IN4MATX 285:
Interactive Technology Studio

Programming: Variables, Loops, and Conditionals in JavaScript

Today’s goals

By the end of today, you should be able to…

- Conduct basic debugging activities in JavaScript
- Create code in JavaScript which follows programming syntax
- Create variables which hold values and use them to perform actions
- Execute code conditional on values, and loop over them

Disclaimer

- This class is introducing a <u>lot</u> of programming concepts, and quickly
- We don't expect you to pick up everything immediately
- Assignments will help practice these concepts
  - Live demos, online resources, and office hours aim to help supplement

Language Roles

HTML

Specify how content is rendered

CSS

Visually style content

JS

Dynamically manipulate content

Language Roles

Markup
language

Styling
language

Programming
language

Why JavaScript?

- Make pages dynamic
- Make pages personalized
- Make pages interact
  with other sources,
  like databases and Application
  Programming Interfaces (APIs)

So, how do we use JavaScript?

Loading JavaScript

```html
<html>
  <head>
    <script src="test.js"></script>
  </head>
</html>
```

- Separate file, just like CSS

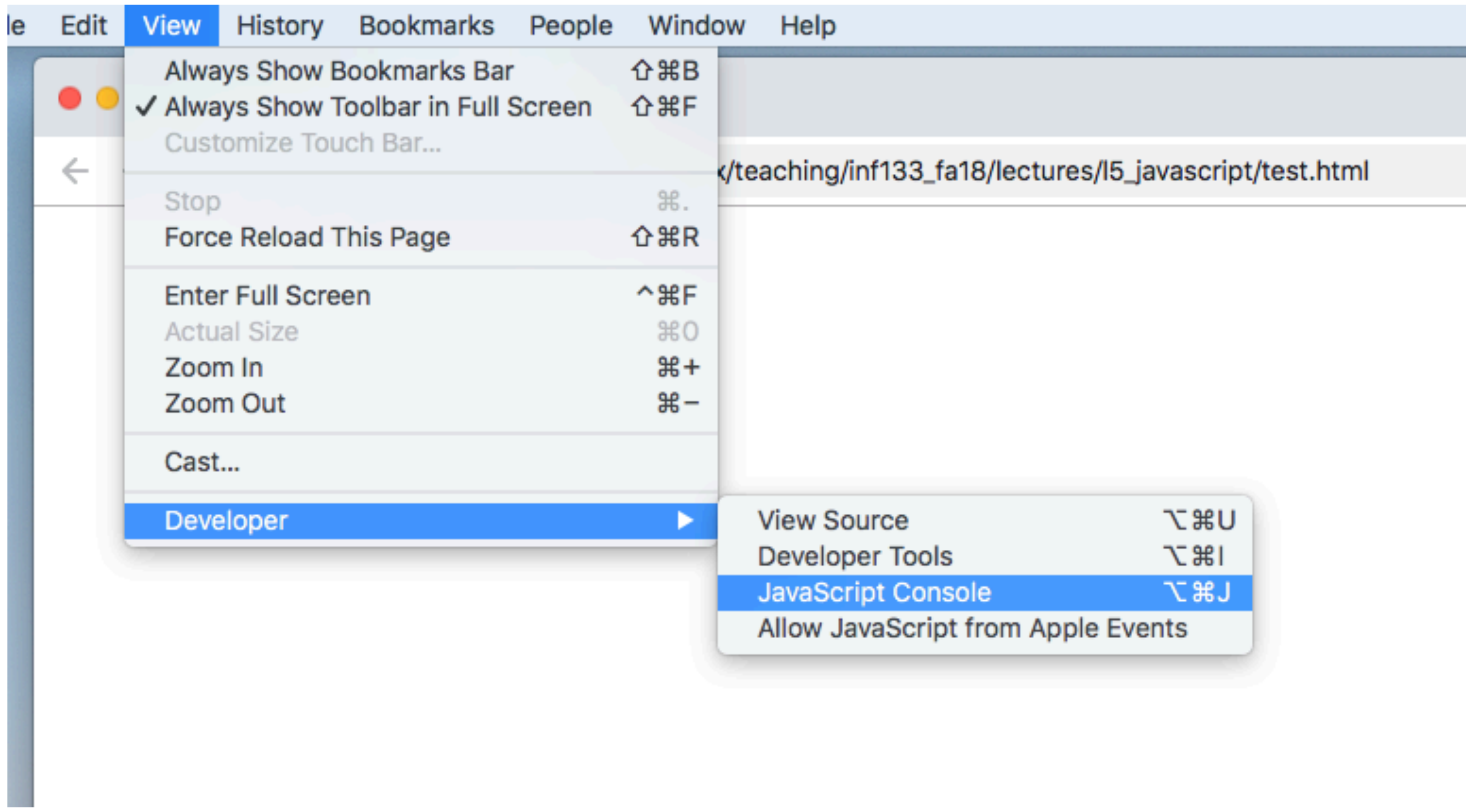
Hello, World!

