheduled times and/or the details of the special accommodations you require.
- A list of two-hour periods (or longer if you have OAE accommodations) on Tuesday or Wednesday during which you could take the exam. These time blocks must be during the regular working day and must therefore start between 8:30 and end by 5:00.

To schedule an alternate exam, I must receive an email message from you by 5:00P.M. on Friday, October 24th. Instructions for taking the alternate midterm will be sent to you via email by Monday, October 27th.

Coverage

The exam covers the material presented in class through Friday, October 17th, which means that you are responsible for the material in Chapters 1 through 8 of the textbook. You're also expected to understand and have implemented Assignments 1 through 4.

Note: To conserve trees, we have cut back on answer space for the practice midterm. The actual exam will have much more room for your answers and for any scratch work.

https://web.stanford.edu/class/cs106ax/exams/midterm.html

Any questions before we get started?

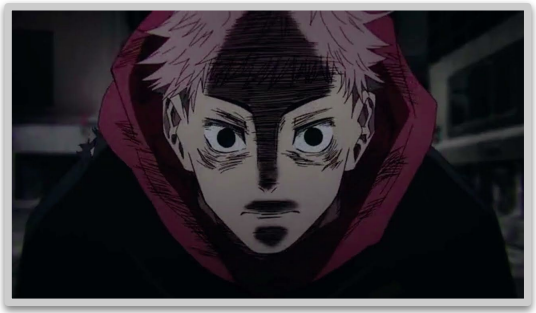






Alright, let's jump into the problems! :)



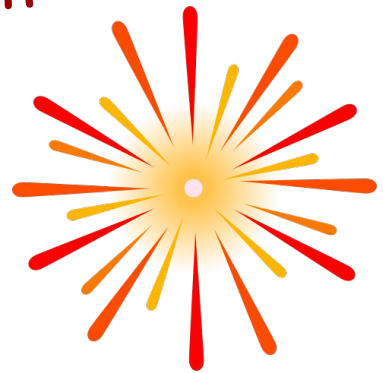




JavaScript Tracing & Expressions

Problem 1: Tracing That Dream
Ben's Practice Midterm





Problem 1: Tracing that Dream (JavaScript Expressions and Tracing)









1. [3 points] Compute the value of each of the following JavaScript expressions.

```
2026 + 3 % 2 * 2 - 2020
```

_____

```
"IDOL" === "idol" || "YUUSHA" < "SUKI" + "DA"
```

_____

```
6  + "SEIS" + 1 + 2 + 4 + "SIETE"
```

_____

2. [3 points] Assume the function **LoveLetter**, which accepts arrays **truth** and **obfuscation** as input, is defined as follows. Note that it returns a new array of identical length to **truth**.

```
function LoveLetter(truth, obfuscation) {
      let encrypted = [];
      let ambiguity = obfuscation.length;

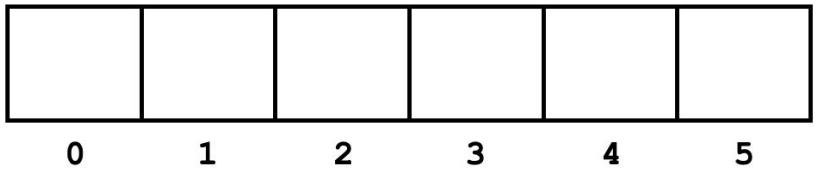
      for (let i = 0; i < truth.length; i++) {
            let mySecret = truth[i];
            let feelings = i % ambiguity;

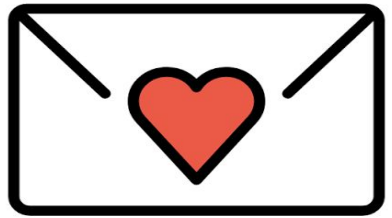
            let myWords = mySecret + obfuscation[feelings];
            if (myWords >= 26) myWords = myWords - 26;

            encrypted.push(myWords);
      }
      return encrypted;
}
```

Feel free to try this out in your own time!

What does the call **LoveLetter([2, 3, 5, 7, 11, 13], [17, 19, 23])** return? Work through this function trace carefully, and indicate the value of each of the elements in the returned array below.





3. [4 points] What output is produced by a call to the function **`DomainExpansion()`**?

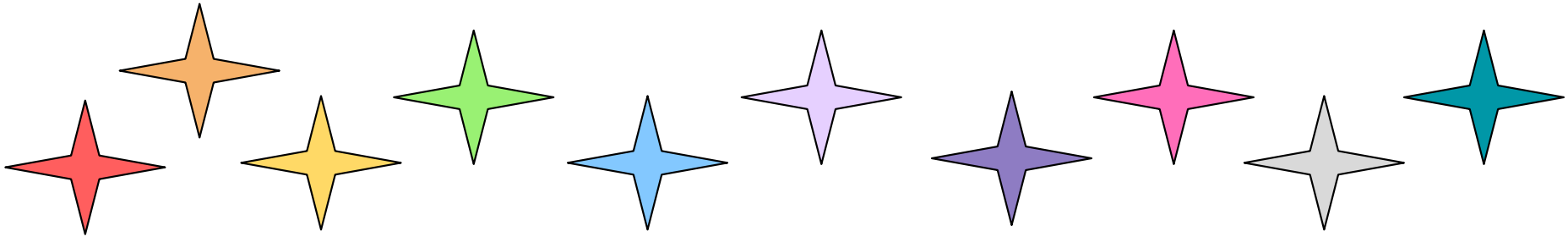


```
function DomainExpansion() {
      let cursed = "energy";
      let technique = function(blue, red, purple) {
            return cursed.substring(blue, red) + purple.substring(red);
      }
      let sorcery = Battle(technique, cursed.lastIndexOf("e"),
                                      cursed.indexOf("e"));
      sorcery.toLowerCase();
      console.log("This truly was our " + sorcery);
}
```

Feel free to try this out in your own time!

```
function Battle(cursedTech, positive, negative) {
      let culling = cursedTech(negative, positive, "jujutsu");
      let games = "shibuya".charCodeAt(5) - "arc".charCodeAt(0);
      let tokyo = "tombsOfStar".charAt(3); // not the same as charCodeAt!
      return tokyo.toUpperCase() + culling + games;
}
```

