

# J220

## Coding for Journalists

LECTURER  
Soo Oh

### PROMPTS

Download files from  
<https://journ220.github.io>

Sign into  
<https://pollev.com/soooh>

Zoom screenshare +  
start Zoom recording

# Agenda

Announcements

Homework review + how much time

JavaScript

**BREAK**

Activity

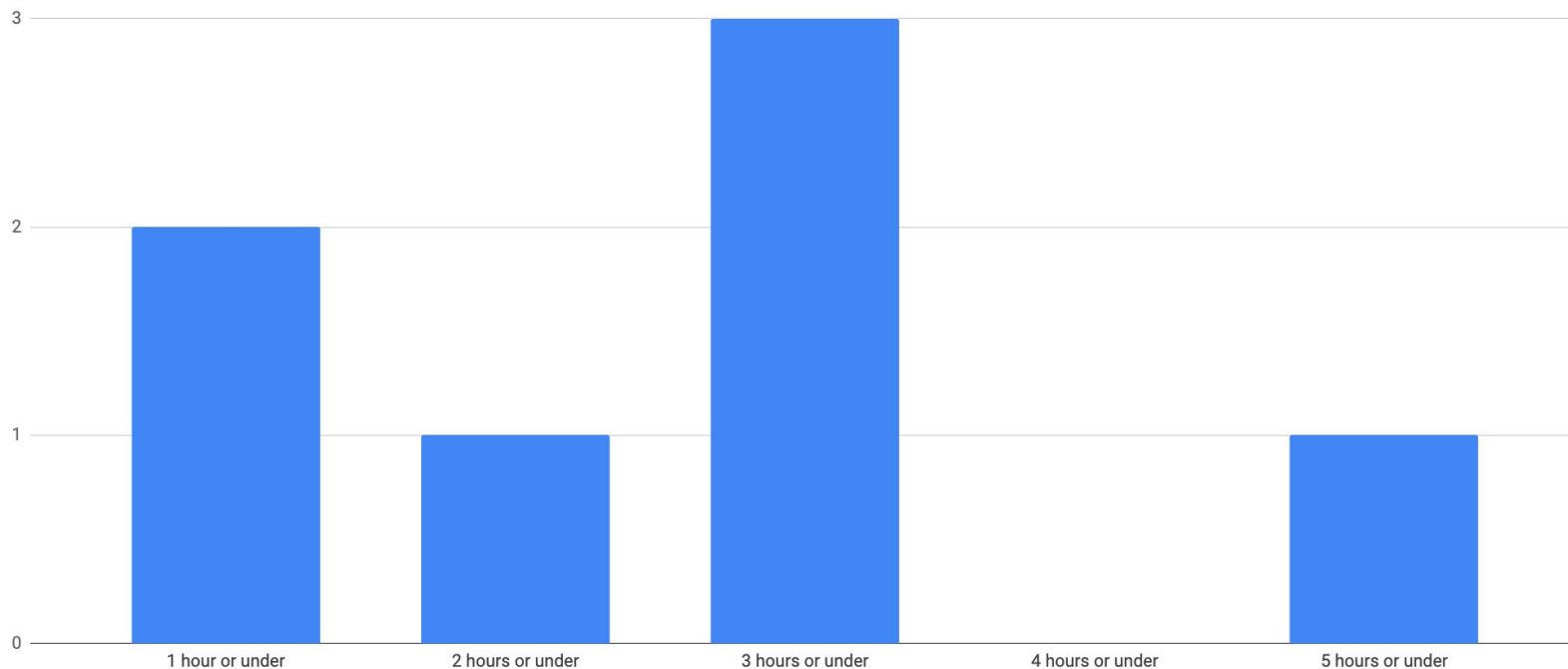
Homework

# Announcements

- April 22 Study Hall [food survey](#). Deadline April 15 EOD.
- Wireframes are due this weekend!
  - You'll now be working on your final project for the duration of the semester.
  - We'll ask for updates to your wireframe or final project in the next weekly assignment (5 points).

