INFO 2124

JavaScript

```
const module    = "Module 1";
const title     = "Introduction to Web Development & JavaScript";

document.addEventListener('DOMContentLoaded', ()=>{
        const message = `
        <h1>${module}</h1>
        <p>${title}</p>
        `;

        document.write(message);
});
```

Speaker notes

speaker notes

Module Objectives

A. List the primary rules for creating a JavaScript identifier.

B. Describe the use of JavaScript comments, including “commenting out” portions of JavaScript code.

C. Describe these three primitive data types that are available from JavaScript: numeric, string, and Boolean.

D. Describe the use of variable and constant declarations and assignment statements with numeric, string, and Boolean data.

E. Distinguish between the “let” and “const” keywords to declare constants and variables and using the “var” keyword to declare variables and constants.

F. Distinguish between using the + operator and a template string literal when working with strings.

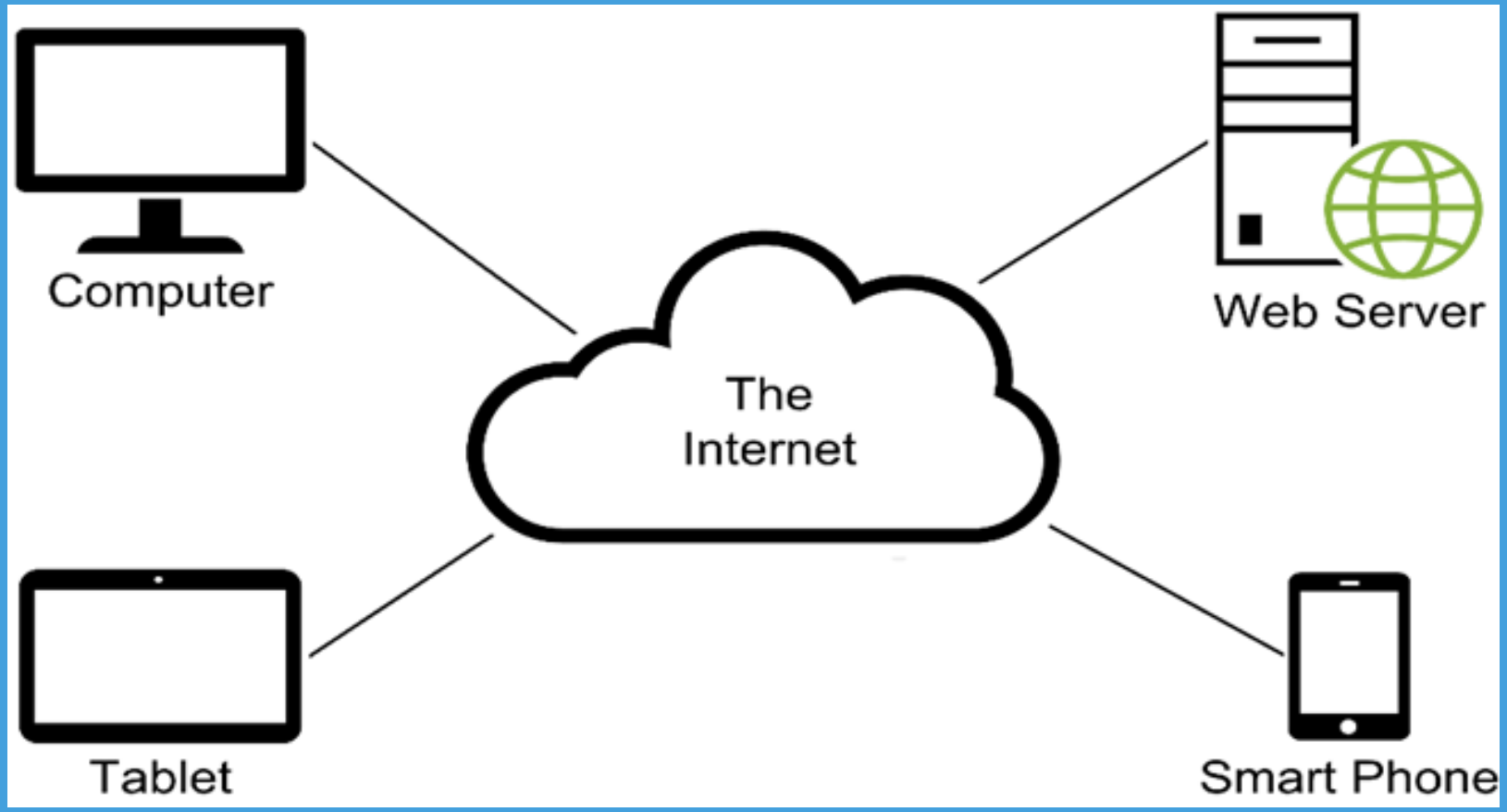
G. Code, test, and debug an application that stores user input in variables, modifies the input, and presents the results back to the user.

Web Development

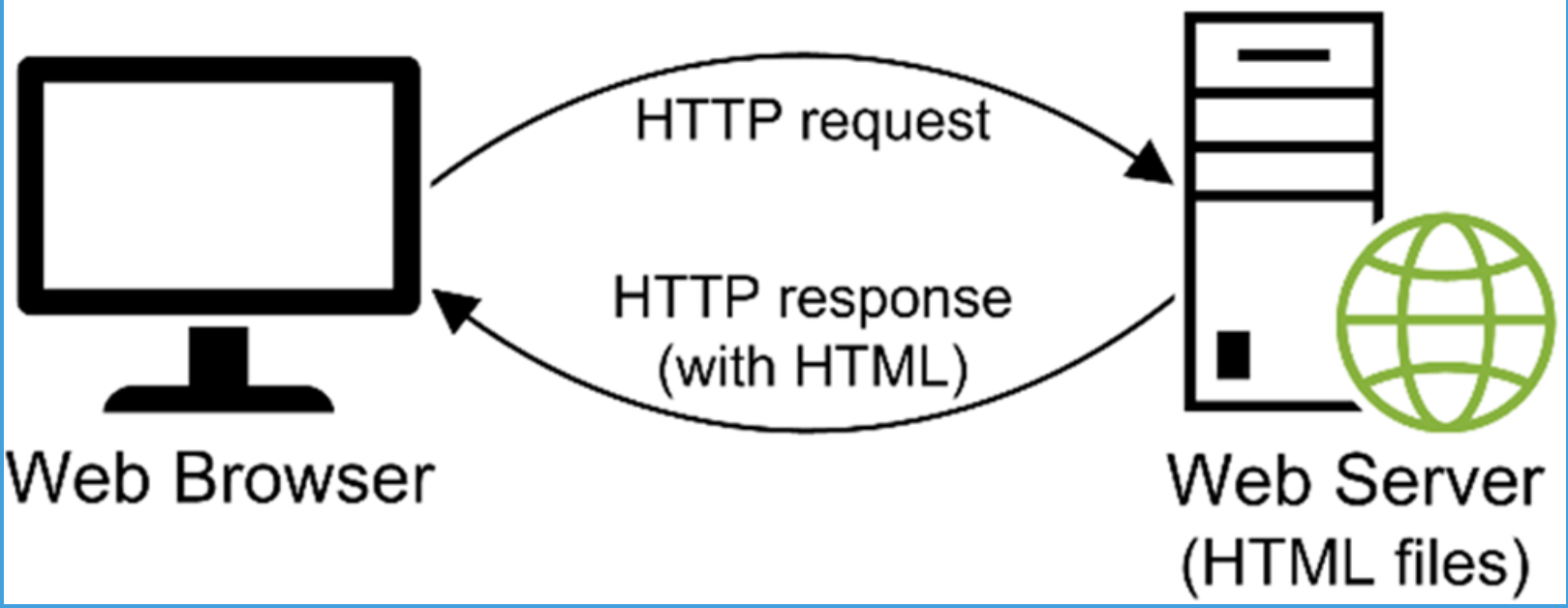
An introduction? Perhaps . . .

An overview? Maybe . . .