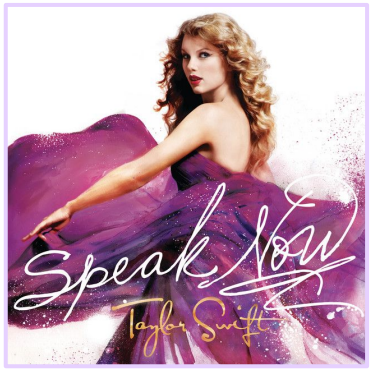




Graphics and Animation



Problem 2: Sparks Fly!
Ben's Practice Midterm

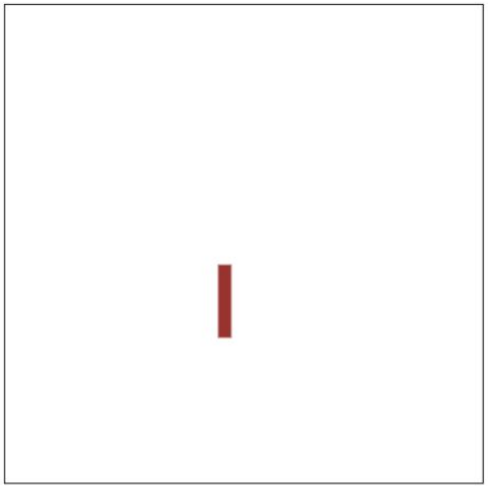


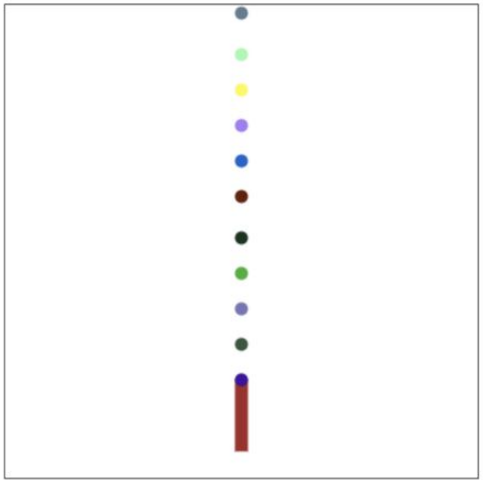


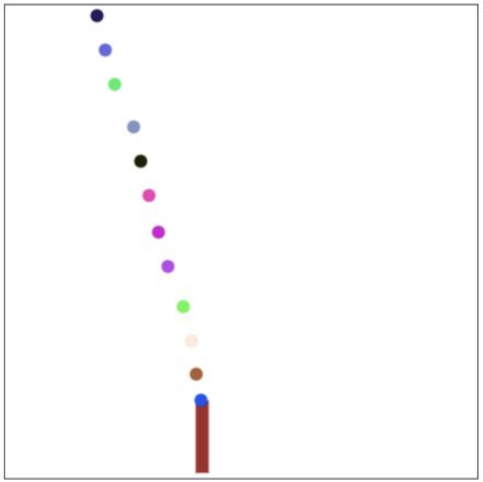
Problem 2: Sparks Fly (Graphics and Animation, 15 points)

"Cause I see sparks fly, whenever you smile" – Taylor Swift (song: Sparks Fly, album: Speak Now)

Implement a graphics program that introduces a magic wand to the canvas, designed as a rectangle with a **`"brown"`** fill color. It should have a height of 60 pixels, a width of 10 pixels, and starts with its top-left corner at the origin **`(0,0)`**. The wand should follow the mouse cursor in both directions so that its top-left corner attaches to the mouse's current coordinate with each **`"mousemove"`** event. Unlike what was required of you in **`Breakout`**, it's perfectly ok if the wand leaves the canvas when your mouse does.







When the user **clicks** anywhere on screen for the first time, sparks fly up from the wand and **float upwards** to the top of the window, moving 10 pixels every 10 milliseconds. The sparks are randomly colored circles with a 10-pixel diameter (notably identical to the wand's width), and one is introduced every 50 milliseconds. At the time of its creation, the spark will share the same top-left coordinate as the wand. When a spark moves off the top of the window—so that it's no longer visible—you should both remove the spark from the window, and terminate that spark's flying animation with **`clearInterval`**.

When the user **clicks** anywhere a second time, the wand **stops** emitting colorful sparks, but any existing sparks will continue flying up. A third click will cause sparks to be sprayed again, a fourth click stops the spraying, and so forth, so that **a `"click"` event effectively toggles the wand** between emitting and not emitting sparks. Along with event listeners to track the mouse (moves and clicks), your approach will require nested interval timers to support two animations: one to create each flying spark, and an inner one to keep moving that particular spark upward until it soars out of view.

You'll likely need some way of tracking whether sparks are currently flying or not, and a way to start and stop the animation of sparks being created, e.g., methods **`setInterval`** and **`clearInterval`**.

Present your implementation of **`sparksFly`**, and callback functions, on the next page. Along with timer functions, you may find the methods **`randomColor`**, **`setLocation`**, and **`addEventListener`** useful.

vim SparksFly.js

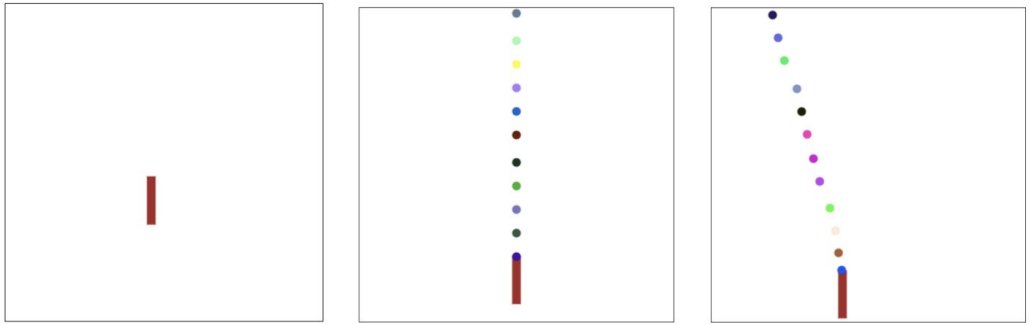
```
/* Constants (in pixels */
const GWINDOW_HEIGHT = 400;
const GWINDOW_WIDTH = 400;
const WAND_WIDTH = 10;
const WAND_HEIGHT = 60;
const SPARK_Y_VELOCITY = -10;
const SPARK_DIAMETER = 10;

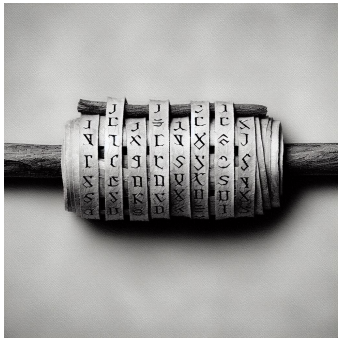
/* Constants (in milliseconds) */
const FLYING_TIME_STEP = 10;
const CREATION_TIME_STEP = 50;

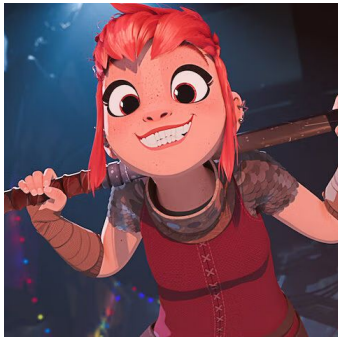
/* Program where you get to do your magic a la wizards of waverly place */
function sparksFly(){
      let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);

      TODO: Fill this in!

}
```