# How much time spent on J220

Week of 04-01: Number of students grouped by hours spent outside of lecture and office hours



## Homework Review

<script>

selecting element

Short answers

4. **True or False:** The script below will execute after the browser has parsed the full HTML document.

```
<script src="./scripts/main.js"></script>
```

**False.** You need to use the keyword **defer** if you'd like that to happen.



MOST  
MISSED Q

## Homework Review

<script>

selecting element

Short answers

6. How would you select the following element? (Refer to [Assignment #9 short answers](#) for full HTML.)

```
<h3><a href="https://journ220.github.io/">Aliquam  
egestas metus a rutrum interdum</a></h3>
```

```
document.querySelector('#hero h3');
```

```
document.querySelector('h3');
```

```
document.querySelector('main h3');
```

```
// all of the above
```



MOST  
MISSED Q

# Homework Review

<script>

selecting element

Short answers

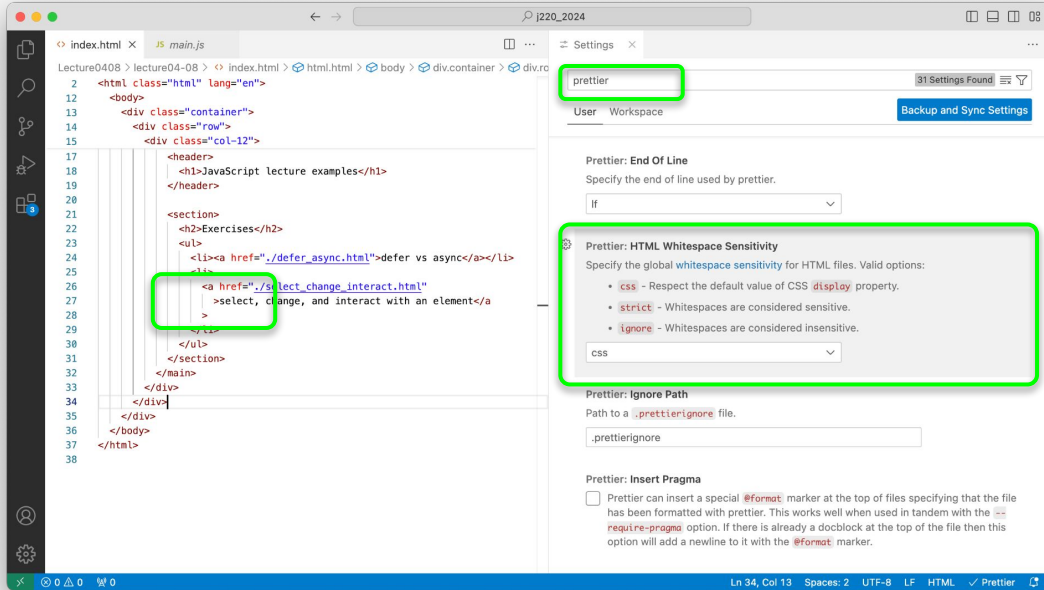
[Assignment #9 short answers](#)

Questions 5, 7, and 8



What questions  
do you have?

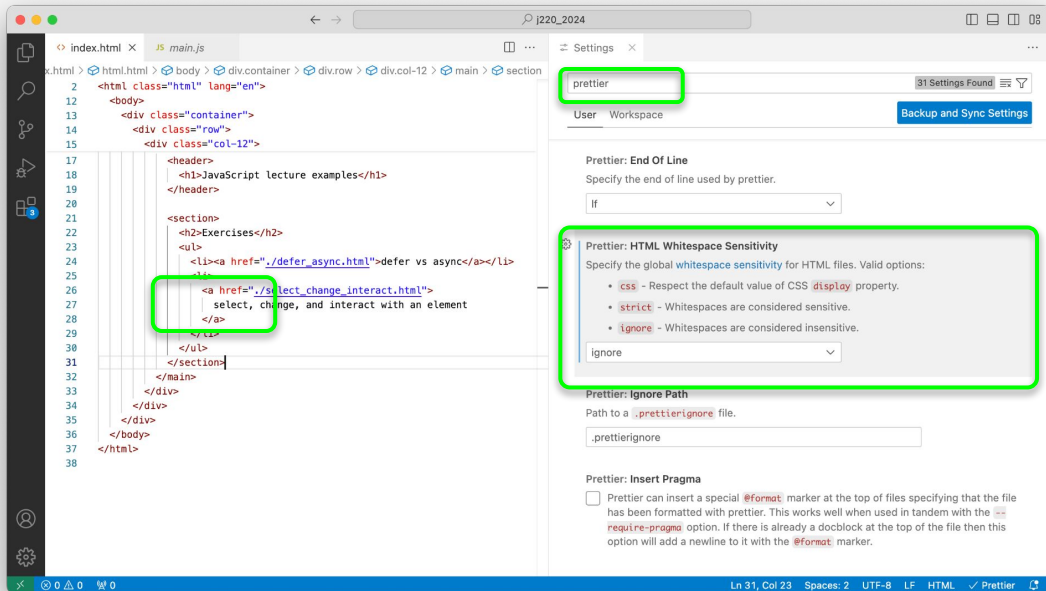
# More VS Code / Prettier formatting tips



I hated that Prettier was doing this to HTML, so I wanted to show you how to turn it off.