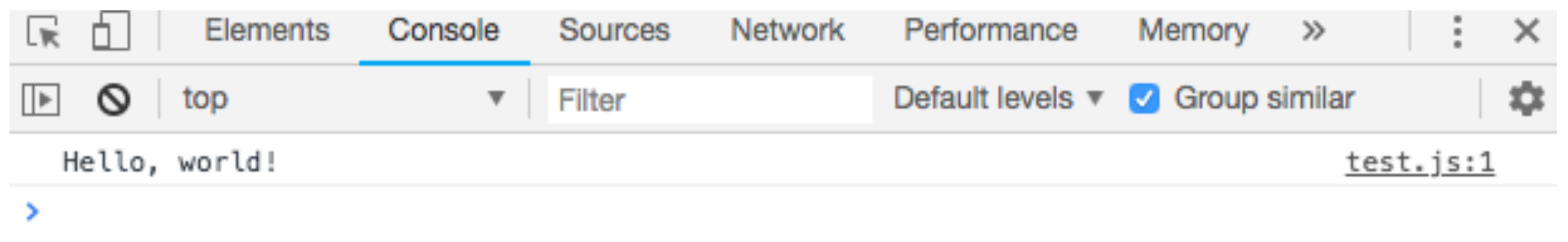
- Typically the first program you produce in a programming language
- Simply prints the phrase “Hello, World!”