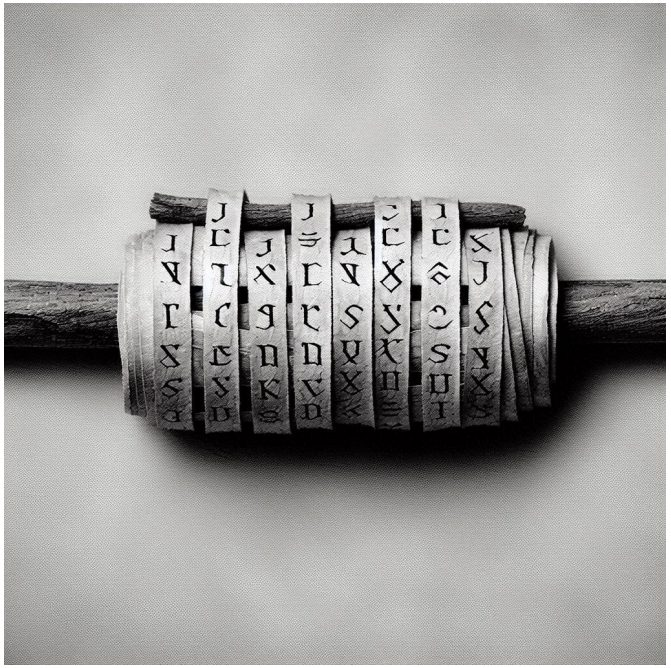




JavaScript Strings

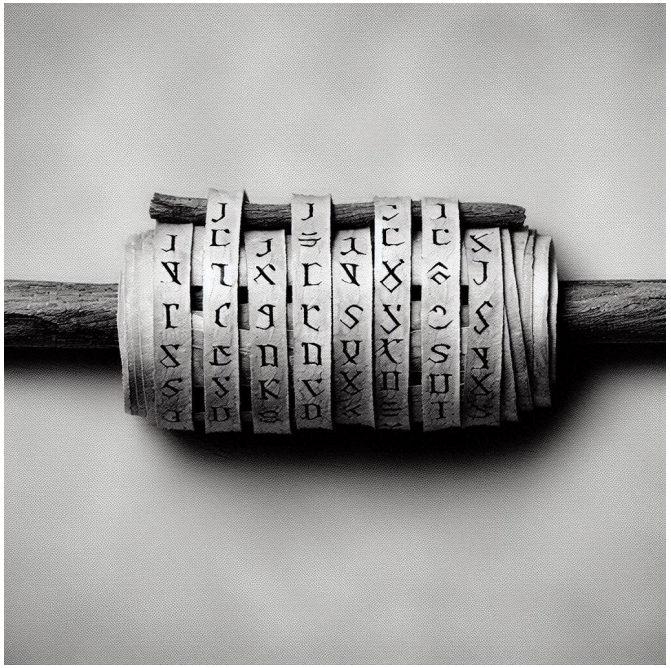
Problem 3: Shape-Shifting Cipher
Ben's Practice Midterm



Scytale?



Scytale ✅



Scytale???



What is the Scytale Cipher?

Inputs: **plaintext**, number of **columns**



What is the Scytale Cipher?

Inputs: **plaintext**, number of **columns**

plaintext: "**DADDYISBACKTELLYOURMOMMY**"
columns: **5**



What is the Scytale Cipher?

Inputs: **plaintext**, number of **columns**

plaintext: "DADDYISBACKTELLYOURMOMMY"
columns: 5

Step 1: Write the plaintext onto a grid with that many columns, padding the text with "**X**"s as needed to fill up the last row



What is the Scytale Cipher?

Step 1: Write the plaintext onto a grid with that many columns, padding the text with "**X**"s as needed to fill up the last row

columns = 5

D	A	D	D	Y
I	S	B	A	C
K	T	E	L	L
Y	O	U	R	M
O	M	M	Y	



What is the Scytale Cipher?

Step 1: Write the plaintext onto a grid with that many columns, padding the text with "**X**"s as needed to fill up the last row

columns = 5

D	A	D	D	Y
I	S	B	A	C
K	T	E	L	L
Y	O	U	R	M
O	M	M	Y	X



What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

columns = 5

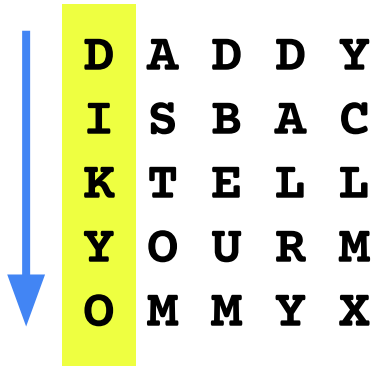
D A D D Y
I S B A C
K T E L L
Y O U R M
O M M Y X



What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

ciphertext = "DIKYO"

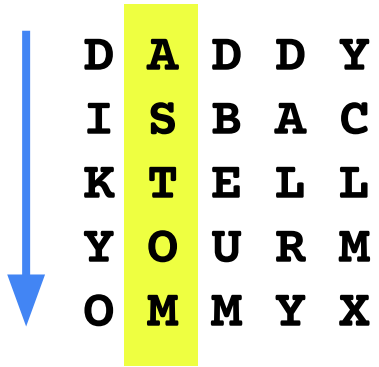




What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

ciphertext = "DIKYOASTOM"

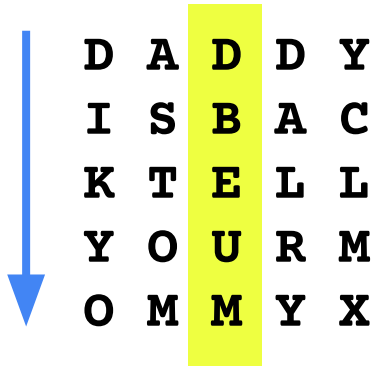




What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

ciphertext = "DIKYOASTOMDBEUM"

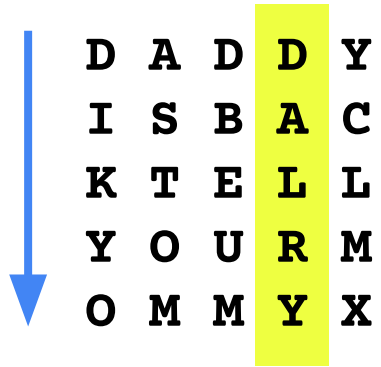




What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

ciphertext = "DIKYOASTOMDBEUMDALRY"

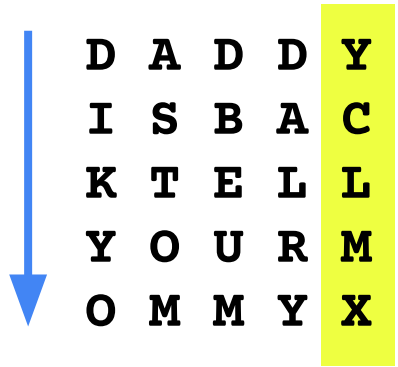




What is the Scytale Cipher?

