Penn Engineering

ONLINE LEARNING

Video 2.2
Chris Murphy

JavaScript Basics

- Like many other programming languages, JavaScript includes:
  - variables, arrays, and objects
  - loops and conditional statements
  - functions

- Even if you know Java, there are still some important differences
  - defining functions and objects
  - interacting with HTML

Declaring a Variable

- The basic syntax for declaring any JavaScript variable is `var` *`variableName`* = …

```
var age = 22;

var name = ‘Jane Doe’;

var isMale = false;
```

Declaring a Variable

- The basic syntax for declaring any JavaScript variable is
  `var` *`variableName`* = …

```
var age = 22;

var name = 'Jane Doe';

var isMale = false;
```

Declaring a Variable

- The basic syntax for declaring any JavaScript variable is `var` *`variableName`* = …

```
var age = 22;

var name = 'Jane Doe';

var isMale = false;
```

Declaring a Variable

- The basic syntax for declaring any JavaScript variable is `var` *`variableName`* = …

```
var age = 22;

var name = ‘Jane Doe’;

var isMale = false;
```

Viewing a variable's value (1)

- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable's value in the HTML

```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```

Viewing a variable's value (1)

- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable's value in the HTML

```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```

Viewing a variable’s value (1)

- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable’s value in the HTML

```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```

Viewing a variable's value (1)

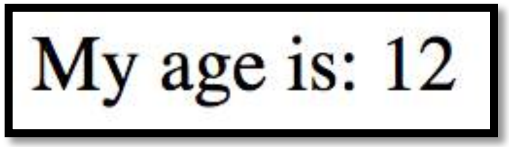
- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable's value in the HTML

```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```

Viewing a variable's value (1)

- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable's value in the HTML

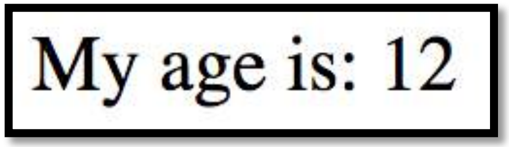
```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```


My age is: 12

Viewing a variable's value (1)

- If using a `<script>` section in a HTML file, or an external .js file, **`document.write(`*`var`*`)`** will display a variable's value in the HTML

```
My age is:
<script>
   var age = 12;
   document.write(age);
</script>
```


My age is: 12

- However, this approach is discouraged
- We will see better alternatives later!

Viewing a variable's value (2)

- You can also use **`console.log(`*`var`*`)`** to print a variable's value in the browser's JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```

Viewing a variable’s value (2)

- You can also use **`console.log(`*`var`*`)`** to print a variable’s value in the browser’s JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```

Viewing a variable's value (2)

- You can also use **`console.log(`*`var`*`)`** to print a variable's value in the browser's JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```

Viewing a variable's value (2)

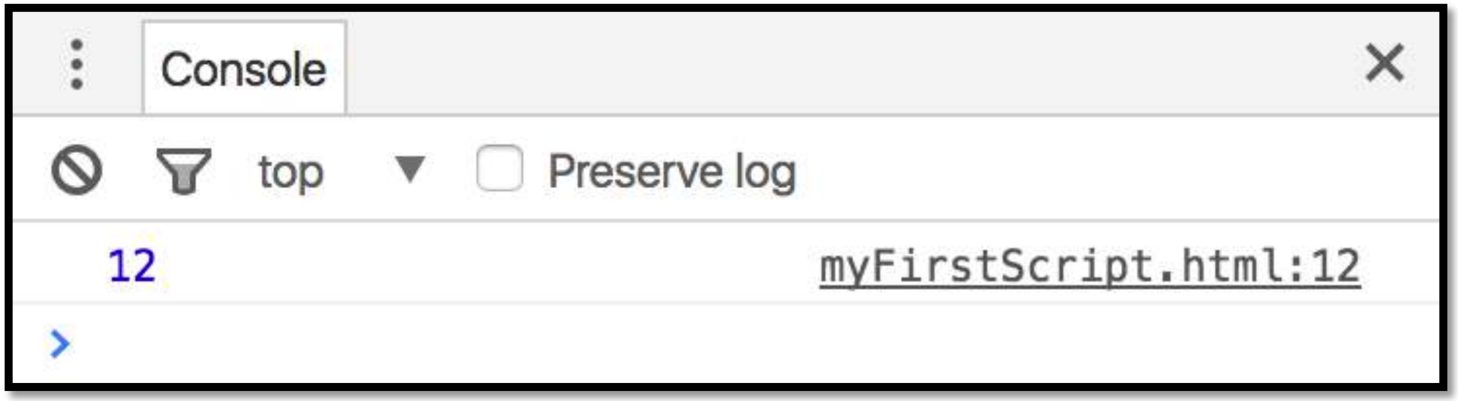
- You can also use **`console.log(`*`var`*`)`** to print a variable's value in the browser's JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```

Viewing a variable's value (2)

- You can also use **`console.log(`*`var`*`)`** to print a variable's value in the browser's JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```

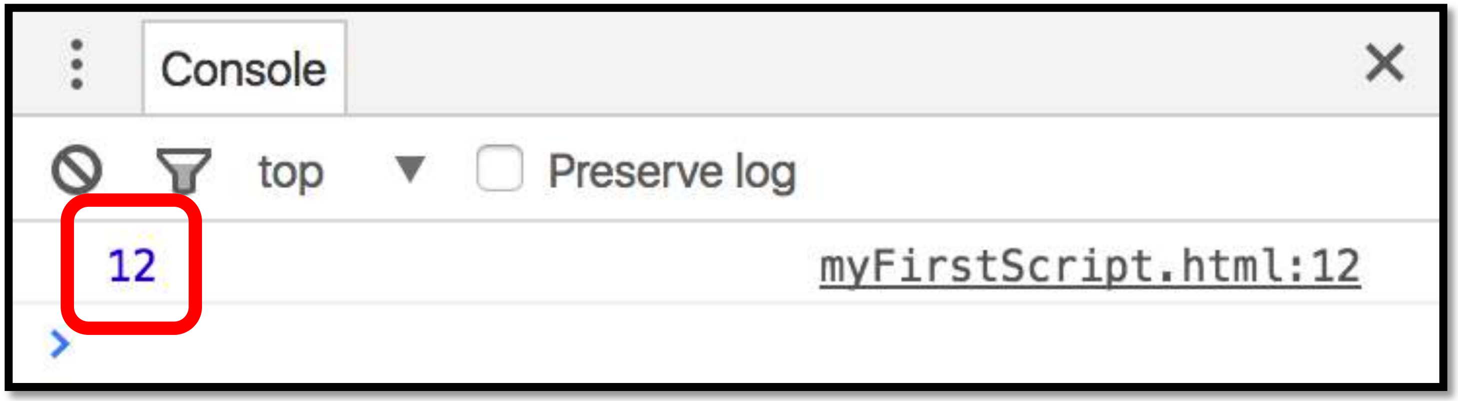