Hello, World!

```
console.log("Hello, world!");
```

- Won't be visible in the browser
- Shows in the JavaScript Console

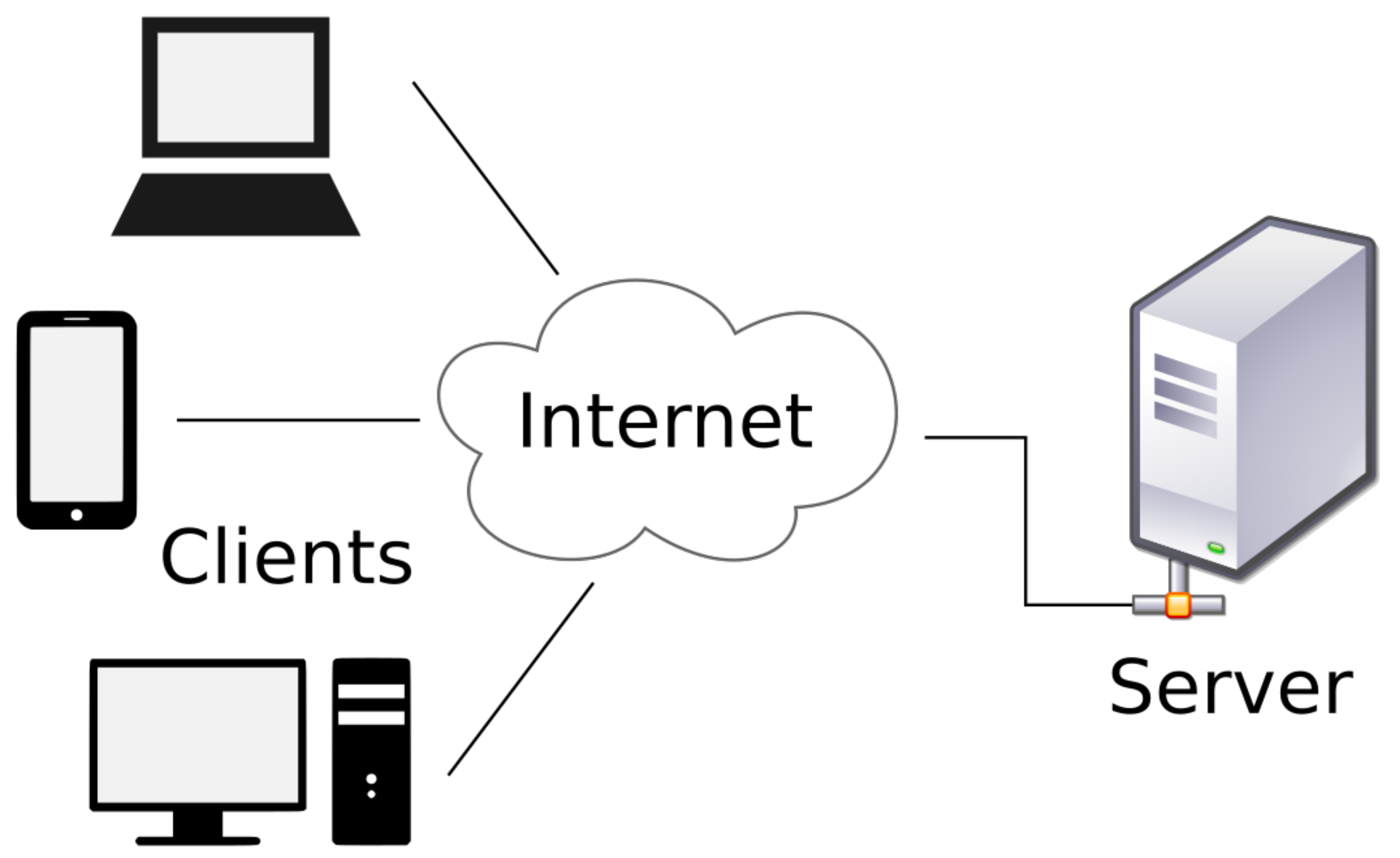

le Edit View History Bookmarks People Window Help
Always Show Bookmarks Bar ⇧⌘B
✓ Always Show Toolbar in Full Screen ⇧⌘F
Customize Touch Bar...
Stop ⌘.
Force Reload This Page ⇧⌘R
Enter Full Screen ^⌘F
Actual Size ⌘0
Zoom In ⌘+
Zoom Out ⌘−
Cast...
Developer
View Source ⌥⌘U
Developer Tools ⌥⌘I
JavaScript Console ⌥⌘J
Allow JavaScript from Apple Events
/teaching/inf133_fa18/lectures/l5_javascript/test.html



```
Hello, world!                                                   test.js:1
>
```

Hello, World!

- Why doesn’t the JavaScript show up on your page?
- One reason: the HTML file is coming from a **server**, while JavaScript is running on the **client**



Comments

```
//Prints "Hello, world!"
console.log("Hello, world!");
```

- `//` designates that a line should not be executed
- Useful for writing plain-text description of what your code does

Now, how do we make JavaScript useful?

Making JavaScript Useful

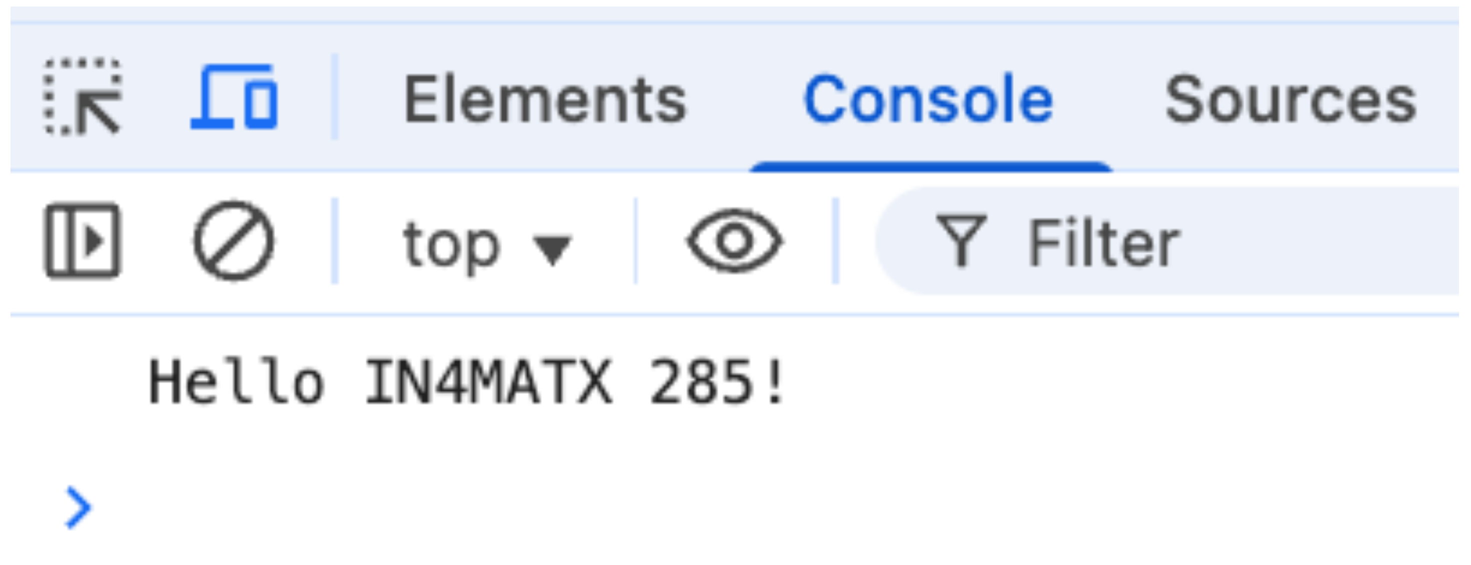
- Variables
- Conditionals
- Loops

Variables

Variables

- Store values
- Specified with `let`, can be named whatever you want (for the most part)
- Use equals (=) to *assign* a value to a variable
- Can be *referenced* later

```
let variable = 285;
console.log('Hello IN4MATX ' + variable + '!');
```