**Code > Settings > Settings** (or you can hit **command-comma**)

# More VS Code / Prettier formatting tips



Scroll down and find  
**Prettier: HTML  
Whitespace Sensitivity.**

Change the dropdown to  
**ignore.**

# JavaScript

Today's lecture is a totally new one that didn't exist in previous J220s.

Apologies in advance for typos/etc.!

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

**JavaScript** is a high-level programming language. It shares a lot of conventions with other high-level programming languages, like **Python** and **R**.

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

## What is a variable?

It's a name that stores a value.

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

```
// JavaScript  
var x = 7;
```

```
// JavaScript/ES6  
let x = 7;  
const x = 7;
```

```
# Python  
x = 7
```

```
# R  
x <- 7
```

Given the definition of variables, what's the variable here?

## What is the variable?

Nobody has responded yet.

Hang tight! Responses are coming in.



Start the presentation to see live content. For screen share software, share the entire screen. Get help at [pollev.com/app](https://pollev.com/app)



# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

```
// JavaScript  
var x = 7;
```

```
// JavaScript/ES6  
let x = 7;  
const x = 7;
```

```
# Python  
x = 7
```

```
# R  
x <- 7
```

**x** is the variable. **7** is the value of the variable.

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

```
// JavaScript  
var x = 7;
```

```
// JavaScript/ES6  
let x = 7;  
const x = 7;
```

```
# Python  
x = 7
```

```
# R  
x <- 7
```

Unlike Python and R, you declare variables in JavaScript using a *keyword* like **var**, **let**, or **const**.

(Javascript variables declared without keywords are “global variables,” a topic that is beyond the scope of this course.)

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

```
// JavaScript  
var x = 7;
```

```
// JavaScript/ES6  
let x = 7;  
const x = 7;
```

```
# Python  
x = 7
```

```
# R  
x <- 7
```

In JavaScript and Python, we use an **equal sign** to “set” a variable.

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

## Naming conventions

```
# JavaScript
```

```
let ucbAge = 156;
```

```
# Python
```

```
ucb_age = 156
```

```
# R
```

```
ucb.age = 156
```

```
ucb_age = 156
```

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

## Naming conventions

```
# JavaScript
```

```
let ucbAge = 156; # camelCase
```

```
# Python
```

```
ucb_age = 156 # snake_case
```

```
# R
```

```
ucb.age = 156 # historic, risk of confusion
```

```
ucb_age = 156 # I prefer this
```

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

Formatting is (mostly) arbitrary! `ucbAge` will work in Python and R, and `ucb_age` will work in JavaScript. But `ucb.age` will break in Python and JavaScript.

It's like AP style or Chicago style — you have your preferences, but you follow whatever the person in charge wants you to use.

You should follow the code style of the organization.

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

Variables that start with an **uppercase** letter denote a *class* — don't use them for now.

For now, start each variable with a **lowercase** letter.

(**All-capped variables** denote constants, or variables that don't change. But it's a little bit different from the JavaScript **const** declaration, which we'll talk about later.)

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

*# Reserved keywords: You can't use these words for variables*

```
abstract arguments await boolean break byte case  
catch char class const continue debugger default  
delete do double else enum eval export extends  
false final finally float for function goto if  
implements import in instanceof int interface let  
long native new null package private protected  
public return short static super switch  
synchronized this throw throws transient true try  
typeof var void volatile while with yield
```



# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

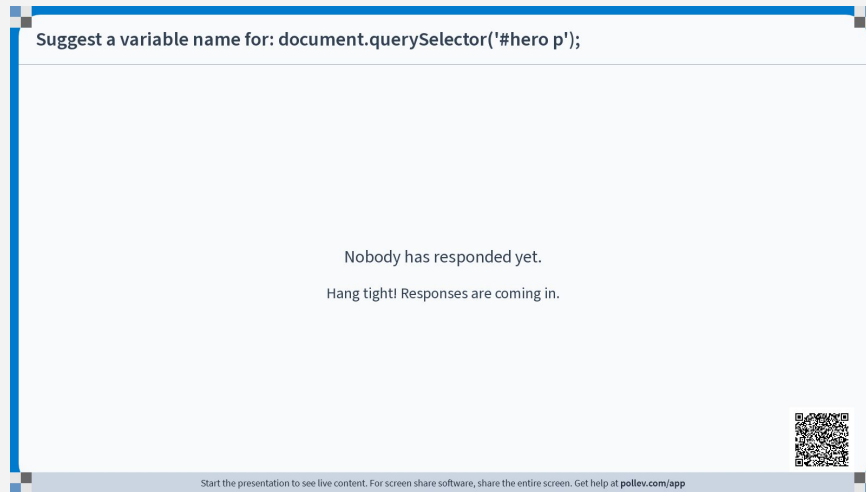
Syntax

*# Suggest a variable name for the following selector*

```
document.querySelector('#hero p');
```

# Suggest a variable name for:


```
document.querySelector('#hero p');
```



Suggest a variable name for: `document.querySelector('#hero p');`

Nobody has responded yet.  
Hang tight! Responses are coming in.

Start the presentation to see live content. For screen share software, share the entire screen. Get help at [pollev.com/app](https://pollev.com/app)



# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

*# Suggest a variable name for the following selector*

```
let heroParagraph = document.querySelector('#hero p');
```

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

*# Suggest a variable name for the following selector*

```
let heroParagraph = document.querySelector('#hero p');
```

*# What's useful about setting a variable name for a selector? You don't need to repeat the entire selector every time you want to make a change.*

# Variables

What is a variable?

Naming conventions

Reserved keywords

Question

Syntax

```
document.querySelector('#hero p').innerText = 'I am the  
hero paragraph.';  
document.querySelector('#hero p').style.color = 'red';
```

*# OR*

```
let heroParagraph = document.querySelector('#hero p');  
heroParagraph.innerText = 'I am the hero paragraph.';  
heroParagraph.style.color = 'red';
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;
```

```
x = 100;
```

```
// What is x?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;
```

```
x = 100;
```

```
// What is x?
```

```
// 100
```



# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;
```

```
x = 100;
```

```
// What is x?
```

```
// 100
```

```
x = 68;
```

```
x = 60;
```

```
x = 27;
```

```
x = 94;
```

```
x = 5;
```

```
// What is x?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;
```

```
x = 100;
```

```
// What is x?
```

```
// 100
```

```
x = 68;
```

```
x = 60;
```

```
x = 27;
```

```
x = 94;
```

```
x = 5;
```

```
// What is x?
```

```
// 5
```

What questions  
do you have?

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
let y = 10;  
let z = x + y;
```

```
// What is z?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
let y = 10;  
let z = x + y;
```

```
// What is z?  
// 17
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
let y = 10;  
let z = x + y;
```

```
// What is z?  
// 17
```

```
x = x + 3;  
// What is x?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
let y = 10;  
let z = x + y;
```

```
// What is z?  
// 17
```

```
x = x + 3;  
// What is x?  
// 10
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
x = 100;
```

```
// What is x?  
// 100
```

```
x = 68;
```

```
x = 60;
```

```
x = 27;
```

```
x = 94;
```

```
x = 5;
```

```
// What is x?  
// 5
```

Let's go back to  
the older slide for  
**Assignment  
Operators**



# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let x = 7;  
let y = 10;  
let z = x + y;
```

```
// What is z?  
// 17
```

```
x = x + 3;  
// What is x?  
// 10
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?
```

```
# -4
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?  
# -4
```

```
a = a * 3;  
# What is a?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?  
# -4
```

```
a = a * 3;  
# What is a?  
# 24
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?
```

```
# -4
```

```
a = a * 3;
```

```
# What is a?
```

```
# 24
```

```
# What is a / b?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 8;  
let b = 12;
```