```
12                                  myFirstScript.html:12
>
```

Viewing a variable's value (2)

- You can also use **`console.log(var)`** to print a variable's value in the browser's JavaScript console

```
<script>
   var age = 12;
   console.log(age);
</script>
```



```
12                                    myFirstScript.html:12
>
```

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```

Viewing a variable’s value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable’s value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```

Viewing a variable's value (3)

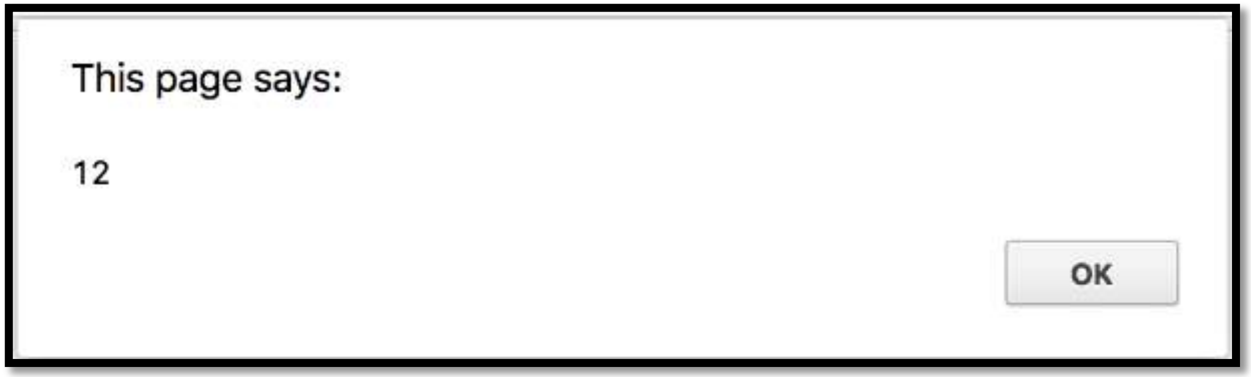
- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```

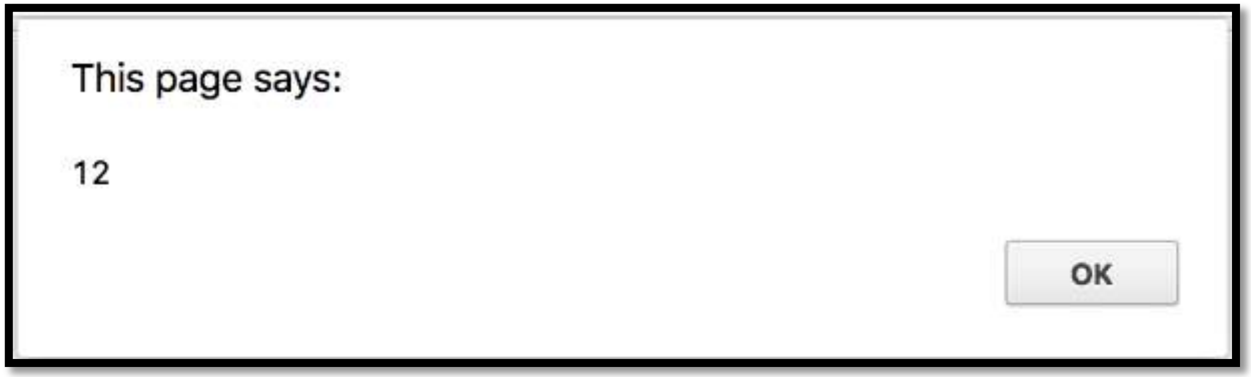

This page says:

12

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```


This page says:

12

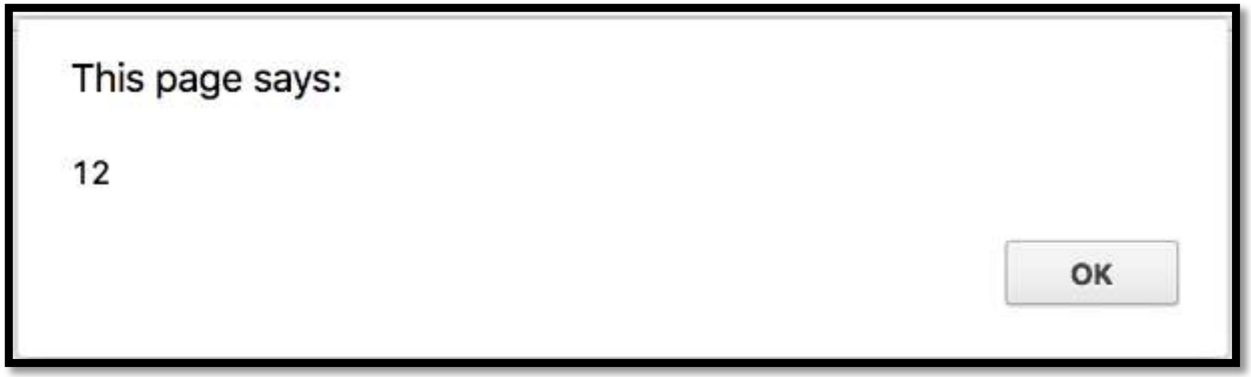