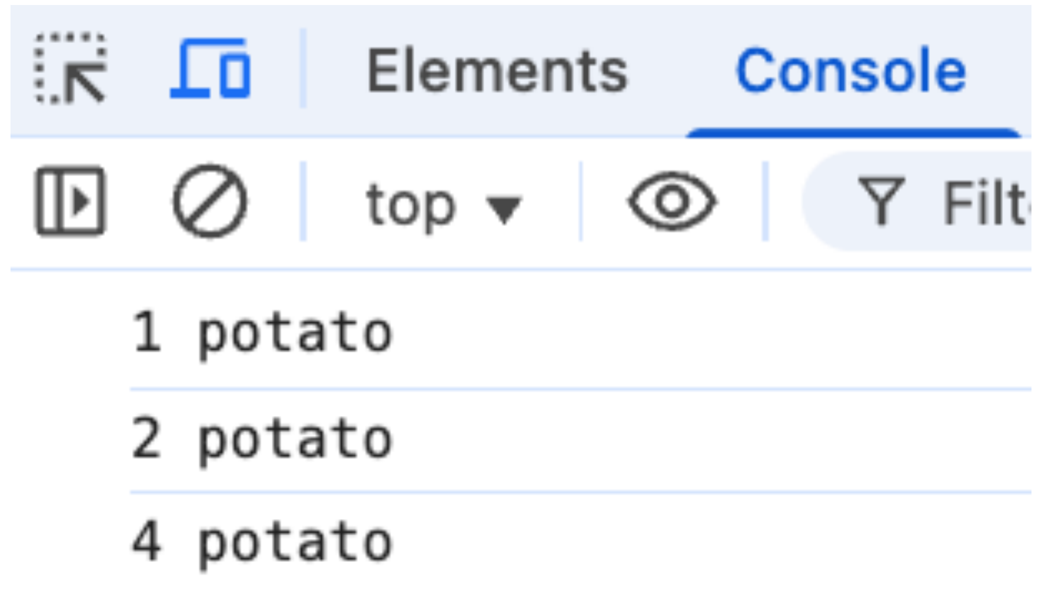
```
Hello IN4MATX 285!
>
```

Variables

- Can be updated

- Can use all sorts of math functions

```
let count = 1;
console.log(count + ' potato');
count = count + 1;
console.log(count + ' potato');
count = 2 * count;
console.log(count + ' potato');
```