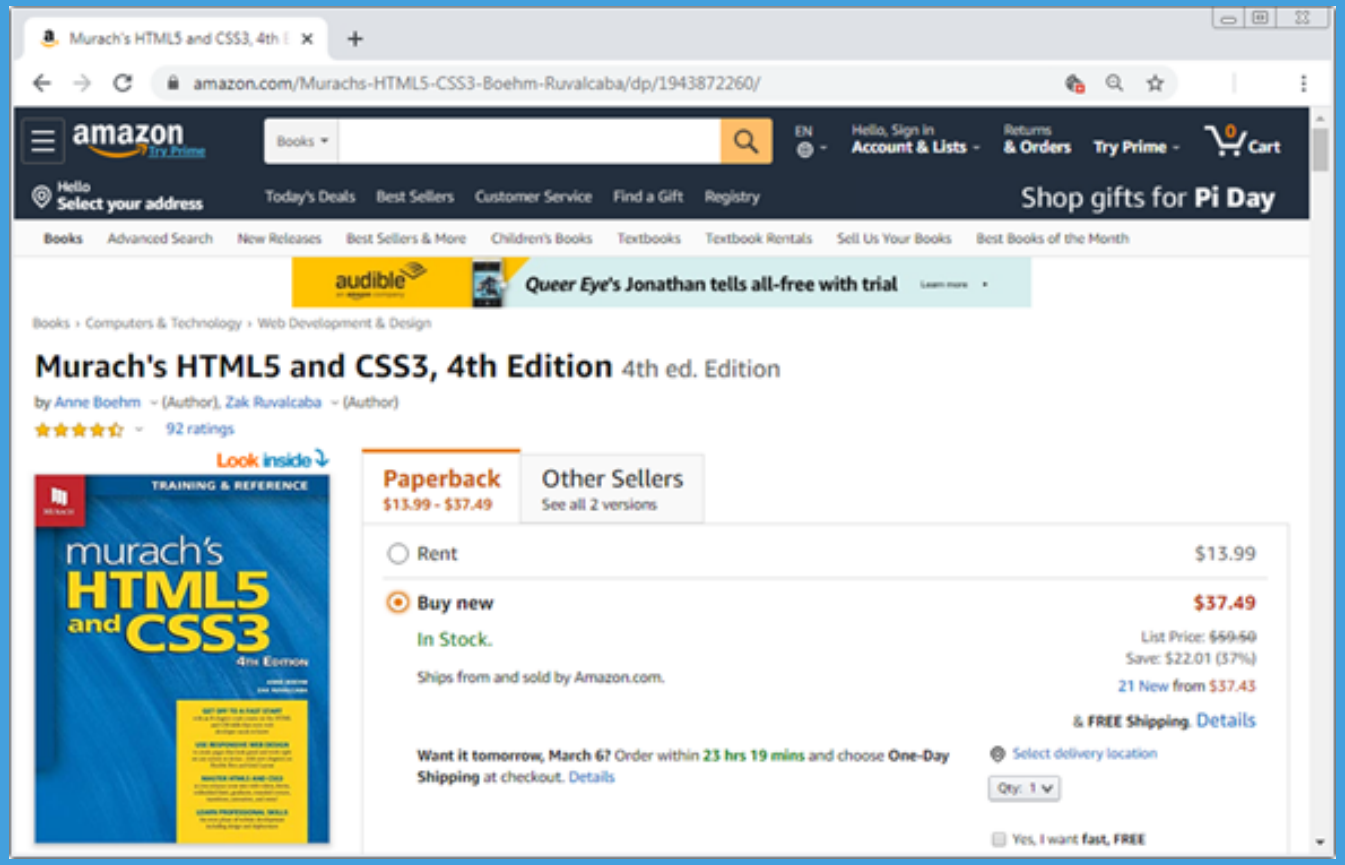
The Components of a Web Application



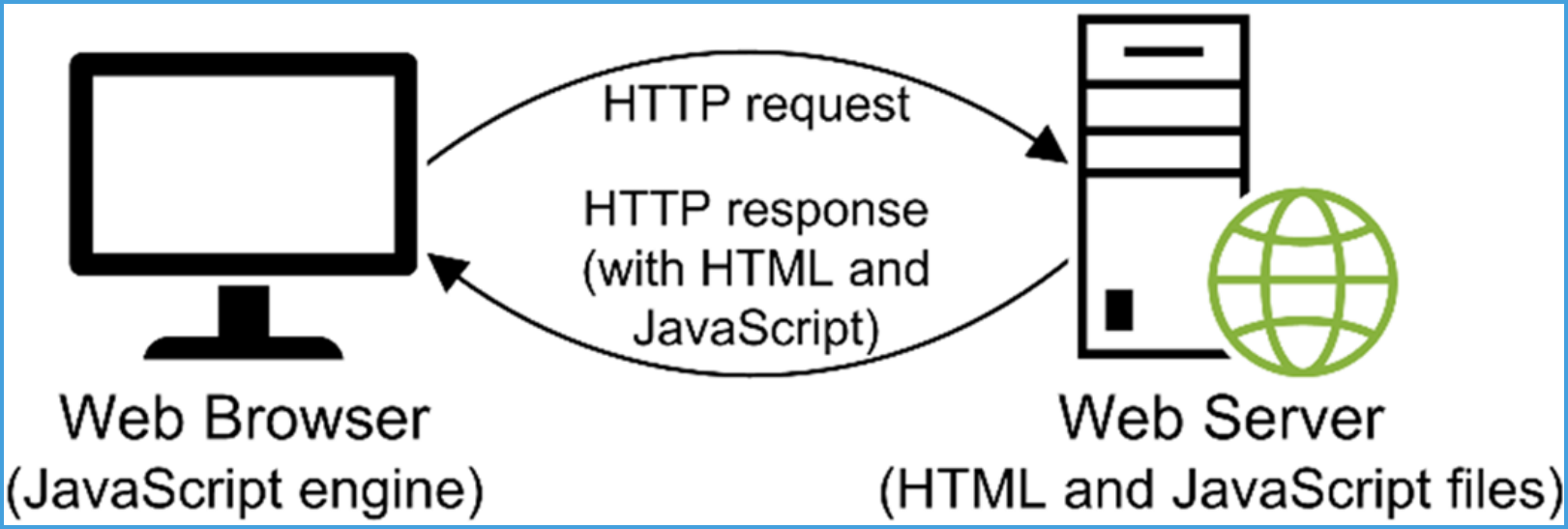
How a Web Server Processes a Static Web Page



A Dynamic Web Page at Amazon.com



How JavaScript Fits Into This Architecture



Speaker notes

It does need to be noted that JavaScript started off as a client-based scripting language, which means it only ran on the computer that was viewing the web page.

Today, though, JavaScript can be client-side and/or server-side (NodeJS is typically associated with server-side JavaScript).

Adding JavaScript to a Page

- You use the HTML <script> tag to add JavaScript to a page.
  - JavaScript can be within <script></script> tags in an HTML document.
    - This is known as inline or embedded JavaScript.

```
<!DOCTYPE html>
  <html>
  <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>Sample Page With Embedded/Inline JavaScript</script>
        <link rel="stylesheet" href="style.css">
  </head>
  <body>
      <main>
        <h1>Sample Page With Embedded/Inline JavaScript</h1>
                <script>
                        //This is JavaScript **inside** an HTML document
                        alert('This is embedded/inline JavaScript!');
                </script>
  </body>
  </html>
```

Adding JavaScript to a Page

- You use the HTML <script> tag to add JavaScript to a page.
  - JavaScript can be within <script></script> tags in an HTML document.
    - This is known as inline or embedded JavaScript.

```
<!DOCTYPE html>
  <html>
  <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>Sample Page With Embedded/Inline JavaScript</script>
        <link rel="stylesheet" href="style.css">
  </head>
  <body>
      <main>
        <h1>Sample Page With Embedded/Inline JavaScript</h1>
                <script>
                        //This is JavaScript **inside** an HTML document
                        alert('This is embedded/inline JavaScript!');
                </script>
  </body>
  </html>
```

Adding JavaScript to a Page

- You use the HTML <script> tag to add JavaScript to a page.
  - JavaScript can be within <script></script> tags in an HTML document.
    - This is known as inline or embedded JavaScript.

```
<!DOCTYPE html>
  <html>
  <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1" />
        <title>Sample Page With Embedded/Inline JavaScript</script>
        <link rel="stylesheet" href="style.css">
  </head>
  <body>
      <main>
        <h1>Sample Page With Embedded/Inline JavaScript</h1>
                <script>
                        //This is JavaScript **inside** an HTML document
                        alert('This is embedded/inline JavaScript!');
                </script>
  </body>
  </html>
```

Adding JavaScript to a Page

- JavaScript can also be placed into an external plain text file. The JavaScript is linked to the page using the src attribute of the Script tag.
  - External JavaScript is preferred over inline/embedded JavaScript – this is because external JavaScript can be used by multiple HTML documents.
  - This is external JavaScript.

```
<!DOCTYPE html>
  <html>
  <head>
        <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1"
        <title>Sample Page With Embedded/Inline JavaScript</script>
        <link rel="stylesheet" href="style.css">
  </head>
  <body>
      <main>
        <h1>Sample Page With Embedded/Inline JavaScript</h1>
        <script src="path/to/javascriptfile.js"></script>
  </body>
  </html>
```

Adding JavaScript to a Page

- JavaScript can also be placed into an external plain text file. The JavaScript is linked to the page using the src attribute of the Script tag.
  - External JavaScript is preferred over inline/embedded JavaScript – this is because external JavaScript can be used by multiple HTML documents.
  - This is external JavaScript.

```
1  <!DOCTYPE html>
2    <html>
3    <head>
4          <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1"
6          <title>Sample Page With Embedded/Inline JavaScript</script>
7          <link rel="stylesheet" href="style.css">
8    </head>
9    <body>
10       <main>
11         <h1>Sample Page With Embedded/Inline JavaScript</h1>
12         <script src="path/to/javascriptfile.js"></script>
13   </body>
14   </html>
```