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

ciphertext = "DIKYOASTOMDBEUMDALRYYCLMX"





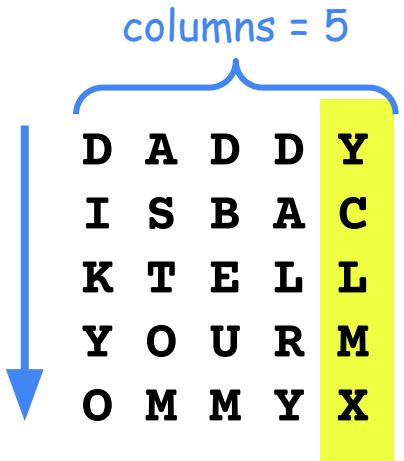
Summary

Inputs: **plaintext**, number of **columns**

Step 1: Write the plaintext onto a grid with that many columns, padding the text with "**X**"s as needed to fill up the last row

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

plaintext: "**DADDYISBACKTELLYOURMOMMY**"
columns: 5



ciphertext: "DIKYOASTOMDBEUMDALRYYCLMX"



Summary

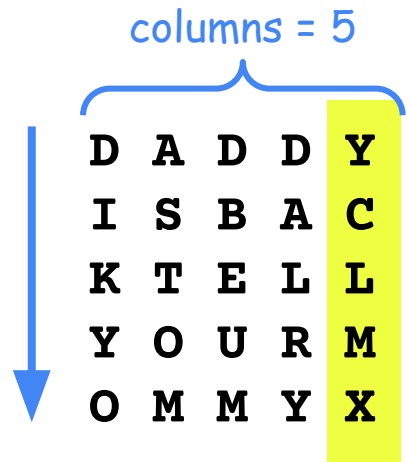
No arrays or maps! Only strings…

Inputs: **plaintext**, number of **columns**

Step 1: Write the plaintext onto a grid with that many columns, padding the text with "**X**"s as needed to fill up the last row

Step 2: To produce the ciphertext, read **top-to-bottom first**, then **left-to-right**

plaintext: "DADDYISBACKTELLYOURMOMMY"
columns: 5



ciphertext: "DIKYOASTOMDBEUMDALRYYCLMX"



JavaScript Arrays

Problem 4: U.S. National Debt
Extra Midterm Practice Problems



Handing baton to Eugene!



Problem 5, Ben's Extra Practice Midterm

You've been looking at the U.S. national debt and…

```
37,918,570,029,078
```

wow that is a big number!

Problem 5, Ben's Extra Practice Midterm

You've been looking at the U.S. national debt and…

```
37,918,570,029,078
```

wow that is a big number!

So you've decided to implement **Big Numbers** in JavaScript which will allow you to represent arbitrarily large numbers.

Big Numbers are numbers that we store using an array.

Important: our numbers are stored in order of increasing digits.

Big Numbers are numbers that we store using an array.

Important: our numbers are stored in order of increasing digits.

The number `37,918,570,029,078` becomes the array

```
[8, 7, 0, 9, 2, 0, 0, 7, 5, 8, 1, 9, 7, 3]
```

Big Numbers are numbers that we store using an array.

Important: our numbers are stored in order of increasing digits.

The number `37,918,570,029,078` becomes the array



This has 14 digits

```
[8, 7, 0, 9, 2, 0, 0, 7, 5, 8, 1, 9, 7, 3]
```



So this has 14 elements

We will implement a function to add two *Big Numbers* together

We will implement a function to add two *Big Numbers* together

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
[2, 1, 5, 2]
```

We will implement a function to add two *Big Numbers* together

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
[2, 1, 5, 2]
```

Because 2051 + 461 = 2512

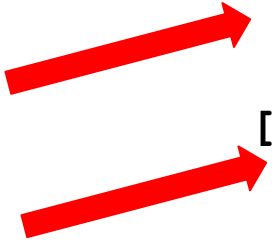
We will implement a function to add two *Big Numbers* together

let's go step by step

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
      []
```

We will implement a function to add two *Big Numbers* together

first the ones digit



```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
     []
```

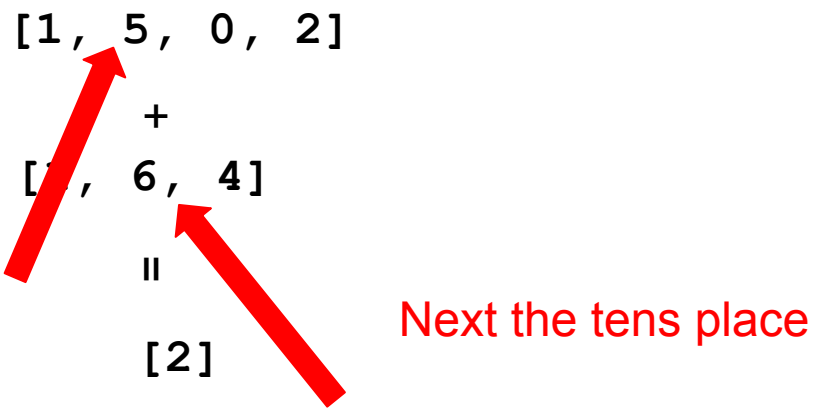
We will implement a function to add two *Big Numbers* together

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
     [2]
```



1 + 1 = 2

We will implement a function to add two *Big Numbers* together



We will implement a function to add two *Big Numbers* together

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
  [2, 11?]
```



5 + 6 = 11?

We will implement a function to add two *Big Numbers* together

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
  [2, 11?]
```



We need to split this up!

We will implement a function to add two *Big Numbers* together

```
carry = 1
```

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
   [2, 1]
```

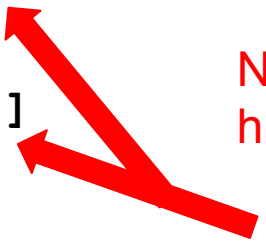


And then we “remember” we carried a 1.

We will implement a function to add two *Big Numbers* together

```
carry = 1
```

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
   [2, 1]
```



Now we move on to hundreds

We will implement a function to add two *Big Numbers* together

`carry = 1`

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
  [2, 1, 5]
```

Note this is 1 + 0 + 4



Now we move on to hundreds

We will implement a function to add two *Big Numbers* together

```
carry = 0
```

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
 [2, 1, 5]
```



Finally, we do thousandths

We will implement a function to add two *Big Numbers* together

```
carry = 0
```

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
 [2, 1, 5]
```



The second number doesn't have a thousandths place

We will implement a function to add two *Big Numbers* together

```
carry = 0
```

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
[2, 1, 5, 2]
```



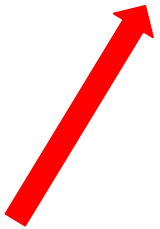
So we can just get 2 + 0

We will implement a function to add two *Big Numbers* together

`carry = 0`

```
[1, 5, 0, 2]
     +
[1, 6, 4]
     =
[2, 1, 5, 2]
```

2512 is our answer

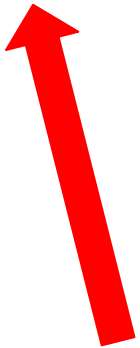


To recap:

- We add two *big numbers* digit by digit.
- When a sum exceeds 9, we set a *carry flag* to add an extra one on the next sum.

```
function addBigNumbers(bigNumOne, bigNumTwo) {
    let sum = [];
}
```

```
function addBigNumbers(bigNumOne, bigNumTwo) {
    let sum = [];
    let carryFlag = 0;
}
```



Remember we need that carry flag
to tell us when to add an extra one!

