J220
Coding for Journalists

LECTURERS
Yoli Martinez
Soo Oh

PROMPTS

- Log into iClicker for attendance.
- Download lecture examples: https://journ220.github.io/assets/static/lecture0418examples.zip

start Zoom recording + captions

Agenda

Announcements

In-class critiques

BREAK

JavaScript

In-class exercises

Announcements

No homework or extra credit this week. Work on final project.

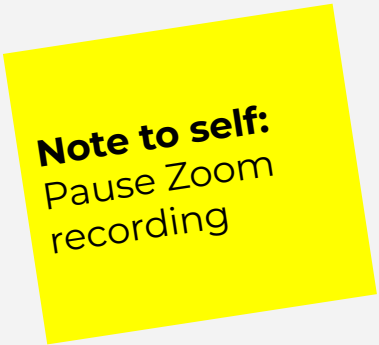
Check grades

Next lecture (April 25)

- **DUE WEDNESDAY**
  Tell us what you want from **Bańh Mi & Rolls Factory**:
  https://forms.gle/o1jNCTrWNFRiDTZ88
- Presentations
- Study Hall

In-class critiques

BREAK



Meet back in 15 minutes

start Zoom recording

JavaScript

Variables

`let`

const

var

Naming conventions

What is a variable?

It's a name that stores a value.

To declare a variable in JavaScript, we use the keywords **`let`**, **`const`**, or **`var`**.

We'll start with **`let`**.

Variables

`let`

const

var

Naming conventions

Declaring a variable with `let`

You can keep re-assigning different values to a variable with **`let`**.

Variables

let

const

var

Naming conventions

```
let x = 100;
```

x is the name of the variable. The name is arbitrary, but there are conventions.

Variables

let

const

var

Naming conventions

```
let x = 100;
```

Use a semicolon to separate lines of code.

Variables

`let`

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;
```

You can change the value of variables declared with `let`.

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
```

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
// 1300
```

What questions do you have?

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
// 1300

x = 68;
x = 60;
x = 27;
x = 'chicken';
```

You don't have to stick with just numbers.

When you use letters or other non-numeric characters, you need to put quotes around it. (Otherwise, it looks like a variable!)

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
// 1300

x = 68;
x = 60;
x = 27;
x = 'chicken';
```

You don't have to stick with just numbers.

When you use letters or other non-numeric characters, you need to put quotes around it. (Otherwise, it looks like a variable!)

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
// 1300

x = 68;
x = 60;
x = 27;
x = 'chicken';

x = true;
x = false;
```

You can also set variables with true or false values.

Variables

let

const

var

Naming conventions

```
let x = 100;
x = 7;
x = 1300;

// What is x?
// 1300
```

Demo in Chrome

```
x = 60;
x = 27;
x = 'chicken';

x = true;
x = false;
```

What questions do you have?

Variables

`let`

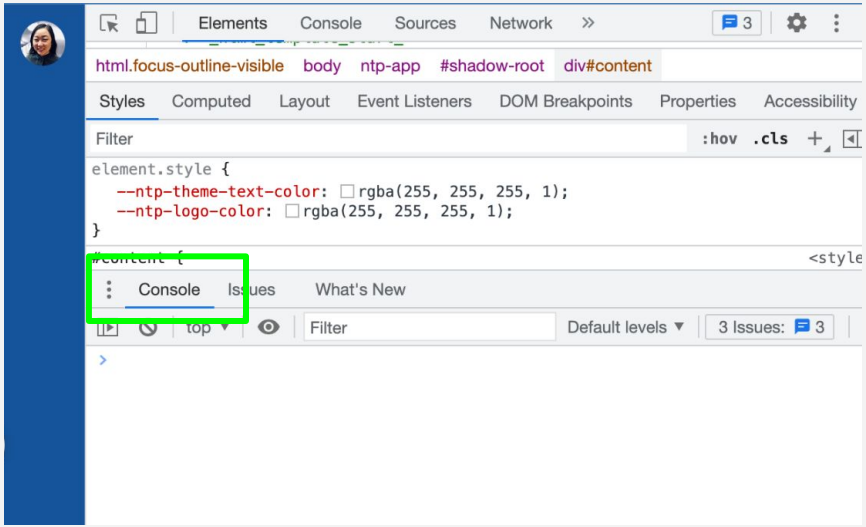
`const`

`var`

Naming conventions

Open up your browser console

- In Chrome, right click on an empty page, and select “Inspect Element”
- Then tab to “Console”.