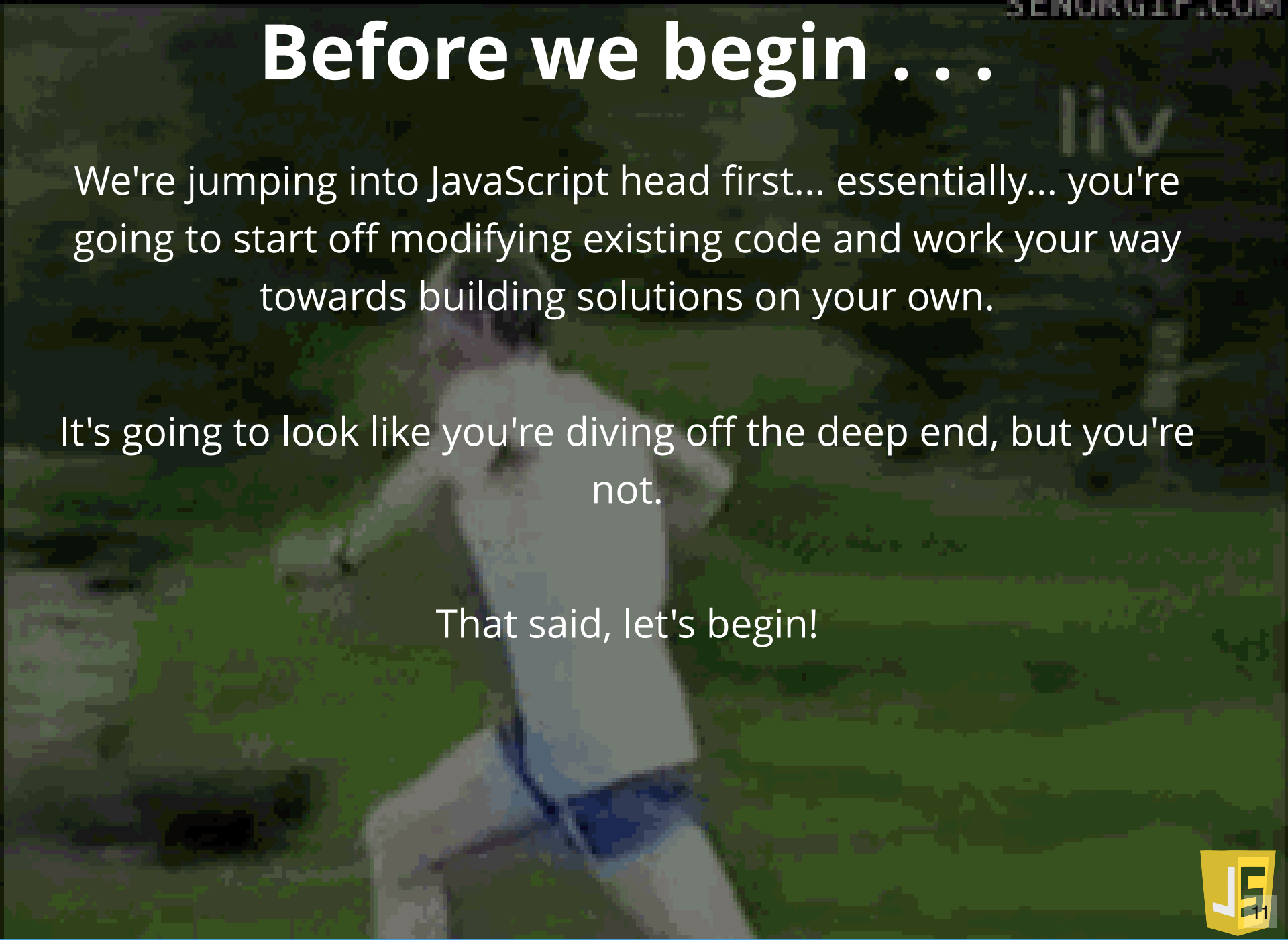
Adding JavaScript to a Page

- JavaScript can also be placed into an external plain text file. The JavaScript is linked to the page using the src attribute of the Script tag.
  - External JavaScript is preferred over inline/embedded JavaScript – this is because external JavaScript can be used by multiple HTML documents.
  - This is external JavaScript.

```
<!DOCTYPE html>
  <html>
  <head>
        <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1"
        <title>Sample Page With Embedded/Inline JavaScript</script>
        <link rel="stylesheet" href="style.css">
  </head>
  <body>
      <main>
        <h1>Sample Page With Embedded/Inline JavaScript</h1>
        <script src="path/to/javascriptfile.js"></script>
  </body>
  </html>
```

Important Note

- A <script></script> tag pair can either contain inline/embedded JavaScript ***or*** they can point to an external JavaScript file.
  - You cannot use a single <script></script> tag pair to point to an external JavaScript file, and then attempt to use inline/embedded JavaScript in the same <script></script> tag pair.
  - JavaScript will execute as soon as it's loaded in the browser. This can cause problems since your browser may reference page elements before the elements are loaded and available.
    - To work around this, the <script></script> tag pair should be at the bottom of our HTML document, before the closing </body> tag.
      - We'll eventually learn how to work around this, but this is where we're at for now 😉



Before we begin . . .

We're jumping into JavaScript head first... essentially... you're going to start off modifying existing code and work your way towards building solutions on your own.

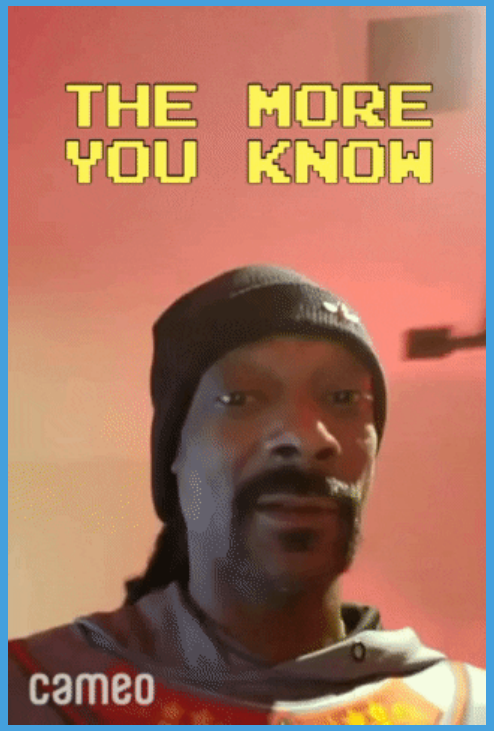
It's going to look like you're diving off the deep end, but you're not.

That said, let's begin!

Let's talk about some important things,
first . . .

Coding JavaScript Statements

- JavaScript is a ***case sensitive language***.
  - This means uppercase letters and lowercase letters are ***not*** seen as equivalents.
  - It's important to pay attention to the case of your code because careless capitalization can introduce frustrating bugs 🐜🐛

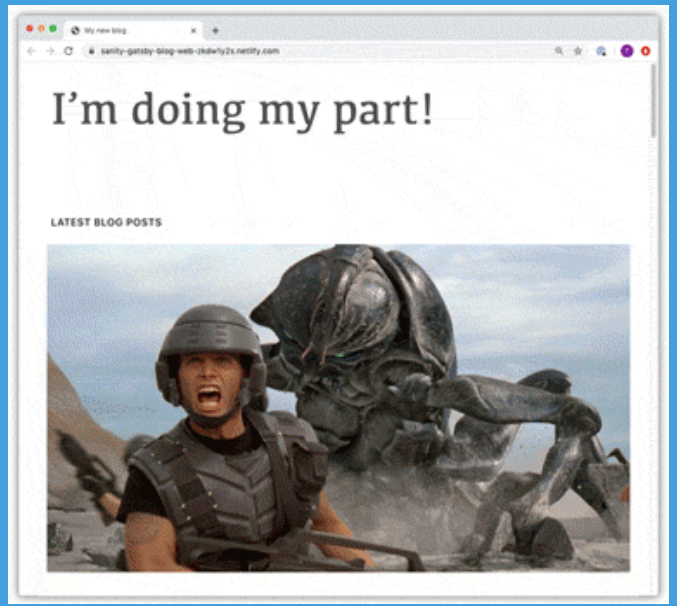


Coding JavaScript Statements

- Each JavaScript statements ends with a semicolon -- that's this character:

;

- In JavaScript, a semicolon acts like a period in a written sentence.
  - The semicolon tells the JavaScript interpreter where your JavaScript statement ends. Without the semicolon, the interpreter can become confused and merge multiple statements, which can introduce frustrating bugs 🐜🐛



Coding JavaScript Statements

- JavaScript ignores extra whitespaces within statements.
  - Some languages, like Python, use spaces and indentation to tell the computer where statements or blocks of code reside.
  - JavaScript ignores whitespace. This is important because this makes it easier for you to use whitespace to help make your code more legible/readible.

```
1  //Notice how spacing and indentation
2  //is used in the snippet below to
3  //help organize the code
4  const joinList = function() {
5      "use strict";
6
7      const emailAddress1 = $("#email_address1").value;
8      const emailAddress2 = $("#email_address2").value;
9      const firstName = $("#first_name").value;
10
11     if (emailAddress1 == "") {
12         alert("Email Address is required.");
13     } else if (emailAddress2 == "") {
14         alert("Second Email Address is required.");
15     } else {
16         $("email_form").submit();
17     }
18 };
```

Coding JavaScript Statements

- When JavaScript is in ***strict mode***, it disallows certain JavaScript features and coding practices that are considered unsafe.
  - This helps you write safer, better code.
  - To enable strict mode, add the "use strict" directive to the top of a code file.

```
1  //Notice how spacing and indentation
2  //is used in the snippet below to
3  //help organize the code
4  const joinList = function() {
5      "use strict";
6
7      const emailAddress1 = $("#email_address1").value;
8      const emailAddress2 = $("#email_address2").value;
9      const firstName = $("#first_name").value;
10
11     if (emailAddress1 == "") {
12         alert("Email Address is required.");
13     } else if (emailAddress2 == "") {
14         alert("Second Email Address is required.");
15     } else {
16         $("email_form").submit();
17     }
18 };
```

Coding JavaScript Statements

- When JavaScript is in ***strict mode***, it disallows certain JavaScript features and coding practices that are considered unsafe.
  - This helps you write safer, better code.
  - To enable strict mode, add the "use strict" directive to the top of a code file.

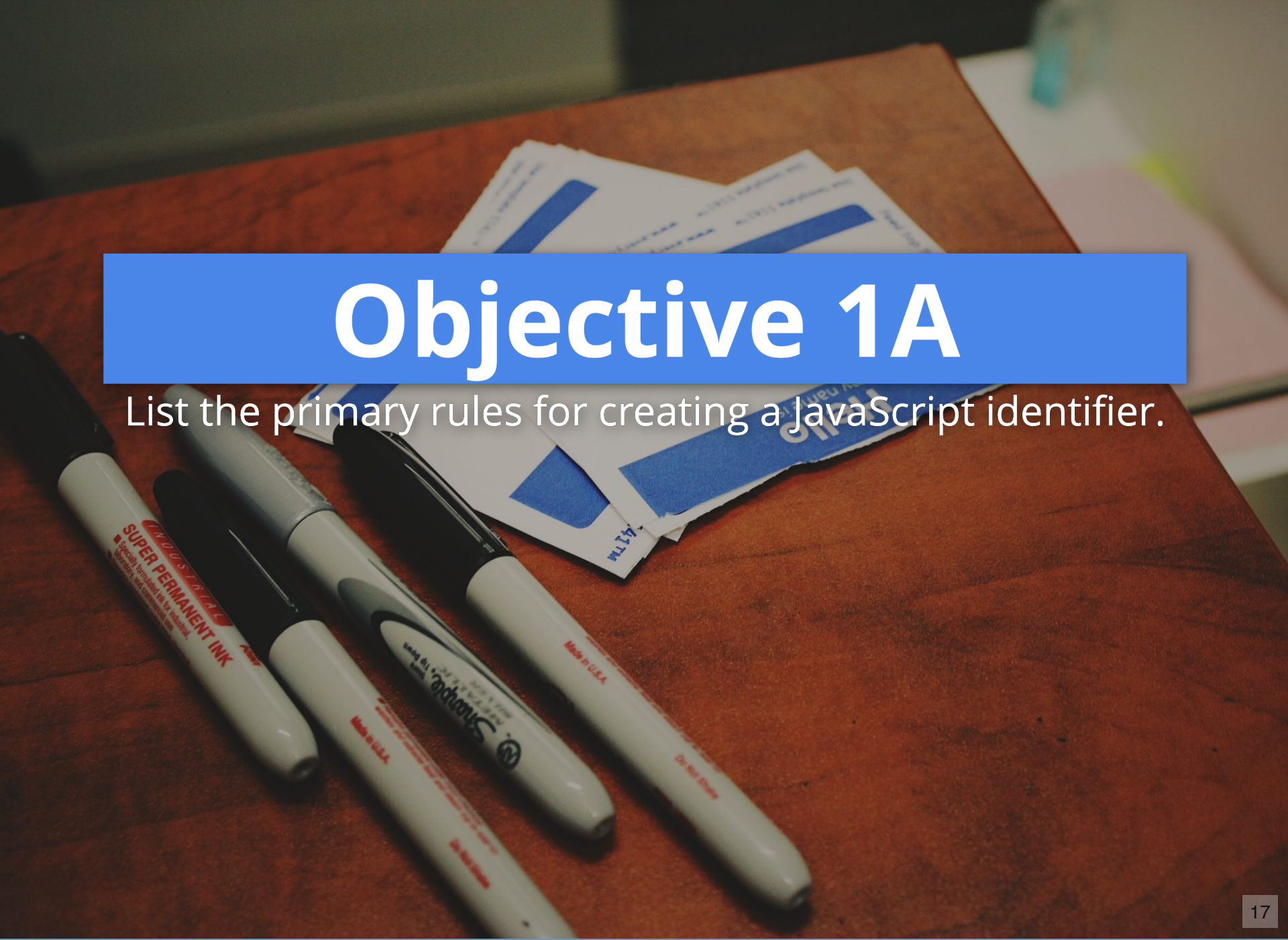
```
1  //Notice how spacing and indentation
2  //is used in the snippet below to
3  //help organize the code
4  const joinList = function() {
5      "use strict";
6
7      const emailAddress1 = $("#email_address1").value;
8      const emailAddress2 = $("#email_address2").value;
9      const firstName = $("#first_name").value;
10
11     if (emailAddress1 == "") {
12         alert("Email Address is required.");
13     } else if (emailAddress2 == "") {
14         alert("Second Email Address is required.");
15     } else {
16         $("email_form").submit();
17     }
18 };
```