- Last, if using the browser JavaScript console (REPL), just type the name of the variable

```
> var age = 12;
```

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```



This page says:

12

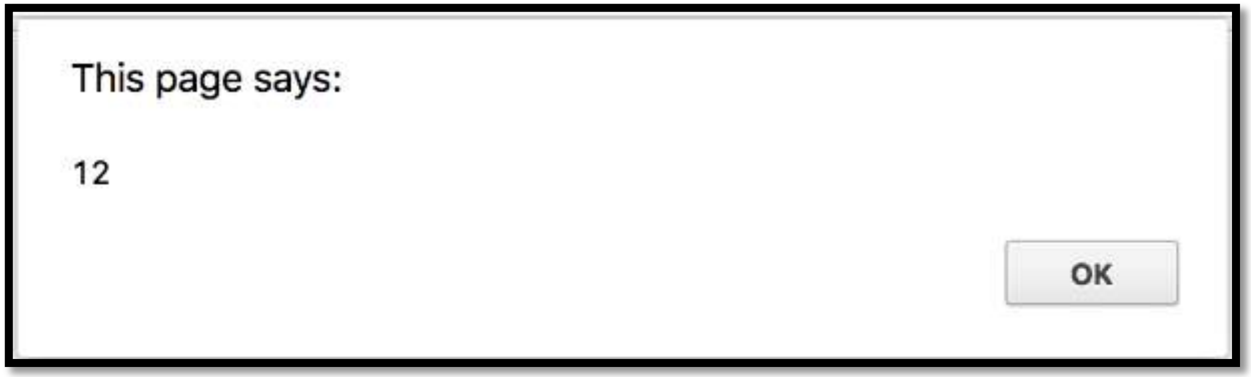
- Last, if using the browser JavaScript console (REPL), just type the name of the variable

```
> var age = 12;
> age
```

Viewing a variable's value (3)

- Also, **`alert(`*`var`*`)`** will create a popup with the variable's value that appears on top of the browser

```
<script>
  var age = 12;
  alert(age);
</script>
```


This page says:

12

- Last, if using the browser JavaScript console (REPL), just type the name of the variable

```
> var age = 12
> age
12
```

Changing a variable’s type

- The type of each variable does not need to be specified and can be changed at any time.