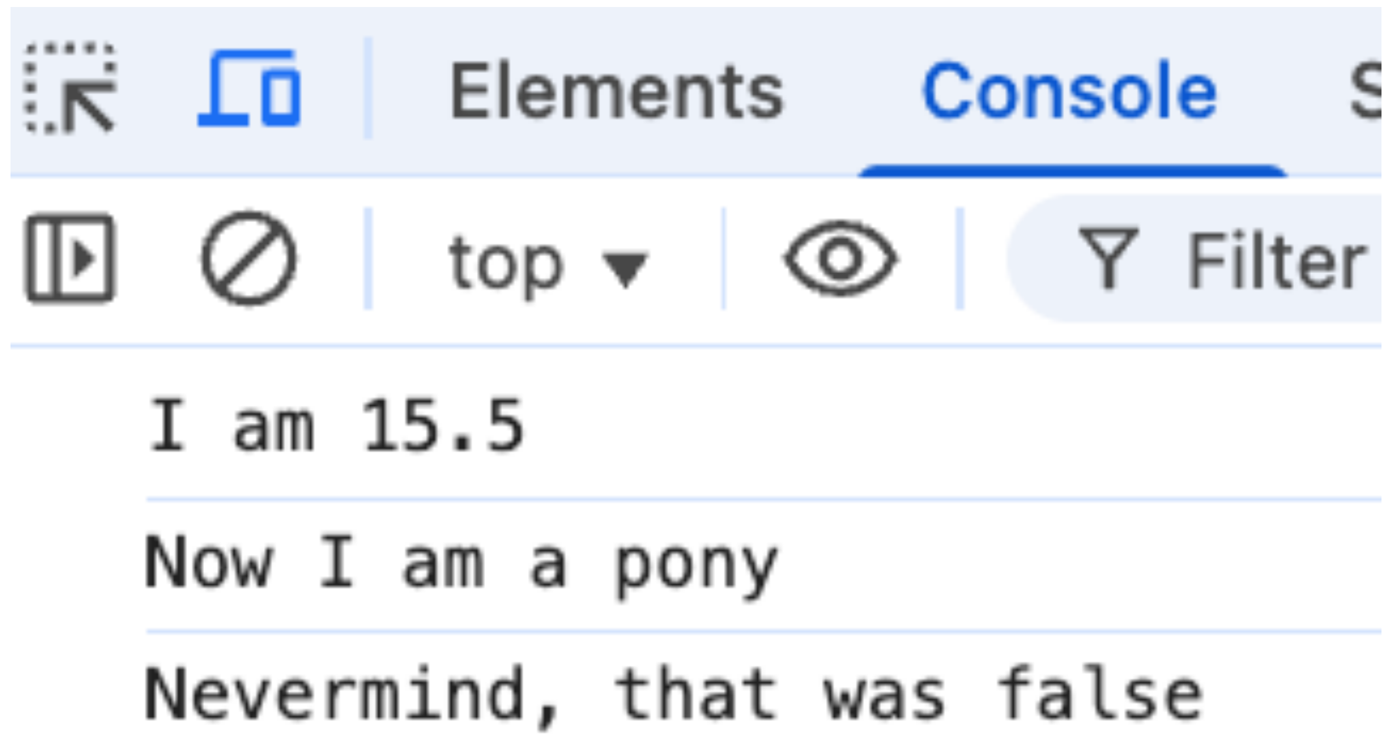
```
1 potato
2 potato
4 potato
```

Variables

- Variables have *types*
  - Numbers (whole numbers and decimals)
  - Strings (text)
  - Booleans (either true or false)
  - Many, many other types that we will ignore
- A variable can change between types in JavaScript

Variables

```
let value = 15.5;
console.log('I am ' + value);
value = 'a pony';
console.log('Now I am ' + value);
value = false;
console.log('Nevermind, that was ' + value);
```



```
I am 15.5
Now I am a pony
Nevermind, that was false
```

Conditionals

Conditionals

- Often, you want some code to run only *if* a particular condition is met
- Brackets `{ }` indicate blocks of code to run
- Can optionally designate other code to run *else* condition is not met

```
let isEighteen = true;
if(isEighteen) {
    console.log("I can vote!");
} else {
    console.log("Need to wait to vote");
}
```

Conditionals

- Can chain `if`s with `else if`
- Brackets `{ }` can run multiple lines of code

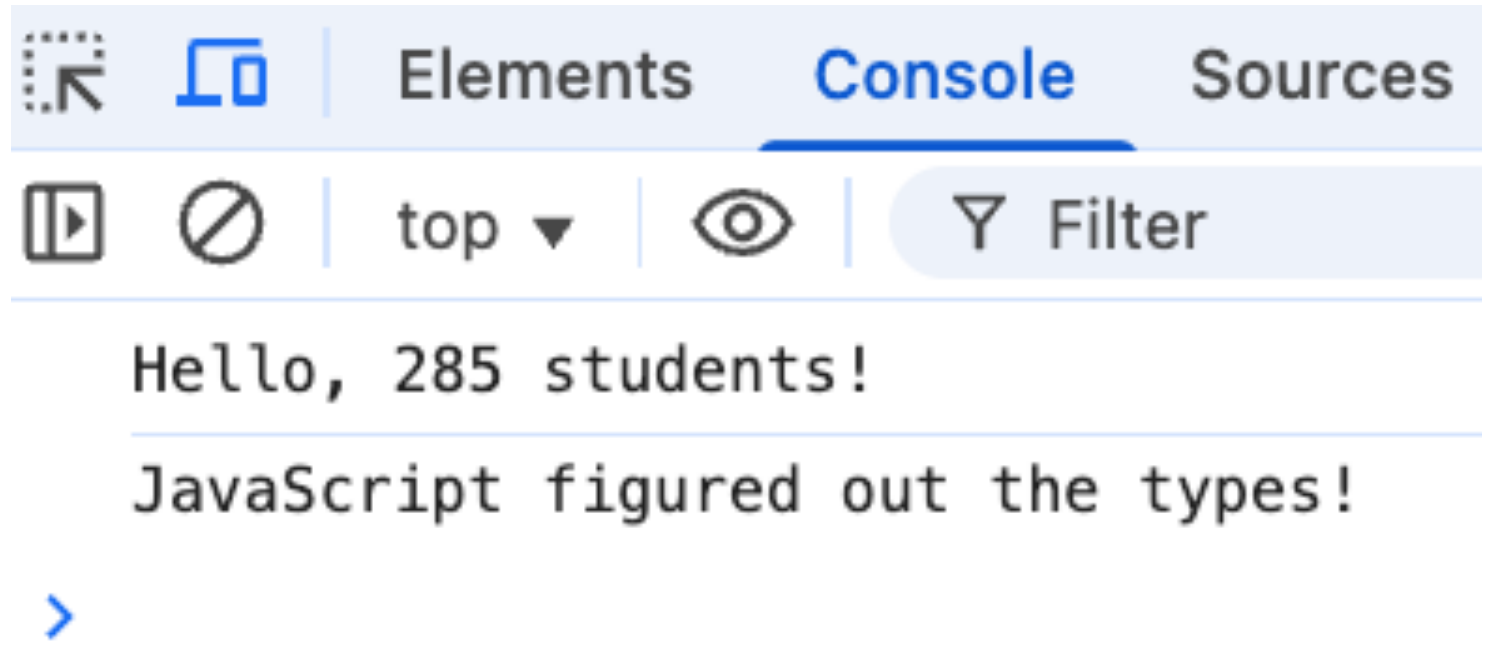
```
let myAge = 23;
if(myAge < 18) {
    console.log("Need to wait to vote");
} else if (myAge < 21) { //We know I am at least 18
    console.log("I can vote!");
    console.log("But I need to wait to drink");
} else { //We know I am at least 21
    console.log("I can vote and drink!");
}
```