Speaker notes

What does strict mode prevent?

Little things like using a variable before declaring it.

For a more detailed writeup on the topic of strict mode, see this resource:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

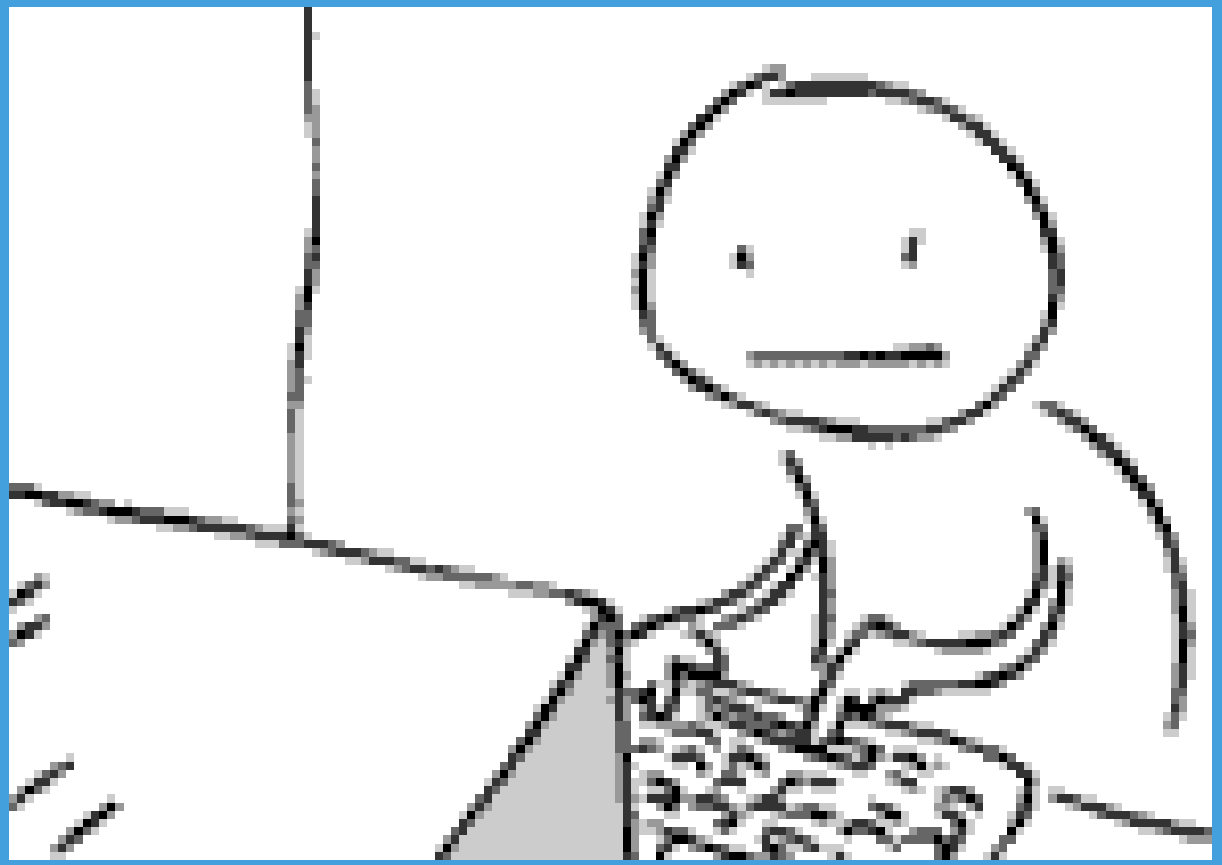


Objective 1A

List the primary rules for creating a JavaScript identifier.

How to Create Identifiers

- Variables, constants, functions, objects, properties, methods, and events must have **name**s so you can refer to them in your JavaScript code.
  - Say what!? -- we'll cover what all these things mean as we go on.
- The **name** you assign to a variable, a constant, etc. is known as an *identifier*.



Rules for Creating Identifiers

- Identifiers can only contain letters, numbers, the underscore, and the dollar sign.
- Identifiers can’t start with a number.
- Identifiers are case-sensitive.
- Identifiers can be any length.
- Identifiers can’t be the same as reserved words.
- Avoid using global properties and methods as identifiers.

Reserved Words/Keywords

- You cannot use any of JavaScript's reserved words or keywords as an identifier.
  - Reserved words/keywords are reserved for use by the JavaScript language.
- A listing of JavaScript's reserved/keywords is shown below.

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

Example Identifiers

- The following is just a small listing of valid JavaScript identifiers:
  - subtotal
  - taxRate
  - index_1 //this example includes a number, but it does not start with a number
  - calculate_click
  - $log
  - $

Camel/Snake Casing

- A ***convention*** is the way something is usually done. There are many conventions in programming.
- When creating an identifier that contains more than one word (e.g., first name), you may encounter one of the following conventions:
  - camelCasing
    - With this convention the first letter is in uppercase ***except*** for the first word.
  - snake_casing
    - With this convention, all the words in the identifier are lower case and each word is separated by an underscore character.
- You may use use whichever convention you prefer (camel or snake casing) but your ***should be consistent***.
- Random aside:
  - You'll likely see Python and PHP make heavy use of snake casing.

Naming Recommendations

- Use meaningful names for identifiers. That way, your identifiers aren't likely to be reserved keywords or global properties:
  - Example: firstName
- **Be consistent**
  - Either use camel casing (taxRate) or snake casing (tax_rate) to identify the words within the variables in your scripts.
- If you use snake casing, use lowercase for all letters.

Objective 1B

Describe the use of JavaScript comments, including “commenting out” portions of JavaScript code.

Comments

- ***Comments*** let you add descriptive notes to your code that are ignored by the JavaScript interpreter.
  - These comments can be helpful in documenting your code (e.g., what was I thinking!? 🤦)
  - Comments can also be helpful when troubleshooting issues, since comments allow you to "disable" portions of your code.
    - This is known as ***commenting out*** a portion of code.
- JavaScript supports two types of comments:
  - Single line comments
  - Block comments

NOTE

All programming languages support comments, though comments are designated different across languages.

Single-Line Comment

- Comments-out a ***single*** line of code.
- Denoted by //

```
"use strict";

const $ = function(selector) {                  // the $ function
    return document.querySelector(selector);
}
// this function gets and validates the user entries
const joinList = function() {
    const emailAddress1 = $("#email_address1").value;
    const emailAddress2 = $("#email_address2").value;
    const firstName = $("#first_name").value;
    let isValid = true;                         // set default value

    // validate the first entry
    if (emailAddress1 == "") {
        $("email_address1_error").textContent = "This field is required.";
        isValid = false;                        // set valid flag to off
    } else {
        $("email_address1_error").textContent = "";
    }

};
```

Single-Line Comment

- Comments-out a ***single*** line of code.
- Denoted by //

```
"use strict";

const $ = function(selector) {                  // the $ function
    return document.querySelector(selector);
}
// this function gets and validates the user entries
const joinList = function() {
    const emailAddress1 = $("#email_address1").value;
    const emailAddress2 = $("#email_address2").value;
    const firstName = $("#first_name").value;
    let isValid = true;                         // set default value

    // validate the first entry
    if (emailAddress1 == "") {
        $("email_address1_error").textContent = "This field is required.";
        isValid = false;                        // set valid flag to off
    } else {
        $("email_address1_error").textContent = "";
    }

};
```

Single-Line Comment

- Comments-out a ***single*** line of code.
- Denoted by //

```
"use strict";

const $ = function(selector) {                  // the $ function
    return document.querySelector(selector);
}
// this function gets and validates the user entries
const joinList = function() {
    const emailAddress1 = $("#email_address1").value;
    const emailAddress2 = $("#email_address2").value;
    const firstName = $("#first_name").value;
    let isValid = true;                         // set default value

    // validate the first entry
    if (emailAddress1 == "") {
        $("email_address1_error").textContent = "This field is required.";
        isValid = false;                        // set valid flag to off
    } else {
        $("email_address1_error").textContent = "";
    }

};
```

Single-Line Comment

- Comments-out a ***single*** line of code.
- Denoted by //

```
"use strict";

const $ = function(selector) {                  // the $ function
    return document.querySelector(selector);
}
// this function gets and validates the user entries
const joinList = function() {
    const emailAddress1 = $("#email_address1").value;
    const emailAddress2 = $("#email_address2").value;
    const firstName = $("#first_name").value;
    let isValid = true;                         // set default value

    // validate the first entry
    if (emailAddress1 == "") {
        $("email_address1_error").textContent = "This field is required.";
        isValid = false;                        // set valid flag to off
    } else {
        $("email_address1_error").textContent = "";
    }

};
```