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);
    }
}
```

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);
    }
}
```



For now, just assume this gives you the ith digit of the big number.

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;
    }
}
```



We might have a carry flag to add.

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;
    }
}
```

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;
    }
}
```



Remember to (re)set the carry flag if we (don’t) have to carry.

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }
}
```

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }
}
```

Why the %?



```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }
}
```



Because if the digitSum is too big we can only take the first digit!

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }

    if (carryFlag) sum.push(carryFlag);
}
```



Edge case! A carry flag at the end means we need to add 1.

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }

    if (carryFlag) sum.push(carryFlag);
    return sum;
}
```

```
function addBigNumbers(bigOne, bigTwo) {
    let sum = [];
    let carryFlag = 0;

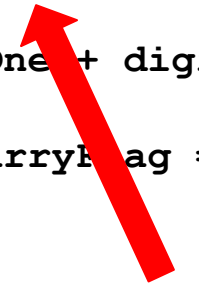
    for (let i = 0; i<Math.max(bigOne.length, bigTwo.length; i++){
        let digitOne = getDigitSafe(bigOne, i);
        let digitTwo = getDigitSafe(bigTwo, i);

        let digitSum = digitOne + digitTwo + carryFlag;

        if (digitSum <= 9) carryFlag = 0;
        else carryFlag = 1;

        sum.push(digitSum % 10);
    }

    if (carryFlag) sum.push(carryFlag);
    return sum;
}
```



Rewind: What was this about?

Remember in this example, the numbers had different number of digits?

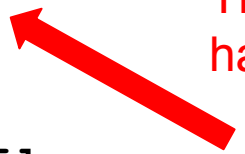
bigOne = `[1, 5, 0, 2]`

\+

bigTwo = `[1, 6, 4]`

=

`[2, 1, 5]`



The second number doesn't have a thousandths place

Remember in this example, the numbers had different number of digits?

bigOne = `[1, 5, 0, 2]`

\+

bigTwo = `[1, 6, 4, 0]`

=

`[2, 1, 5]`



We can pretend there is a 0 here.

bigOne = `[1, 5, 0, 2]`

\+

bigTwo = `[1, 6, 4, 0]`

=

`[2, 1, 5]`

Then `getDigitSafe(bigTwo, 3) = 0`

bigOne = `[1, 5, 0, 2]`

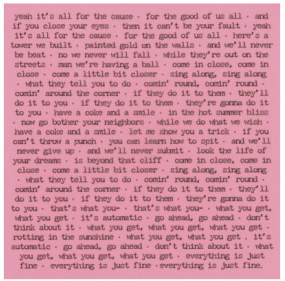
\+

bigTwo = `[1, 6, 4, 0]`

=

`[2, 1, 5]`

```
function getDigitSafe(bigNum, index) {
    if (index < bigNum.length) {
        return bigNum[index];
    }
    return 0;
}
```



yeah it's all for the cause · for the good of us all · and if you close your eyes · then it can't be your fault · yeah it's all for the cause · for the good of us all · here's a tower we built · painted gold on the walls · and we'll never be beat · no we never will fall · while they're out on the streets · man we're having a ball · come in close, come in close · come a little bit closer · sing along, sing along · what they tell you to do · comin' round, comin' round · comin' around the corner · if they do it to them · they'll do it to you · if they do it to them · they're gonna do it to you · have a coke and a smile · in the hot summer bliss · now go botzer your neighbors · while we do what we wish · have a coke and a smile · let me show you a trick · if you can't throw a punch · you can learn how to spit · and we'll never give up · and we'll never submit · look the life of your dreams · is beyond that cliff · come in close, come in close · come a little bit closer · sing along, sing along · what they tell you to do · comin' round, comin' round · comin' around the corner · if they do it to them · they'll do it to you · if they do it to them · they're gonna do it to you · that's what you · that's what you · what you get, what you get · it's automatic · go ahead, go ahead · don't think about it · what you get, what you get, what you get · what you get · rotting in the sunshine · what you get, what you get · it's automatic · go ahead, go ahead · don't think about it · what you get, what you get, what you get · everything is just fine · everything is just fine · everything is just fine.







Working with Data Structures



Problem 5: Spotify Wrapped!
Ben's Practice Midterm



How You Like That That That That



```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love": {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

    "serena joy": {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "serena joy", "streams": 50},
        ]
    },
```

```
    "DeBI TiRAR Más FOToS": {
        "artist": "Bad Bunny",
        "year": 2025,
        "songs": [
            {"title": "EoO", "streams": 80},
            {"title": "DtMF", "streams": 20},
            {"title": "NUEVAYoL", "streams": 20},
        ] },

    "Nevermind": {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
}
```

In this (teen) spirit, implement the function **`MusicWrapped`**, which accepts as input a data structure **`streamsByAlbum`** like the one above, and returns an object with two keys, **`topArtist`** and **`topSong`**. These map to our most streamed artist and song, respectively. For the example above, we'd return

```
{
      "topArtist": "Olivia Rodrigo",
      "topSong": "Smells Like Teen Spirit"
}
```



You may assume that **`streamsByAlbum`** contains at least one album, and at each album contains at least one song. If there are two or more artists tied for most streamed, you can return any one of them, and the same applies for songs. And as an implementation hint, **to identify the top artist, you'll almost certainly want to build up a JavaScript map of each artist to the total number of streams across all their songs**.

Present your implementation of **`MusicWrapped`** on the next page.

For the sake of time, and because the solution is way longer than I intended oops :(, just **find and return the most streamed artist, e.g., "Olivia Rodrigo"** for the provided **`streamsByAlbum`** example.



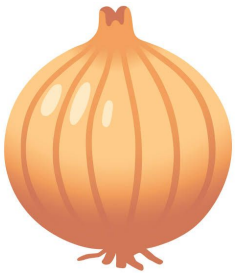




Let's drill into this nested structure!

Go one layer at a time, like peeling an onion

Fsck it





Let's cook!