Conditionals

- Equals can be checked with ==

- JavaScript will try to convert between types

```
let courseNumber = 285;
if(courseNumber == 285) {
    console.log('Hello, 285 students!');
}
if(courseNumber == '285') {
    console.log('JavaScript figured out the types!');
}
```



```
Hello, 285 students!
JavaScript figured out the types!
```

Conditionals

- “Not” can be specified with an !
- Either checking “not equal” or inverting a boolean value

```
let courseNumber = 285;
if(courseNumber != 285) {
    console.log('Hello, other students!');
}
let isEighteen = false;
if(!isEighteen) {
    console.log('Need to wait to vote');
}
```

Conditionals

- Multiple conditionals can be combined with `&&` **(and)**, `||` **(or)**

```
let age = 19;
let hasFakeId = true;
if(age >= 18 && age < 21) {
    console.log('I can vote, but not drink');
}

if(age >= 21 || hasFakeId) {
    console.log('I can drink... :-)');
}
```

Loops

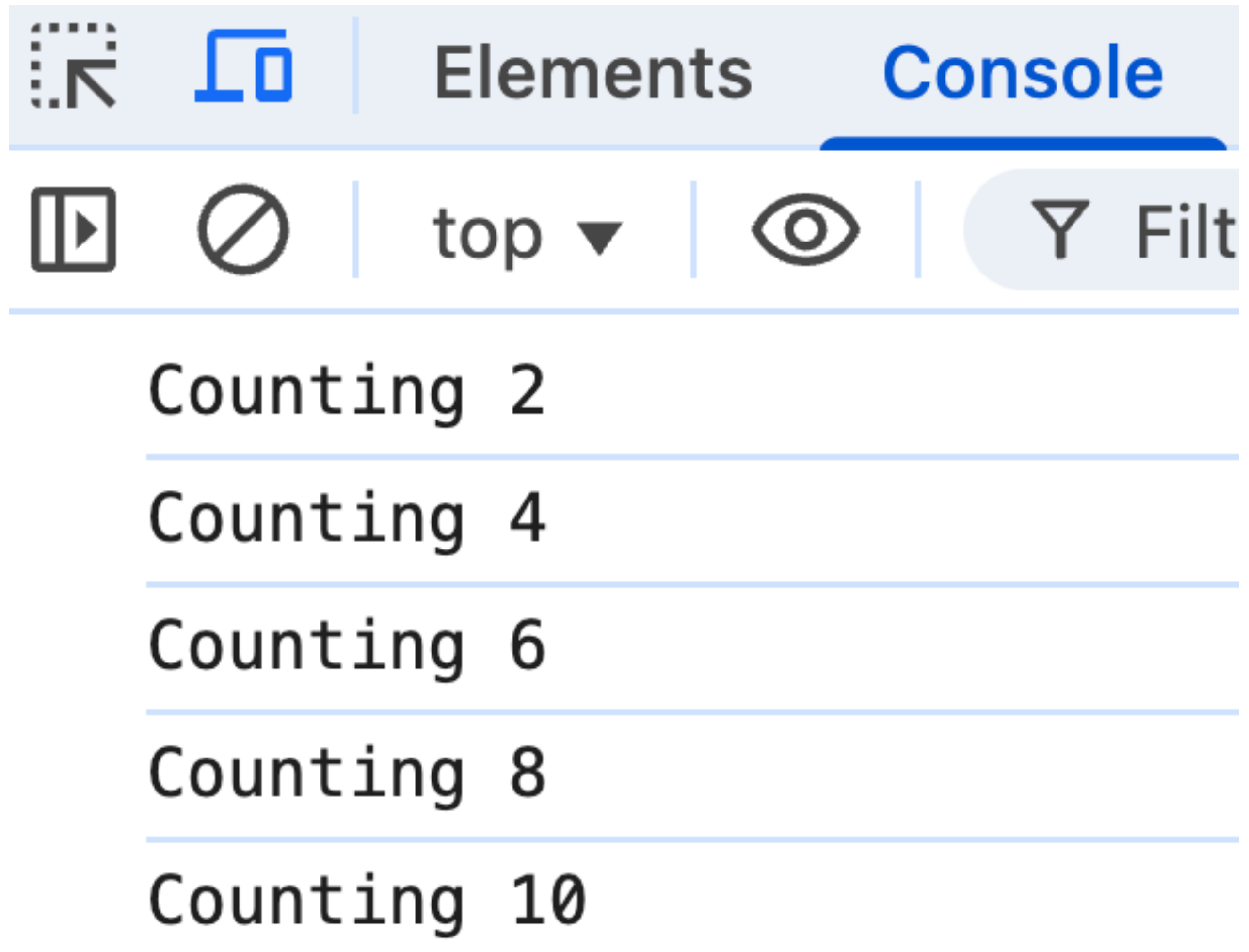
Loops

- Thus far, every line of code is executing sequentially
- Loops allow you to *repeat* lines of code, while changing variables every time

Loops

Initialize End condition Advancement

```
//Let's count to 10 by twos!
for(let i = 1; i <= 5; i = i + 1) {
    let iTimesTwo = i * 2;
    console.log("Counting " + iTimesTwo);
}
```



```
Counting 2
Counting 4
Counting 6
Counting 8
Counting 10
```

Loops

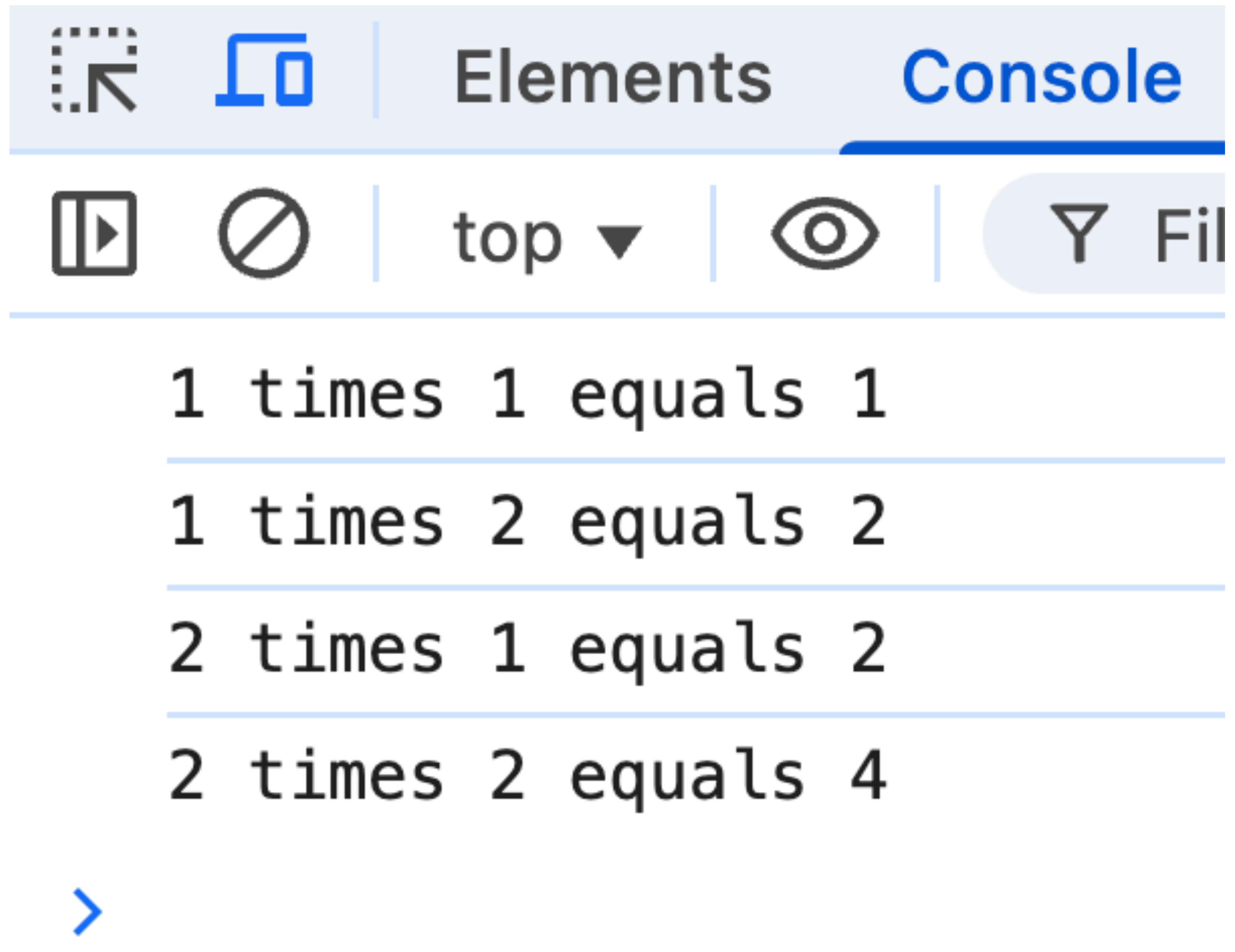
```
//Let's count to 10 by twos!
for(let i = 1; i <= 5; i = i + 1) {
    let iTimesTwo = i * 2;
    console.log("Counting " + iTimesTwo);
}

//Does the same thing
for(let i = 2; i <= 10; i = i + 2) {
    console.log("Counting " + i);
}
```