```
# What is a - b?
```

```
# -4
```

```
a = a * 3;
```

```
# What is a?
```

```
# 24
```

```
# What is a / b?
```

```
# 2
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let s = 5**3; // this is an exponent (5 * 5 * 5)  
// What's s?
```



# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let s = 5**3; // this is an exponent (5 * 5 * 5)
// What's s?
// 125
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let s = 5**3; // this is an exponent (5 * 5 * 5)
// What's s?
// 125
```

```
let m = 10 % 3; // this is a modulo operator
// What's m?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let s = 5**3; // this is an exponent (5 * 5 * 5)
// What's s?
// 125
```

```
let m = 10 % 3; // this is a modulo operator
// What's m?
// 1
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let s = 5**3; // this is an exponent (5 * 5 * 5)
// What's s?
// 125
```

```
let m = 10 % 3; // this is a modulo operator
// What's m?
// 1
```

```
// A modulo is the remainder after dividing
// two numbers.
// 10 divided by 3 is 3 remainder 1.
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 7;  
a = a + 7;
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 7;  
a = a + 7;
```

```
// this is a compound operator  
// it means "operate on that same variable"  
a += 7; // equivalent to `a = a + 7`  
// What's a?
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 7;  
a = a + 7;
```

```
// this is a compound operator  
// it means "operate on that same variable"  
a += 7; // equivalent to `a = a + 7`  
// What's a?  
// 21
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let a = 7;  
a = a + 7;
```

```
// this is a compound operator  
// it means "operate on that same variable"  
a += 7; // equivalent to `a = a + 7`  
// What's a?  
// 21
```

```
a -= 7; // subtract 7 from a  
a *= 7; // multiply 7 and a  
a /= 7; // divide a by 7
```



# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

A **relational operator** tells us if 2 values are **True** or **False**.

You can also call this a **comparison operator**.

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
1 == 1  
// true
```

```
1 == 2  
// false
```

```
1 != 2  
// true
```

```
1 < 2  
// true
```

```
1 <= 2  
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
1 == 1  
// true
```

```
1 == 2  
// false
```

```
1 != 2  
// true
```

```
1 < 2  
// true
```

```
1 <= 2  
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
1 == 1 // 1 equals 1  
// true
```

```
1 == 2 // 1 equals 2  
// false
```

```
1 != 2 // 1 is not equal to 2  
// true
```

```
1 < 2 // 1 is less than 2  
// true
```

```
1 <= 2 // 1 is less than or equal to 2  
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

**Logical operators** are also a kind of comparison or relational operator.

You use it to compare **true** or **false** values.

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;
```

```
let t = 31;
```

```
n == t
```

```
//
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;
```

```
let t = 31;
```

```
n == t
```

```
// false
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;
```

```
let t = 31;
```

```
n == t
```

```
// false
```

```
n >= t
```

```
//
```



# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;
```

```
let t = 31;
```

```
n == t
```

```
// false
```

```
n >= t
```

```
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;  
let t = 31;
```

```
n == t  
// false
```

```
n >= t  
// true
```

```
(n == t) && (n >= t)  
//
```

```
(n == t) || (n >= t)  
//
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;  
let t = 31;
```

```
n == t  
// false
```

```
n >= t  
// true
```

```
(n == t) && (n >= t) // Logical "AND" operator  
// false
```

```
(n == t) || (n >= t) // Logical "OR" operator  
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
let n = 92;  
let t = 31;
```

```
n == t  
// false
```

```
n >= t  
// true
```

```
(n == t) && (n >= t)  
// false
```

```
(n == t) || (n >= t)  
// true
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
true && true
```

```
//
```

```
false && false
```

```
//
```

```
true && false
```

```
//
```

```
true || false
```

```
//
```

```
false || false
```

```
//
```

# Operators

Assignment operator

Arithmetic operators

Compound operators

Relational operators

Logical operators

```
true && true  
// true
```

```
false && false  
// false
```

```
true && false  
// false
```

```
true || false  
// true
```

```
false || false  
// false
```

What questions  
do you have?

# Break

Meet back in 15 minutes. **7:51 pm**



start Zoom screenshare +  
recording

# JavaScript

## var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

When should you use **var**, **let** or **const**?

The short answer is: we don't use **var** as much these days in modern JavaScript.

**var** used to be the only way to declare variables in JavaScript. But **var** had a lot of problems with “**scope**,” which is how we talk about *where* in the code we can access a variable. (I mentioned “global” variables earlier; that is part of “scope.”) Don't worry about scope for now, but it will be important in just a few more slides.

# JavaScript

## var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

In modern JavaScript, you declare variables using **let** and **const**.

Use **let** when you will change the variable by setting it to something else at some point (using the equal sign).

Use **const** when you will not change the variable after declaring it.