Single-Line Comment

- Comments-out a ***single*** line of code.
- Denoted by //

```
1  "use strict";
2
3  const $ = function(selector) {                  // the $ function
4      return document.querySelector(selector);
5  }
6  // this function gets and validates the user entries
7  const joinList = function() {
8      const emailAddress1 = $("#email_address1").value;
9      const emailAddress2 = $("#email_address2").value;
10     const firstName = $("#first_name").value;
11     let isValid = true;                          // set default value
12
13     // validate the first entry
14     if (emailAddress1 == "") {
15         $("email_address1_error").textContent = "This field is required.";
16         isValid = false;                         // set valid flag to off
17     } else {
18         $("email_address1_error").textContent = "";
19     }
20
21 };
```

Block Comment

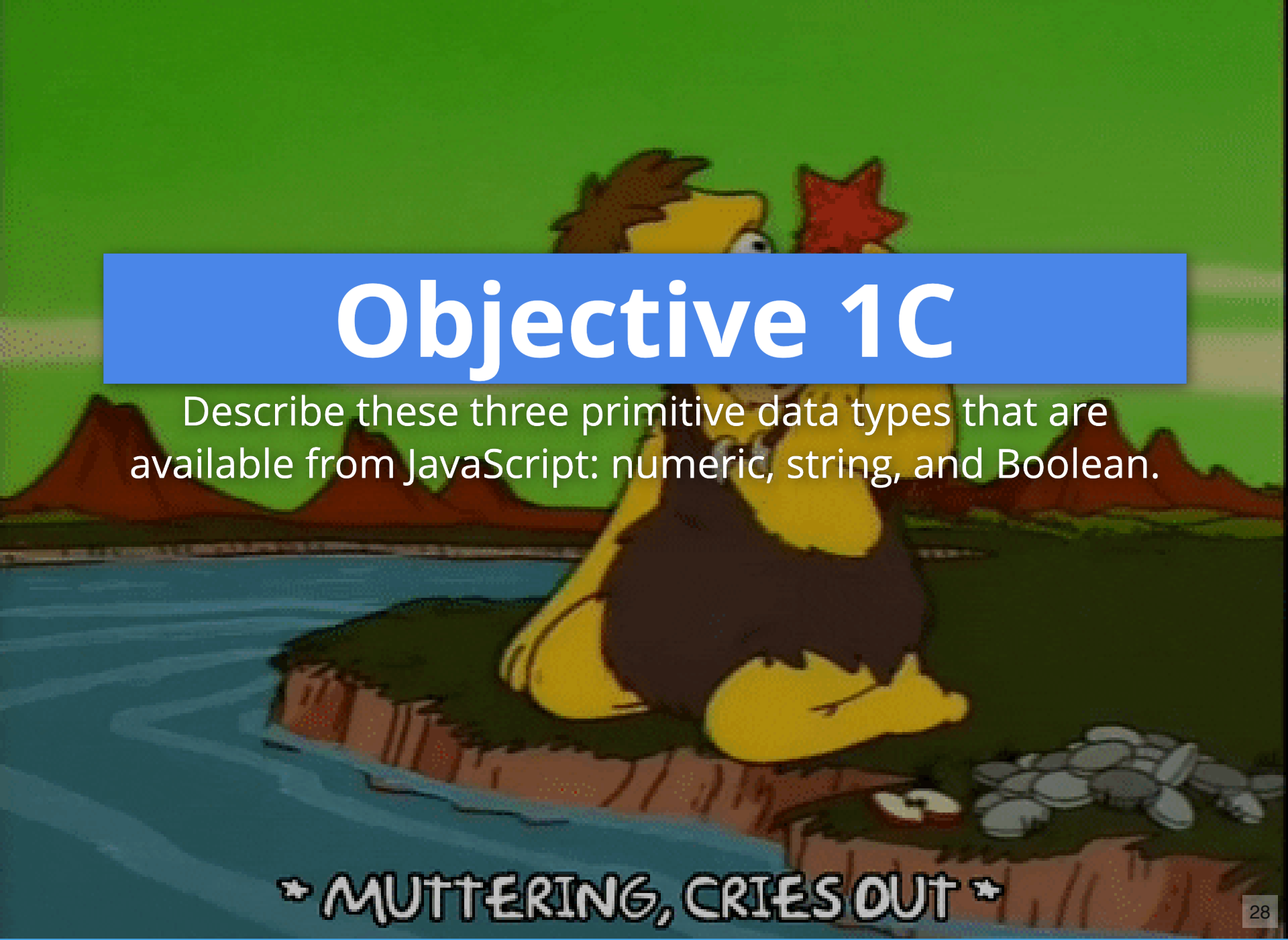
- Comments-out **one or more** lines of code.
- Started with /*
- Ended with */

```
1  /*
2     this application validates a user's entries for joining
3     our email list
4  */
5
6  "use strict";
7
8  const $ = function(selector) {                // the $ function
9      return document.querySelector(selector);
10 }
11 // this function gets and validates the user entries
12 const joinList = function() {
13         //code here
14 };
```

Block Comment

- Comments-out **one or more** lines of code.
- Started with /*
- Ended with */

```
/*
   this application validates a user's entries for joining
   our email list
*/

"use strict";

const $ = function(selector) {                  // the $ function
    return document.querySelector(selector);
}
// this function gets and validates the user entries
const joinList = function() {
        //code here
};
```

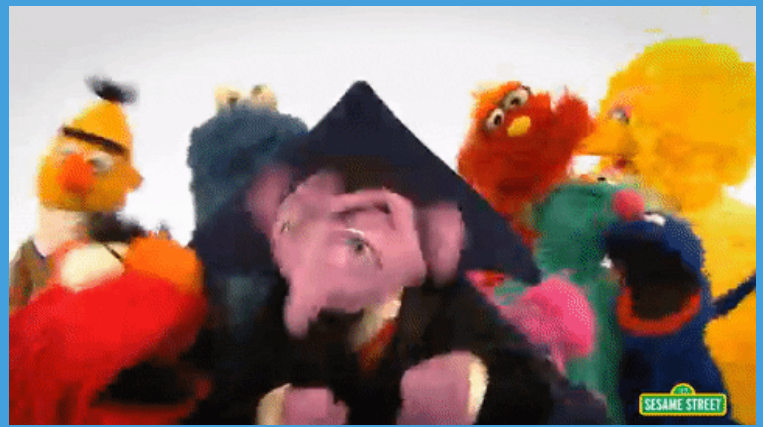

Objective 1C
Describe these three primitive data types that are available from JavaScript: numeric, string, and Boolean.
* MUTTERING, CRIES OUT *

"Data Type"

- A computer sees information as different "types" of data.
  - The data type specifies the operations that can be performed on the data.
    - For example, you can divide two numbers (e.g., 4 / 2) to obtain the quotient (2). But you cannot divide two words (e.g., cat / dog).
- JavaScript supports seven data types:
  - Number
  - String
  - Boolean
  - Symbol
  - Null
  - Undefined
  - BigInt
- Let's look at Number, String, and Boolean . . .

Number

- In JavaScript, a **Number** is an integer or a decimal value that can start with a positive or negative sign.
  - An integer is a whole number without a decimal point (e.g., 42)
  - A decimal value is also known as a **floating point number**.
- Here are some examples of number values:
  - 15 //An integer
  - -21 //A negative integer
  - 21.5 //A decimal/float
  - -124.82 //A negative decimal/floating-point number
  - -3.7e-9 //A negative decimal/floating-point number w/ neg. exponent notation



String

- In JavaScript, a **String** is character data
  - This can be a single letter (e.g., "a") or a word (e.g., "cat").
- To represent string data, you code the string within quotation marks (quotes).
  - You can use double quotes "like this" or single quotes 'like this' -- the key thing is that you need to close your string using the same type of quotation mark you started with.
  - Again, ***be consistent*** with your choices when coding.
- **Examples**:
  - "JavaScript" //A string within double quotes
  - 'String data' //A string within single quotes
  - "" //An empty string specified with double quotes
  - '' //An empty string specified with single quotes

Boolean

- In JavaScript, a **Boolean** is a value that has two possible states: true or false.
- To represent Boolean data, you code either the word *true* or *false* with no quotation marks.

```
1 //some examples:
2
3 let isLightOn = true;            //Boolean
4 let isDoorLocked = false;        //Boolean
```

Objective 1D

Describe the use of variable and constant declarations and assignment statements with numeric, string, and Boolean data.

Variables

- Let's come back to variables . . .
- A **variable** stores a value that can change as the program executes.
  - To declare a variable in JavaScript, use the **let** keyword followed by the variable name (identifier).
    - Thou shalt not use the **var** keyword to declare variables in this class.

```
1 // To declare a variable in JavaScript,
2 // use the let keyword followed by the variable name (identifier).
3
4 let firstName;
```

Variables

- Let's come back to variables . . .
- A **variable** stores a value that can change as the program executes.
  - To declare a variable in JavaScript, use the **let** keyword followed by the variable name (identifier).
    - Thou shalt not use the **var** keyword to declare variables in this class.

```
1 // To declare a variable in JavaScript,
2 // use the let keyword followed by the variable name (identifier).
3
4 let firstName;
```

Variables

- When you declare a variable, you should assign an initial value to the variable.
- To ***initialize*** a variable, you code an equals sign (=) followed by a value after the variable name.
  - The single equals sign is known as the ***assignment operator***.

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";          //This is a String value
let age = 32;                             //This is an Integer value
let weight = 110.5;                       //This is a floating-point value
```

Variables

- When you declare a variable, you should assign an initial value to the variable.
- To ***initialize*** a variable, you code an equals sign (=) followed by a value after the variable name.
  - The single equals sign is known as the ***assignment operator***.

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";          //This is a String value
let age = 32;                             //This is an Integer value
let weight = 110.5;                       //This is a floating-point value
```

Constants

- A ***constant*** stores a value that cannot be changed once it's been initialized.
  - The syntax for declaring a constant is the same as for a variable except you use the ***const*** keyword instead of ***let***.