```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love": {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

    "Nevermind": {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
```

```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love": {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

    "Nevermind": {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
```

Object

Collection of Key-Value Pairs!

```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love" : {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

    "Nevermind" : {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
```

```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love" : {
        "artist": "Olivia Rodrigo",
        "year": 2026,
        "songs": [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

    "Nevermind" : {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
```

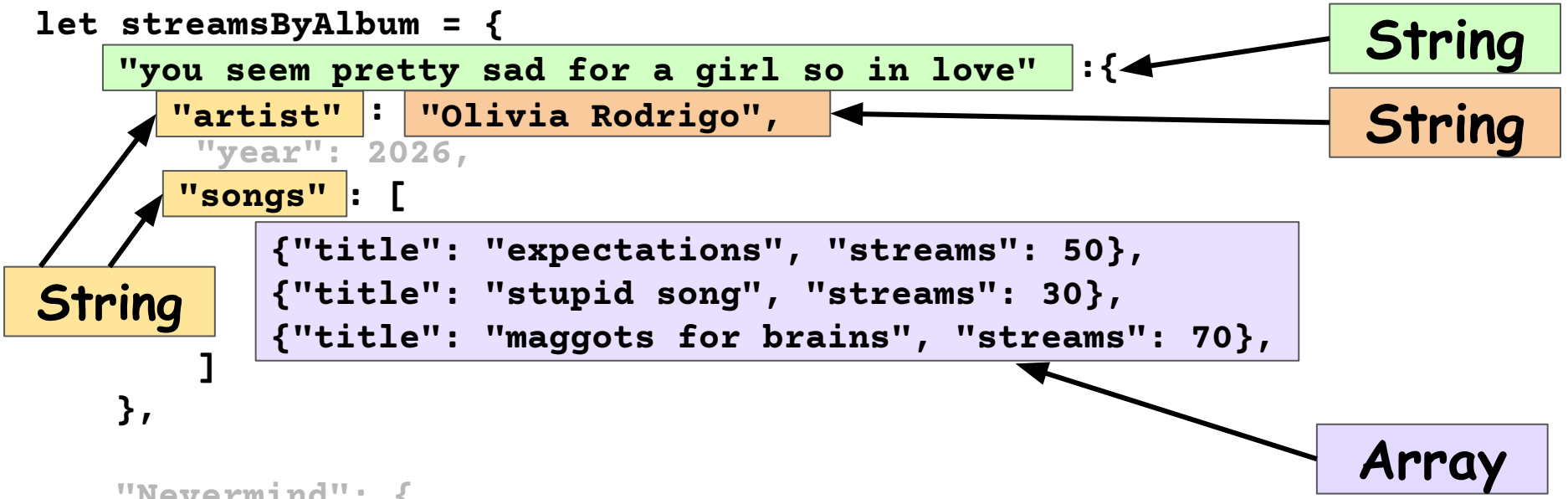
String

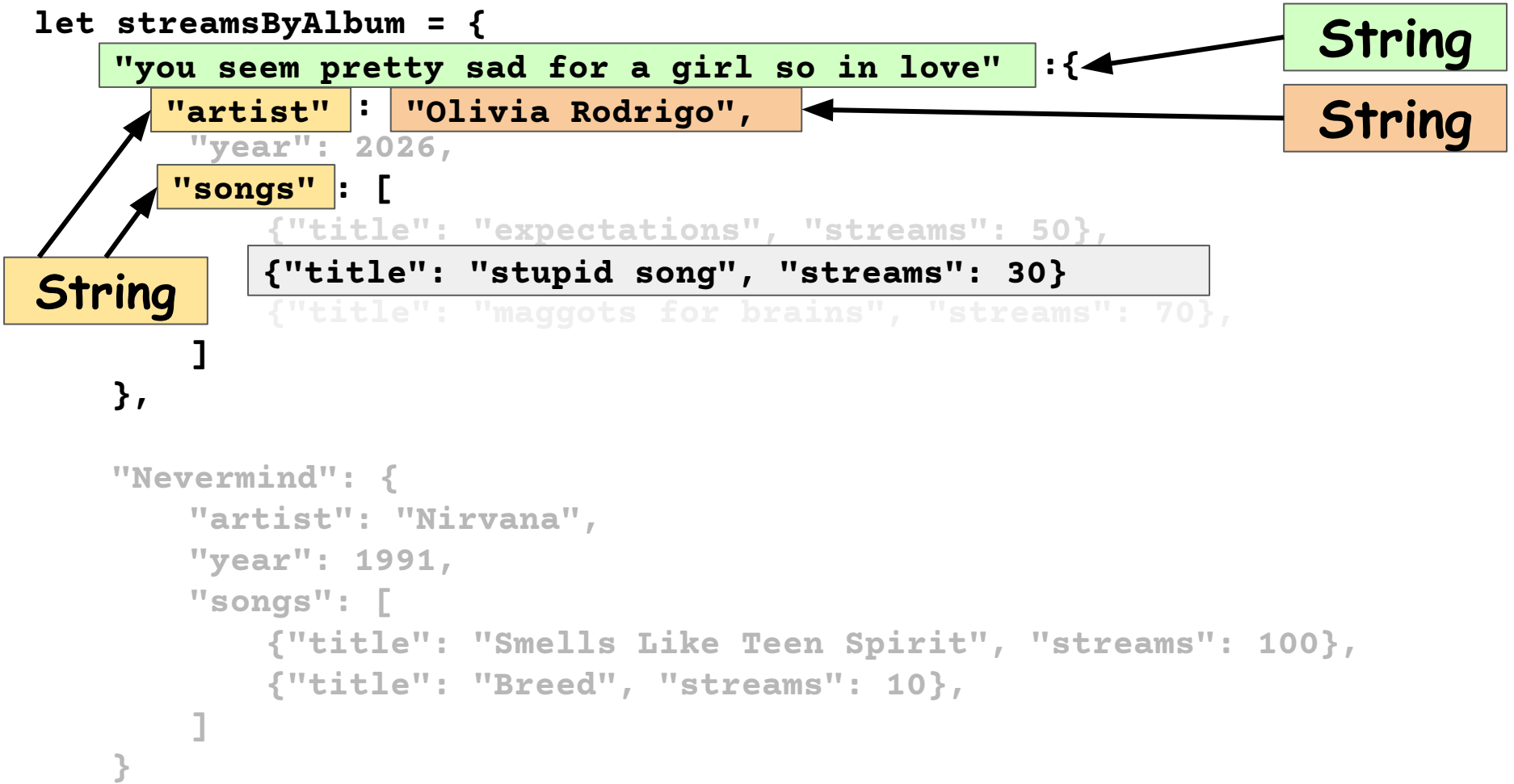
Object

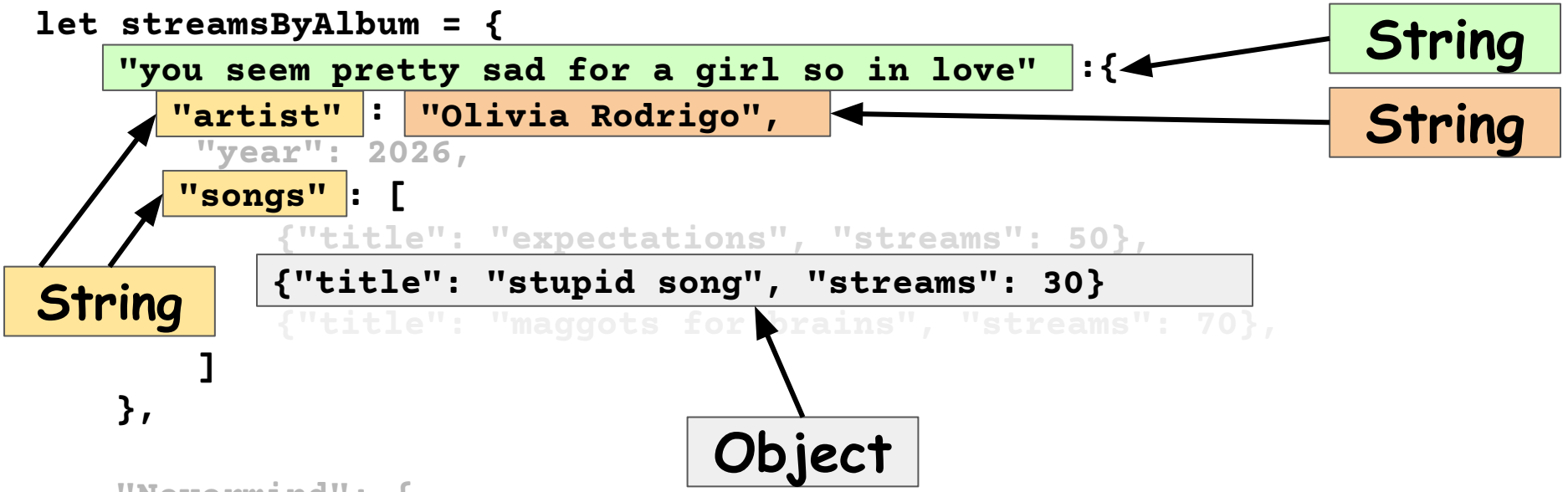
```
let streamsByAlbum = {
    "you seem pretty sad for a girl so in love" :{
        "artist" : "Olivia Rodrigo",
        "year": 2026,
        "songs" : [
            {"title": "expectations", "streams": 50},
            {"title": "stupid song", "streams": 30},
            {"title": "maggots for brains", "streams": 70},
        ]
    },