Loops

- Loops can be nested

```
//Two by two times table
for(let i = 1; i <= 2; i = i + 1) {
    for(let j = 1; j <= 2; j = j + 1) {
        console.log(i + ' times ' + j + ' equals ' + i*j);
    }
}
```



```
1 times 1 equals 1
1 times 2 equals 2
2 times 1 equals 2
2 times 2 equals 4
```

Other concepts

- There are <u>tons</u> of basic programming concepts I am skipping over
  - While loops
  - Different modifiers (++, +=, %, /)
  - String manipulation
- A lot of these aim to make code more efficient or more concise
- Ask on Slack, or search online, if you need to understand how they work

Today’s goals

By the end of today, you should be able to…

- Conduct basic debugging activities in JavaScript
- Create code in JavaScript which follows programming syntax
- Create variables which hold values and use them to perform actions
- Execute code conditional on values, and loop over them

IN4MATX 285:
Interactive Technology Studio

Programming: Variables, Loops, and Conditionals in JavaScript

Additional slides

Let, Var, and Const

- Both `let` and `var` can be used to create a new variable
- `const` can be used to declare a variable that won't be changed
- There are some differences in how variables created with each are visible across your code
  - But these differences are pretty subtle and advanced, and largely won't matter for the code that we're creating in this class
- You might see some examples which use `var` to make new variables

null, undefined, and NaN

- `null`: a nonexistent object
  - Therefore it is an object, just unitialized

```
var nullObj = null;

console.log(typeof nullObj); //object
if(!nullObj) {
 console.log("It's falsy");
}
//but it's not equal to false
console.log(nullObj == false); //false
```

https://codeburst.io/understanding-null-undefined-and-nan-b603cb74b44c

null, undefined, and NaN

- `undefined`: an undefined primitive value
  - Therefore it’s a primitive value, like a number or a string

```
var undefinedObj;

console.log(undefinedObj); //undefined
console.log(typeof undefinedObj); //undefined
if(!undefinedObj) {
 console.log("It's falsy");
}
//but it's not equal to false
console.log(undefinedObj == false); //false
```

https://codeburst.io/understanding-null-undefined-and-nan-b603cb74b44c

null, undefined, and NaN

- `NaN`: Not a Number
  - Will be the result of any computation on an `undefined` value
  - Or any other impossible computation
  - But it's type is a number (despite the name)

```
console.log('12' - 5); // 7
console.log('word' - 5);// NaN
console.log(undefined * 3);// NaN
console.log(typeof NaN);// number
if(NaN) {
 console.log("It's not falsy!");
}
```

https://codeburst.io/understanding-null-undefined-and-nan-b603cb74b44c