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";        //This is a String value
let age = 32;                           //This is an Integer value
let weight = 110.5;                     //This is a floating-point value

const ssan = "123-45-5676"              //Social sec. nums don't change
```

Constants

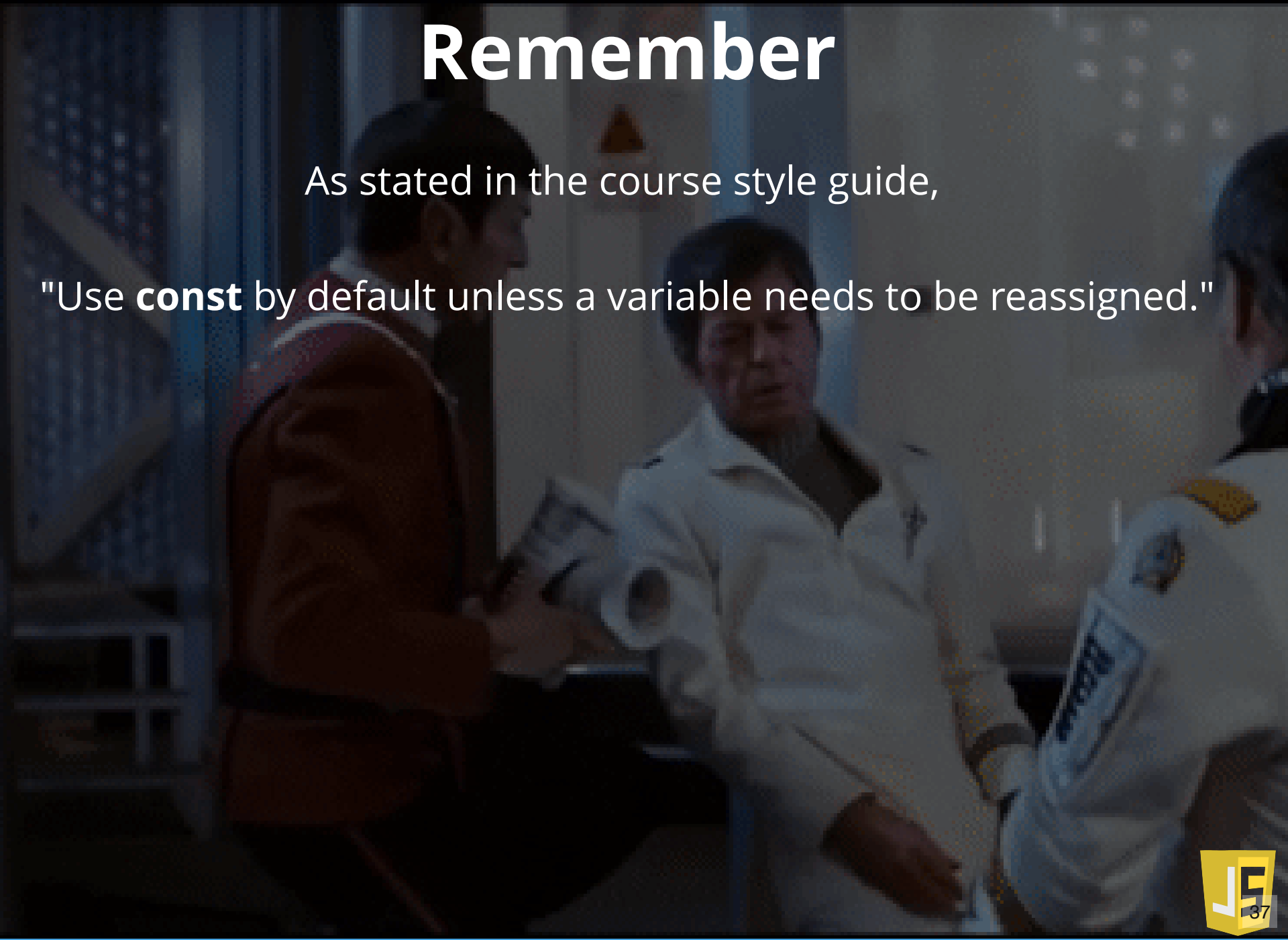
- A ***constant*** stores a value that cannot be changed once it's been initialized.
  - The syntax for declaring a constant is the same as for a variable except you use the ***const*** keyword instead of ***let***.

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";          //This is a String value
let age = 32;                             //This is an Integer value
let weight = 110.5;                       //This is a floating-point value

const ssan = "123-45-5676"                //Social sec. nums don't change
```

Remember

As stated in the course style guide,

"Use **const** by default unless a variable needs to be reassigned."



Reassigning Values to Variables

- To change the value stored in a variable, simply use the ***assignment operator*** (=) to set the variable equal to a new value.
  - REMEMBER: this only works for variables (declared with *let*) and not constants!

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";        //This is a String value
let age = 32;                           //This is an Integer value
let weight = 110.5;                     //This is a floating-point value

const ssan = "123-45-5676"              //Social sec. nums don't change

//Whoops, Bobby Bouchée is
//first and last name...
//let's fix this
firstName = "Bobby";
```

Reassigning Values to Variables

- To change the value stored in a variable, simply use the ***assignment operator*** (=) to set the variable equal to a new value.
  - REMEMBER: this only works for variables (declared with *let*) and not constants!

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";          //This is a String value
let age = 32;                             //This is an Integer value
let weight = 110.5;                       //This is a floating-point value

const ssan = "123-45-5676"                //Social sec. nums don't change

//Whoops, Bobby Bouchée is
//first and last name...
//let's fix this
firstName = "Bobby";
```

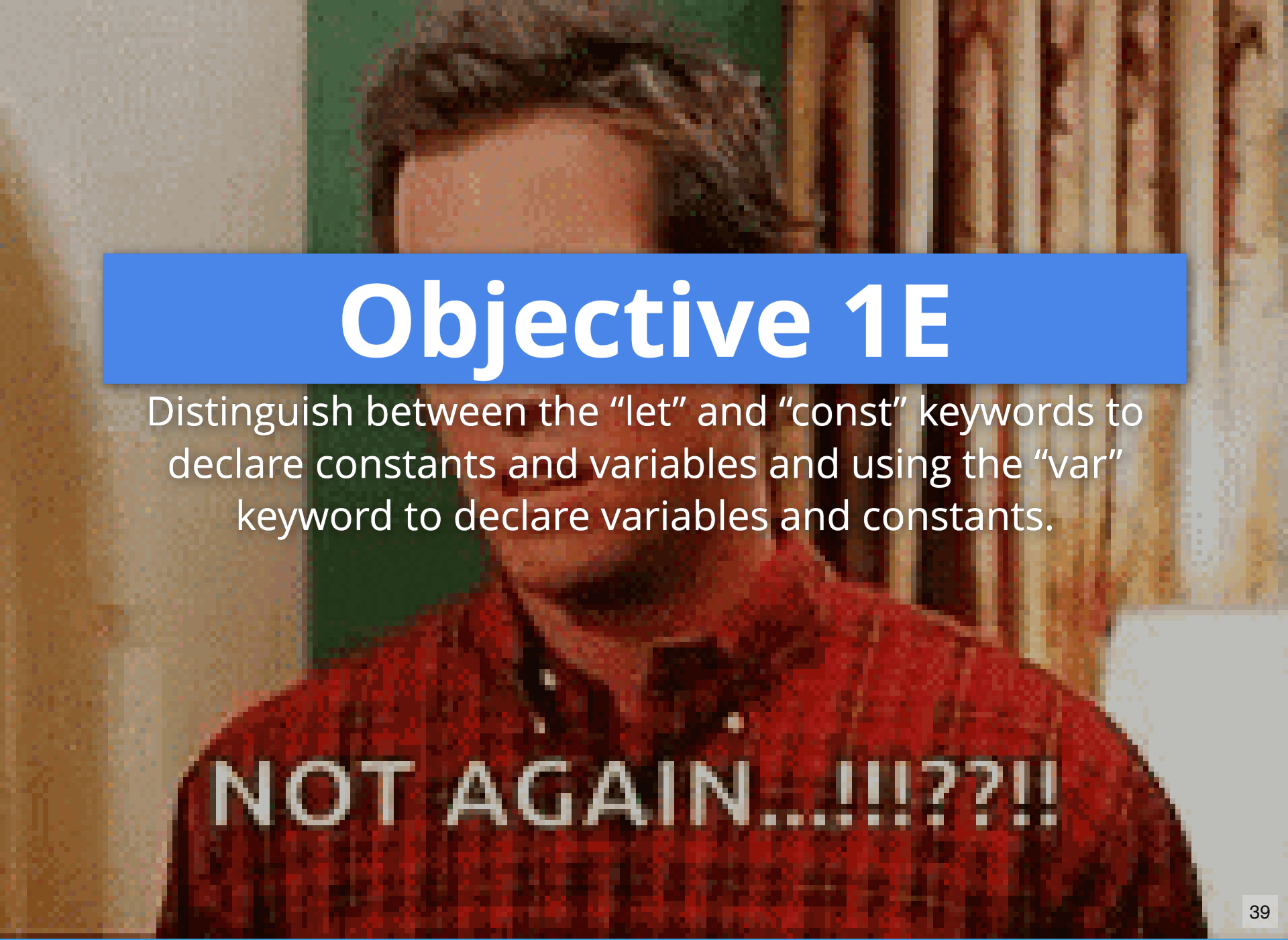
Reassigning Values to Variables

- To change the value stored in a variable, simply use the ***assignment operator*** (=) to set the variable equal to a new value.
  - REMEMBER: this only works for variables (declared with *let*) and not constants!

```
// To declare a variable in JavaScript,
// use the let keyword followed by the variable name (identifier).

// You can initialize the variable by coding
// a value after the variable name

let firstName = "Bobby Bouchée";        //This is a String value
let age = 32;                           //This is an Integer value
let weight = 110.5;                     //This is a floating-point value

const ssan = "123-45-5676"              //Social sec. nums don't change

//Whoops, Bobby Bouchée is
//first and last name...
//let's fix this
firstName = "Bobby";
```