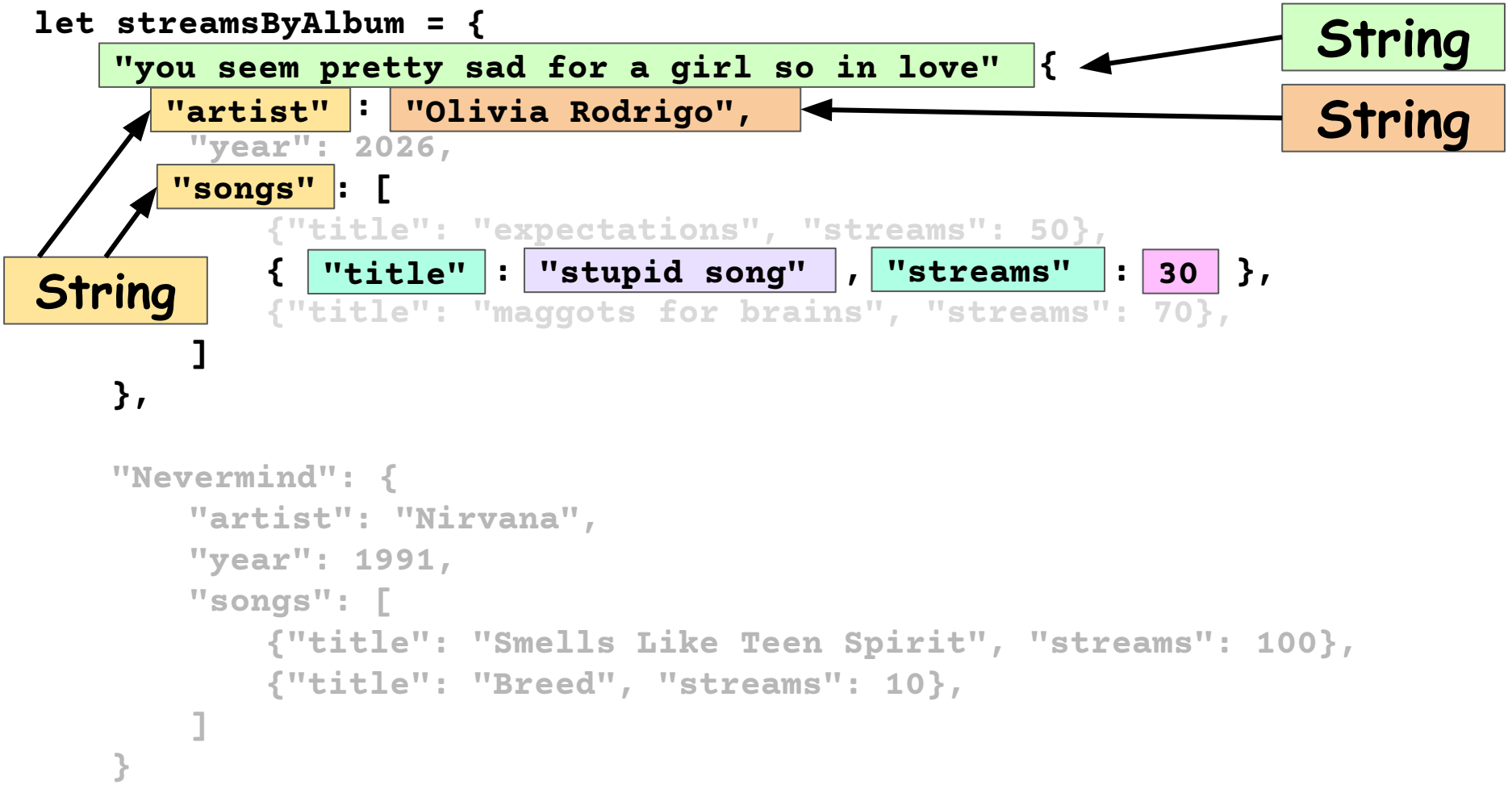
    "Nevermind": {
        "artist": "Nirvana",
        "year": 1991,
        "songs": [
            {"title": "Smells Like Teen Spirit", "streams": 100},
            {"title": "Breed", "streams": 10},
        ]
    }
```

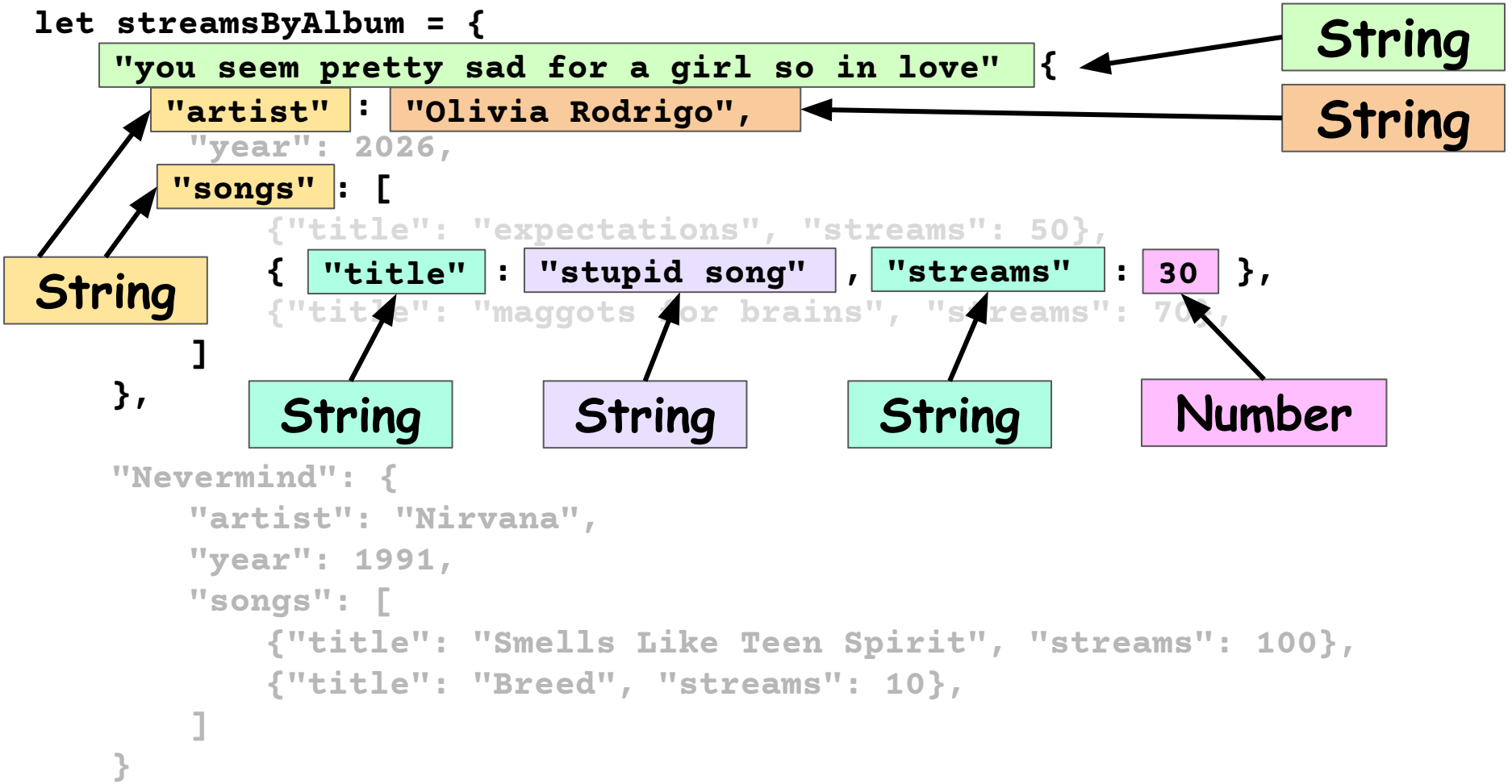
String


let streamsByAlbum = {
"you seem pretty sad for a girl so in love" :{
"artist" : "Olivia Rodrigo",
"year": 2026,
"songs" : [
{"title": "expectations", "streams": 50},
{"title": "stupid song", "streams": 30},
{"title": "maggots for brains", "streams": 70},
]
},
"Nevermind": {
"artist": "Nirvana",
"year": 1991,
"songs": [
{"title": "Smells Like Teen Spirit", "streams": 100},
{"title": "Breed", "streams": 10},
]
}
String
String
String
Array


let streamsByAlbum = {
"you seem pretty sad for a girl so in love" :{
"artist" : "Olivia Rodrigo",
"year": 2026,
"songs" : [
{"title": "expectations", "streams": 50},
{"title": "stupid song", "streams": 30}
{"title": "maggots for brains", "streams": 70},
]
},
"Nevermind": {
"artist": "Nirvana",
"year": 1991,
"songs": [
{"title": "Smells Like Teen Spirit", "streams": 100},
{"title": "Breed", "streams": 10},
]
}
String
String
String


let streamsByAlbum = {
"you seem pretty sad for a girl so in love" :{
"artist" : "Olivia Rodrigo",
"year": 2026,
"songs" : [
{"title": "expectations", "streams": 50},
{"title": "stupid song", "streams": 30}
{"title": "maggots for brains", "streams": 70},
]
},
"Nevermind": {
"artist": "Nirvana",
"year": 1991,
"songs": [
{"title": "Smells Like Teen Spirit", "streams": 100},