# JavaScript

**var** vs. **let** vs. **const**

strings

string indexing

arrays

functions

loops

conditionals

```
let notificationText = 'Alert!';  
notificationText = 'Hi!'; // OK to change value of  
variable declared with `let`
```

```
const warningText = 'Warning!';  
warningText = 'Caution!'; // Error. You can't  
redefine a `const`
```

```
// using all caps because pi is always 3.14169  
const PI = 3.14169;
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

You've already learned two **types** of JavaScript variables: **numbers** and **booleans** (true/false). Now let's learn a third one called **string**.

A **string** is, basically, text, wrapped with quotes.

```
let firstName = 'Soo';  
let lastName = 'Oh';  
  
// It doesn't matter whether you use  
// single or double quotes, but be consistent  
firstName = "Soo";  
lastName = "Oh";
```

# JavaScript

**var** vs. **let** vs. **const**

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
// 8
```



# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
// 8
```

```
university[0]
```

```
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
// 8
```

```
university[0]
```

```
// 'B'
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
// 8
```

```
university[0]
```

```
// 'B'
```

```
university[8]
```

```
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length
```

```
// 8
```

```
university[0]
```

```
// 'B'
```

```
university[8]
```

```
// undefined
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length  
// 8
```

```
university[0]  
// 'B'
```

```
university[8]  
// undefined
```

```
// How would I get 'k' from the string?
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
let university = 'Berkeley';
```

```
university.length  
// 8
```

```
university[0]  
// 'B'
```

```
university[8]  
// undefined
```

```
// How would I get 'k' from the string?  
university[3]
```

# JavaScript

**var** vs. **let** vs. **const**

strings

string indexing

arrays

functions

loops

conditionals

An array is an ordered collection of elements.

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```



# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length
```

```
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
// 'Becca'
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
// 'Becca'
```

```
j220[3]  
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
// 'Becca'
```

```
j220[3]  
// Isabella'
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
// 'Becca'
```

```
j220[3]  
// Isabella'
```

```
j220[j220.length - 1]  
//
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
const j220 = ['Becca', 'Agnee', 'Chelsea',  
'Isabella', 'Raymond', 'Ruchi', 'Iris', 'Hailey',  
'Edison', 'Lucy'];
```

```
j220.length  
// 10
```

```
j220[0]  
// 'Becca'
```

```
j220[3]  
// Isabella'
```

```
j220[j220.length - 1]  
// 'Lucy'
```



# JavaScript

`var` vs. `let` vs. `const`

strings

string indexing

arrays

functions

loops

conditionals

A **function** is a reusable block of code that performs an action. It only runs when it's called.

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

```
// call the function  
average(2, 6);  
// 4
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

name of function

```
// call the function  
average(2, 6);  
// 4
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

assignment operator

```
// call the function  
average(2, 6);  
// 4
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

```
// call the function  
average(2, 6);  
// 4
```

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
    return (num1 + num2)/2;  
};
```

```
// call the function  
average(2, 6);  
// 4
```

If you use `function` you always need these parentheses. You don't always need to fill the parentheses with parameters.

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

```
// call the function  
average(2, 6);  
// 4
```

You'll need these curly brackets, too. Don't forget to end the function with a semicolon.

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

Most functions will **return** something, but not always.

```
// call the function  
average(2, 6);  
// 4
```



# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

*// There are multiple ways of writing functions in JavaScript. We're only going to learn one today.*

```
const average = function(num1, num2) {  
  return (num1 + num2)/2;  
};
```

parameters

*// call the function*

```
average(2, 6);  
// 4
```

arguments

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
document.querySelectorAll("h2").forEach(function  
(element, index) {  
  // Looping  
  if (index % 2) {  
    element.style.color = "blue";  
  } else {  
    element.style.color = "green";  
  }  
});
```

Go to **forEach** folder

# JavaScript

var vs. let vs. const

strings

string indexing

arrays

functions

loops

conditionals

```
document.querySelectorAll("h2").forEach(function  
(element, index) {  
  // Looping  
  if (index % 2) {  
    element.style.color = "blue";  
  } else {  
    element.style.color = "green";  
  }  
});
```

Go to **forEach** folder

What questions  
do you have?

# Let's look at some of the other materials

- docReady/
- 8Ball/

Let's talk about scrollytellers

# Homework

<https://journ220.github.io>

**Please help  
clean up:** close  
windows,  
return tables,  
etc.