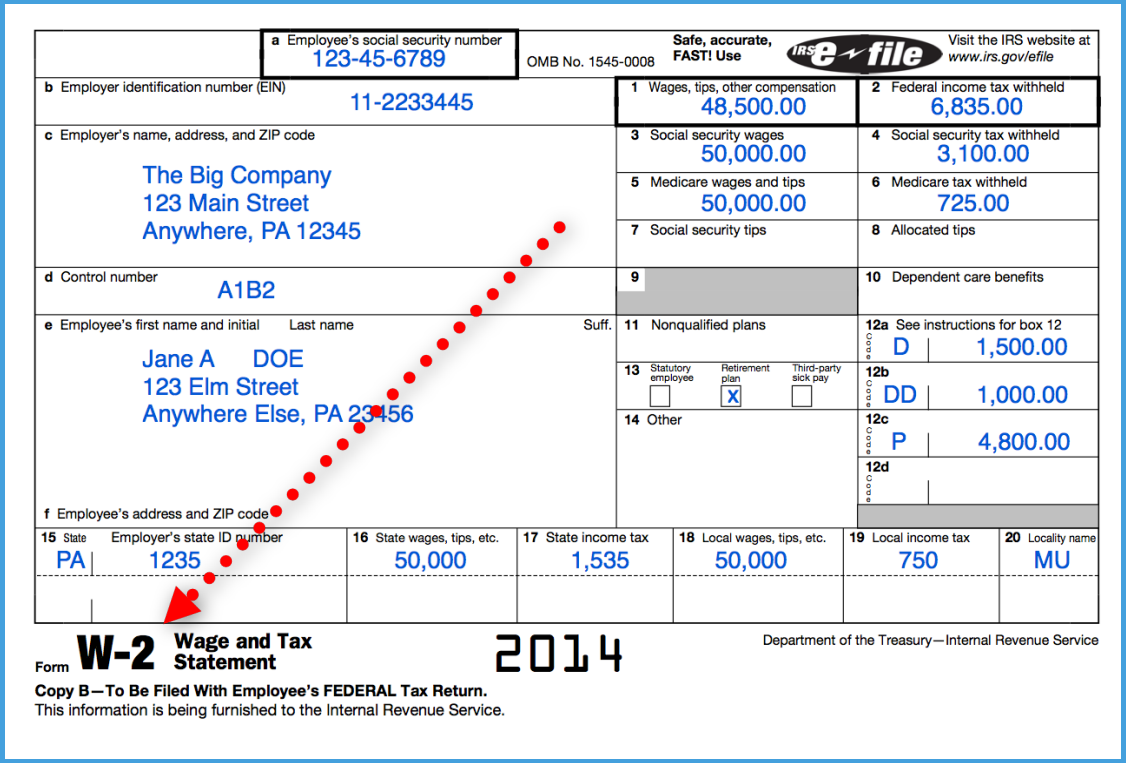
Objective 1E

Distinguish between the “let” and “const” keywords to declare constants and variables and using the “var” keyword to declare variables and constants.



Constants

- Constants are values that do not change.
  - In the form below, the form number **W-2** is a fixed value that does not change.



a Employee's social security number 123-45-6789 | OMB No. 1545-0008 | Safe, accurate, FAST! Use IRS e-file | Visit the IRS website at www.irs.gov/efile

b Employer identification number (EIN): 11-2233445

c Employer's name, address, and ZIP code:
The Big Company
123 Main Street
Anywhere, PA 12345

d Control number: A1B2

e Employee's first name and initial: Jane A | Last name: DOE | Suff.
123 Elm Street
Anywhere Else, PA 23456

f Employee's address and ZIP code

1 Wages, tips, other compensation: 48,500.00
2 Federal income tax withheld: 6,835.00
3 Social security wages: 50,000.00
4 Social security tax withheld: 3,100.00
5 Medicare wages and tips: 50,000.00
6 Medicare tax withheld: 725.00
7 Social security tips:
8 Allocated tips:
9
10 Dependent care benefits:
11 Nonqualified plans:
12a See instructions for box 12 — Code D: 1,500.00
13 Statutory employee: ☐ | Retirement plan: ☒ | Third-party sick pay: ☐
12b Code DD: 1,000.00
14 Other:
12c Code P: 4,800.00
12d Code:

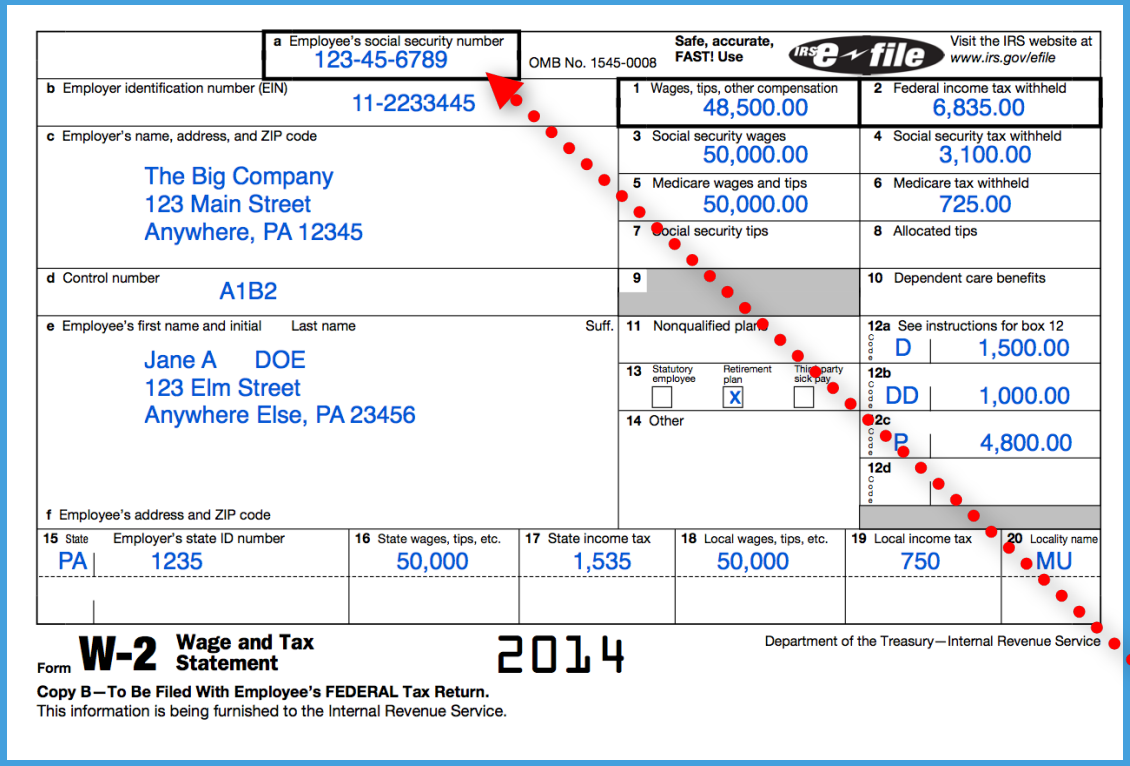
15 State	Employer's state ID number	16 State wages, tips, etc.	17 State income tax	18 Local wages, tips, etc.	19 Local income tax	20 Locality name
PA	1235	50,000	1,535	50,000	750	MU

Form **W-2** Wage and Tax Statement 2014 — Department of the Treasury—Internal Revenue Service

Copy B—To Be Filed With Employee's FEDERAL Tax Return.
This information is being furnished to the Internal Revenue Service.

Variables

- Variables are values that can change.
  - In the form below, the *Employee's social security number* (among other fields) is a value that changes for each employee, so it's variable.



a Employee's social security number 123-45-6789
OMB No. 1545-0008
Safe, accurate, FAST! Use IRS e-file
Visit the IRS website at www.irs.gov/efile

b Employer identification number (EIN) 11-2233445

c Employer's name, address, and ZIP code
The Big Company
123 Main Street
Anywhere, PA 12345

d Control number A1B2

e Employee's first name and initial Last name Suff.
Jane A DOE
123 Elm Street
Anywhere Else, PA 23456

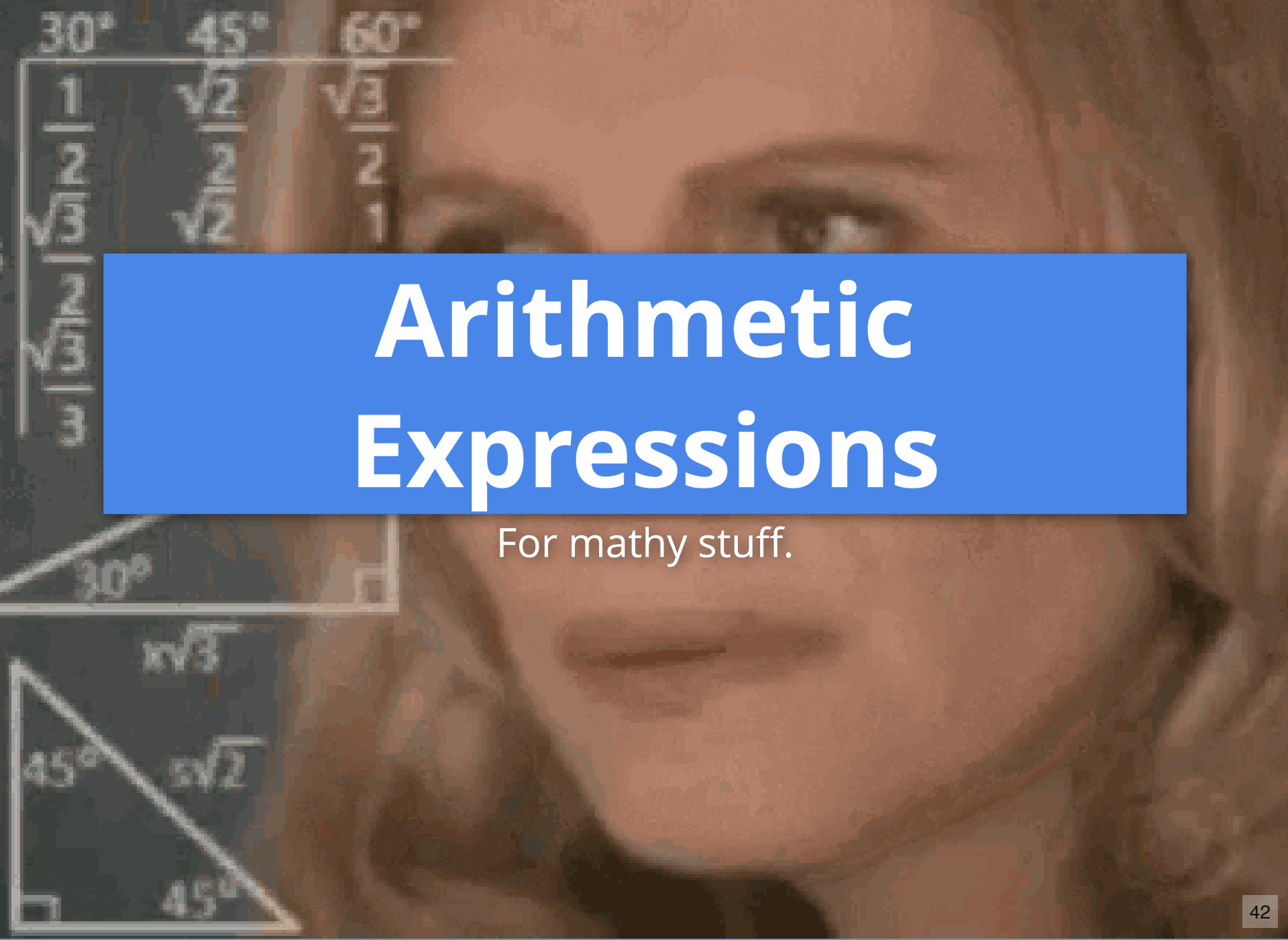
f Employee's address and ZIP code

1 Wages, tips, other compensation 48,500.00
2 Federal income tax withheld 6,835.00
3 Social security wages 50,000.00
4 Social security tax withheld 3,100.00
5 Medicare wages and tips 50,000.00
6 Medicare tax withheld 725.00
7 Social security tips
8 Allocated tips
9
10 Dependent care benefits
11 Nonqualified plans
12a See instructions for box 12 D 1,500.00
13 Statutory employee [ ] Retirement plan [X] Third-party sick pay [ ]
12b DD 1,000.00
14 Other
12c P 4,800.00
12d