{"title": "Breed", "streams": 10},
]
}
String
String
String
Object


let streamsByAlbum = {
"you seem pretty sad for a girl so in love" {
"artist" : "Olivia Rodrigo",
"year": 2026,
"songs" : [
{"title": "expectations", "streams": 50},
{ "title" : "stupid song" , "streams" : 30 },
{"title": "maggots for brains", "streams": 70},
]
},
"Nevermind": {
"artist": "Nirvana",
"year": 1991,
"songs": [
{"title": "Smells Like Teen Spirit", "streams": 100},
{"title": "Breed", "streams": 10},
]
}
String
String
String


let streamsByAlbum = {
"you seem pretty sad for a girl so in love" {
"artist" : "Olivia Rodrigo",
"year": 2026,
"songs" : [
{"title": "expectations", "streams": 50},
{ "title" : "stupid song" , "streams" : 30 },
{"title": "maggots for brains", "streams": 70},
]
},
"Nevermind": {
"artist": "Nirvana",
"year": 1991,
"songs": [
{"title": "Smells Like Teen Spirit", "streams": 100},
{"title": "Breed", "streams": 10},
]
}
String
String
String
String
String
String
String
Number

Closing Remarks

📓 **Check out the Midterm Info Page** on the course website—with many problems to practice with!

🖌️ **Besides the practice exams,** you can look over section handouts and your IG problems to enrich understanding, as well as course assignment code!



💬 **Please feel free to post on EdStem forum with any questions!** (practice exam/solution clarifications, conceptual questions, exam tips, etc.)

Happy to take questions after the review session!!

You can do this! 💖

Good luck, you're going to do amazing!