Variables

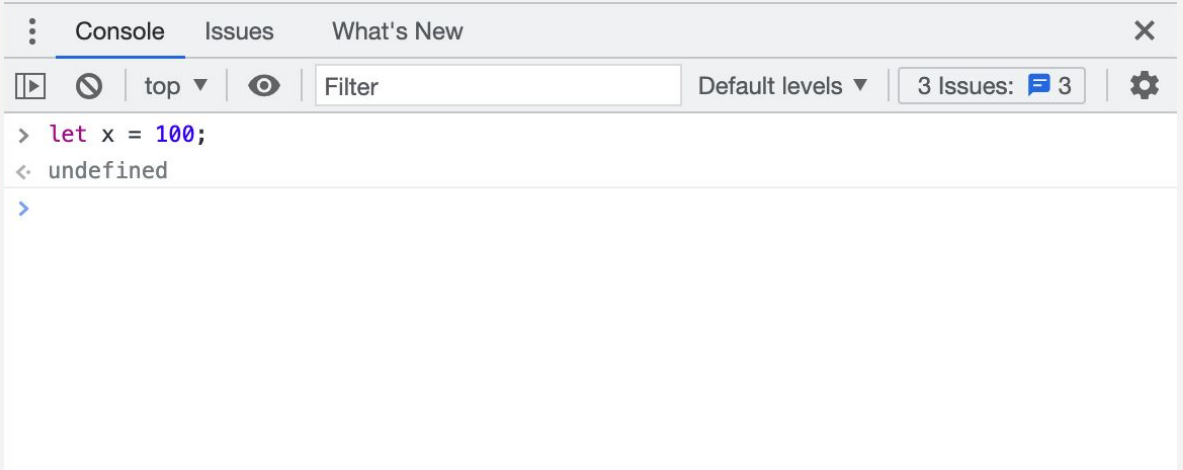
let

const

var

Naming conventions

Type in the following text:



Demo in Chrome

What questions do you have?

Variables

`let`

`const`

`var`

Naming conventions

Declaring a variable with `const`

Once you use the **equal sign** — an "assignment operator" — with **`const`**, you cannot use it again.

Variables

let

const

var

Naming conventions

Declaring a variable with const

Once you use the **equal sign** — an "assignment operator" — with **const**, you cannot use it again.

```
> const x = 100;
< undefined
> x = 10;
✕ ▶ Uncaught TypeError: Assignment to constant variable.        VM276:1
      at <anonymous>:1:3
>
```

Demo in Chrome

What questions do you have?

Variables

`let`

`const`

`var`

Naming conventions

Declaring a variable with `var`

For now, try not to use it. (There's more to this, but we're not going to learn it today!)

Variables

let

const

var

Naming conventions

```
// You can use letters, numbers, and
// underscores for variable names.
let ucb_age = 154; // snake_case
let ucbAge = 154;  // camelCase

// You cannot use hyphens.

// Don't use periods like you would do
// in R:
let ucb.age = 154; // This is an error!
```

Variables

let

const

var

Naming conventions

```
// You can use letters, numbers, and
// underscores for variable names.
let ucb_age = 154; // snake_case
let ucbAge = 154;  // camelCase

// You cannot use hyphens.

// Don't use periods like you would do
// in R:
let ucb.age = 154; // This is an error!
```

camelCase is most commonly used in JavaScript

Demo in Chrome

Variables

`let`

`const`

`var`

Naming conventions

Formatting is arbitrary! **`ucbAge`** will work in Python and R, and **`ucb_age`** will work in JavaScript. (EXCEPTION: **`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

`let`

`const`

`var`

Naming conventions

All-capped variables denote constants, or variables that don't change (it's a style across programming languages).

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.

What questions do you have?

Data structures

Arrays

Objects

Sometimes we want to store more than one value in a variable. What can we use to store more than one value?

Data structures

Arrays

Objects

Array literal

```
let lecturer1 = 'Soo';
let lecturer2 = 'Yoli';

const lecturers = [lecturer1, lecturer2];

lecturers[0]
// 'Soo'

lecturers[1]
// 'Yoli'

lecturers.length
// 2
```

What questions do you have?

Data structures

Arrays

Objects

Object literal

```
const lecturer1 = {
  'name': 'Soo',
  'course': 'J220'
};

lecturer1.name
// 'Soo'

lecturer1.course
// 'J220'
```

This might look familiar to you if you've ever seen JSON data formats.

Data structures

Arrays

Objects

Object literal

```
const lecturer1 = {
  'name': 'Soo',
  'course': 'J220'
};

lecturer1.name
// 'Soo'

lecturer1.course
// 'J220'
```

You should know about objects, but we won't be using them in lecture today.

`Control flow`

`if/else`

`for loop`

Types of **control flow**:

- sequential (what we've been doing so far, we execute one line after another in order)
- selection (**`if`**... **`else if`**... **`else`**)
- repetition (**`while`** and **`for`** loops)

Today we'll learn **if/else** and **for loops**.

Control flow

`if/else`

`for loop`

Structure of `if/else` statement

```
if (condition) {
  // do something
} else {
  // do something else
}
```

Let's look at the examples

What questions do you have?

Control flow

if/else

for loop

Close-up of `if-else/index.html`

```
const age = ageInput.value;

if (age >= 16) {
  // yes, 16+ year olds can drive
  driveInput.innerHTML = 'YES'
} else {
  // no, under-16 cannot drive
  driveInput.innerHTML = 'NO'
}
```

Pair with `if-else/index.html`

Control flow

`if/else`

`for loop`

Structure of `for` loop

An *iterable* is something you can iterate or loop over (like an array).

```
for (let i=0; i < iterable.length; i++) {

// do something to iterable[i]

}
```

Let's look at the examples

What questions do you have?

Note to students: Please help close the windows on your way out

`end Zoom recording`