15 State	Employer's state ID number	16 State wages, tips, etc.	17 State income tax	18 Local wages, tips, etc.	19 Local income tax	20 Locality name
PA	1235	50,000	1,535	50,000	750	MU

Form W-2 Wage and Tax Statement 2014
Department of the Treasury—Internal Revenue Service

Copy B—To Be Filed With Employee's FEDERAL Tax Return.
This information is being furnished to the Internal Revenue Service.



Arithmetic Expressions

For mathy stuff.

Arithmetic Expressions

- The table below shows a listing of arithmetic expression supported by JavaScript.

Operator	Name	Description
+	Addition	Adds two operands.
-	Subtraction	Subtracts the right operand from the left operand.
*	Multiplication	Multiplies two operands.
/	Division	Divides the right operand into the left operand. The result is always a floating-point number.
%	Modulus	Divides the right operand into the left operand and returns the remainder.
++	Increment	Adds 1 to the operand.
--	Decrement	Subtracts 1 from the operand.

Order or Precidence

- The table below shows the order or precedence for arithmetic operations.
  - To override the order of precedence, use parenthesis.

Order	Operators	Direction	Description
1	++	Left to right	Increment operator
2	--	Left to right	Decrement operator
3	* / %	Left to right	Multiplication, division, modulus
4	+ -	Left to right	Addition, subtraction

Coding Arithmetic Statements

- You can code arithmetic expressions using expressions with variables and constants.
  - The code snippet below shows how this can be performed.

```
1 //code that calculates tax
2 //
3
4 const subtotal = 200;
5 const taxPercent = .05;
6 const taxAmount = subtotal * taxPercent          // 10
7 const total = subtotal + taxAmount              // 210
```

Coding Arithmetic Statements

- As with your math classes, you can use parenthesis to either group arithmetic statements, or to change the order of precedence.
  - In the example below, we use parenthesis to calculate twice the width and twice the length before we add the new values together.

```
1 //code that calculates the
2 //perimeter of a rectangle
3
4
5 const width = 4.25;
6 const length = 8.5;
7 const perimeter = (2 * width) + (2 * length)            // (8.5 + 17) = 25.5
```

Compound Assignment Operators

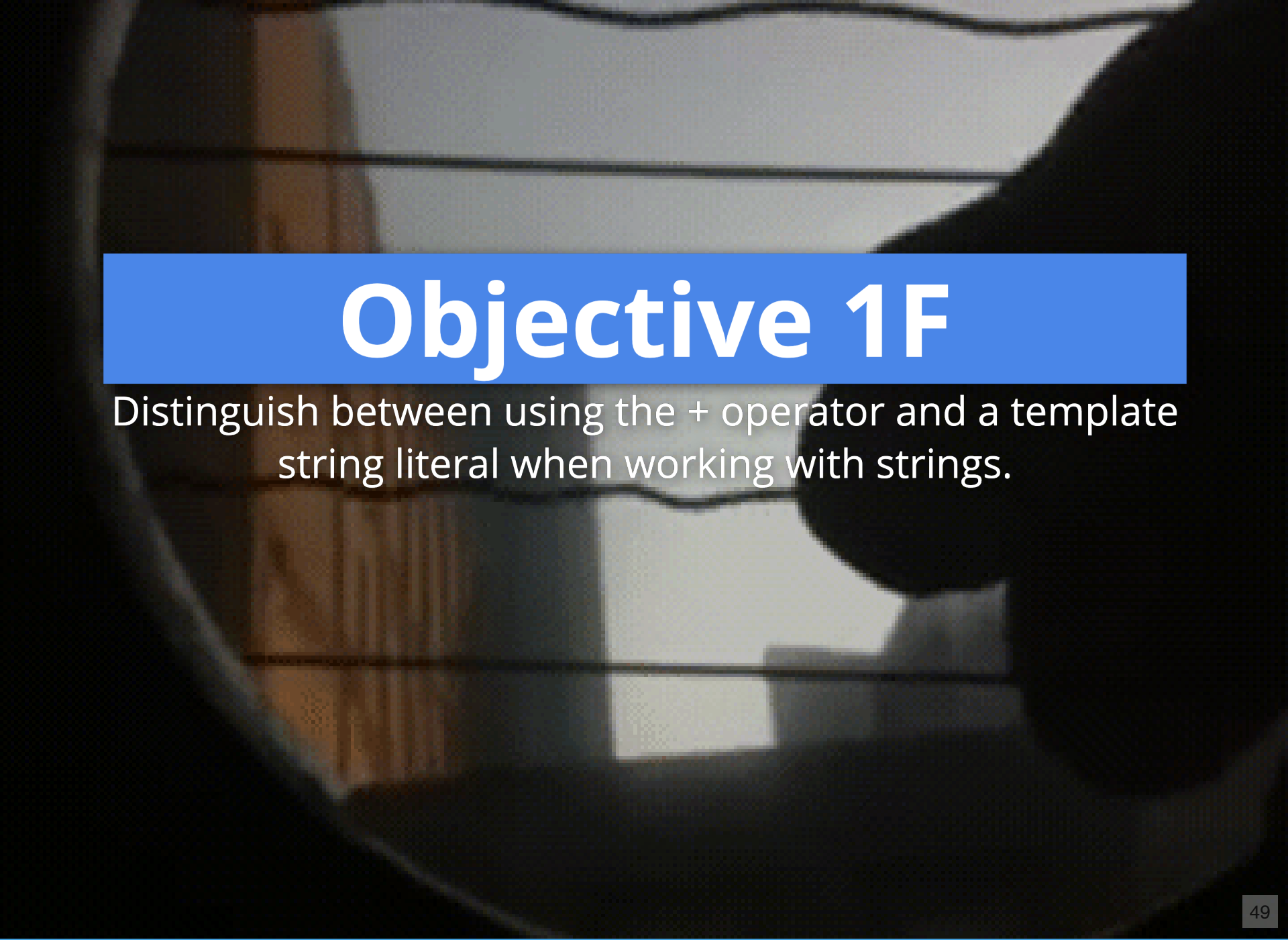
- Compound assignment operators provide a shorthand way to code common assignment statements.
  - The table below shows the compound assignment operators available in JavaScript - these mimic the compound assignment operators available in other C-like languages (e.g., C, C++, C#, etc.)

Operator	Description
+=	Adds the result of the expression to the variable.
-=	Subtracts the result of the expression from the variable.
*=	Multiplies the variable value by the result of the expression.

Compound Assignment Operators

```
1  // statements that use the
2  // compound assignment operators
3
4  let subtotal = 74.95;
5  subtotal += 20.00;                    // subtotal = subtotal + 20.00;
6                                        // subtotal = 94.95
7
8  let counter = 10;
9  counter -= 1;                         // counter = counter - 1;
10                                       // counter = 9
11
12 let price = 100;
13 price *= .8;                          // price = price * .8
14                                       // price = 80
```

You should be aware of these operators and learn to read them and use them - they are used quite a lot -- especially in loop structures.



Objective 1F

Distinguish between using the + operator and a template string literal when working with strings.

Concatenation

- ***Concatenation*** is the process of linking things together in a series.
- You can concatenate multiple strings together to create a single string value.
- To concatenate strings in JavaScript, you use the **+** operator
  - Say what!? 😱 Yep we use the plus, which is the same operator we use to add numbers together . . .

```
1 const firstName = "Bobby";
2 const lastName = "Bouchée";
3
4 const name = firstName + " " + lastName;                    //Bobby Bouchée
```

Concatenation Operator

- When used with a string, the plus symbol becomes the ***concatenation operator***.

Template Literal

- Another way to concatenate strings is to use a ***template literal***.
  - Template literals can create long, semi-formatted strings. They may be easier to work with an easier to read.
- To create a template literal, you enclose the string in tick marks: ` `
  - Note the tick is not a single quote, it's a tick, which is often found next to the "1" key on US keyboards.
- You can embed variables and constants in the template literal by enclosing the variable or constant in a set of { } which are preceded by a dollar sign:
  - ${myConstant}


```
1 const firstName = "Bobby";
2 const lastName = "Bouchée";
3
4 const name = `${firstName} ${lastName}`;                    //Bobby Bouchée
```

Working With Special Characters

- When working with strings, you may occasionally need to display special characters in your string. To do this, you will need to use ***escape sequences***.
  - The table below shows some of the escape sequences that can be used in JavaScript strings.
    - NOTE: these may not work when using JavaScript to write text to an HTML document; however, they will work when displaying a dialog, or when working with NodeJS in the console (command line).

Operator	Description
`\n`	Starts a new line in a string. Doesn't affect HTML but does work with the text in alerts and prompts.
`\'`	Puts a single quotation mark in a string.
`\"`	Puts a double quotation mark in a string.
`\\`	Puts a backslash in a string.

Unicode Characters

- Extended characters can be displayed by using the escape sequence \u followed by the Unicode value for the character you wish to display.
  - The table below shows some examples

Code	Character	Description
\u00A9	©	Copyright
\u00AE	®	Registered trademark
\u263A	☺	Smiley face
\u2665	♥	Heart

Click here for a complete listing of Unicode characters.

Fin

Make sure to review the lecture videos and lab demos before starting your homework. The videos will provide more information on topics discussed in this presentation.