```
var id = 33.2;

id = ‘secret’;
```

Changing a variable’s type

- The type of each variable does not need to be specified and can be changed at any time.

```
var id = 33.2;

id = ‘secret’;
```

Changing a variable’s type

- The type of each variable does not need to be specified and can be changed at any time.

```
var id = 33.2;

id = ‘secret’;
```

Primitive Types

Type	Example values
Number	`5, 1.25, 1.1e5, +Infinity, -Infinity, NaN`

Primitive Types

Type	Example values
Number	`5, 1.25, 1.1e5, +Infinity, -Infinity, NaN`
String	`‘hello’`

Primitive Types

Type	Example values
Number	`5, 1.25, 1.1e5, +Infinity, -Infinity, NaN`
String	`‘hello’`
Boolean	`true, false`

Primitive Types

Type	Example values
Number	`5, 1.25, 1.1e5, +Infinity, -Infinity, NaN`
String	`‘hello’`
Boolean	`true, false`
Null	`null`

Primitive Types

Type	Example values
Number	`5, 1.25, 1.1e5, +Infinity, -Infinity, NaN`
String	`‘hello’`
Boolean	`true, false`
Null	`null`
Undefined	`undefined`

Numbers

- All JavaScript numbers are stored using floating-point notation
  - i.e. 5 is stored internally as 0.5e1
- `+infinity` represents all numbers greater than `Number.MAX_VALUE` (around 10^{308})
- `-infinity` represents all numbers less than `Number.MIN_VALUE` (around 10^{-324})
- `NaN` represents any non-number value
  - `Number('tree')` would return `NaN`

Number Operations

- Basic arithmetic (+, -, *, /, %) can be used on JavaScript numbers
- Precedence will follow MDAS unless parentheses are used
- ++ and -- can be used to increment/decrement JavaScript numbers

```
var a = 4;
a++;                    // a = 5
var c = a - 3;          // 2
var d = c + 3 * a;      // 17
var e = ( c + 3 ) * a;  // 25
```

Strings

- JavaScript strings are series of 16-bit unsigned integers, each integer representing a character
- Convention is to use single quotes for strings unless single quotes exist within the string
  - `‘I am a dolphin’` vs. `“I’m a dolphin”`
- Escape characters use backslash: `‘\n \t \\’`
- All JavaScript strings are immutable
  - Any manipulation results in a new string

String Functions

- `+` or `.concat(`*`otherString`*`)` can be used to concatenate strings (add them together)

```
var firstName = ‘John’;
var lastName = ‘doe’;
```

String Functions

- `+` or `.concat(otherString)` can be used to concatenate strings (add them together)

```
var firstName = ‘John’;
var lastName = ‘doe’;

var fullName= firstName.concat(‘ ’, lastName); // ‘John doe’
```

String Functions

- `+` or `.concat(`*`otherString`*`)` can be used to concatenate strings (add them together)

```
var firstName = 'John';
var lastName = 'doe';

var fullName= firstName.concat(' ', lastName); // 'John doe'
var greeting = 'HELLO, ' + fullName;
```

String Functions

- `+` or `.concat(`*`otherString`*`)` can be used to concatenate strings (add them together)
- `.toUpperCase()` and `.toLowerCase()` change the case of every character in a string

```
var firstName = 'John';
var lastName = 'doe';

var fullName= firstName.concat(' ', lastName); // 'John doe'
var greeting = 'HELLO, ' + fullName;

console.log(greeting.toUpperCase());     // 'HELLO, JOHN DOE'
console.log(greeting.toLowerCase());     // 'hello, john doe'
```

String Functions

- `+` or `.concat(`*`otherString`*`)` can be used to concatenate strings (add them together)
- `.toUpperCase()` and `.toLowerCase()` change the case of every character in a string
- *`var`*`.length` gets the length of a string

```
var firstName = ‘John’;
var lastName = ‘doe’;

var fullName= firstName.concat(‘ ’, lastName); // ‘John doe’
var greeting = ‘HELLO, ’ + fullName;

console.log(greeting.toUpperCase());     // ‘HELLO, JOHN DOE’
console.log(greeting.toLowerCase());     // ‘hello, john doe’

console.log(greeting.length);            // 15
```

Booleans

- Booleans are logical values that can only be `true` or `false`

- Any value can be used as a boolean in JavaScript
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

- Any variable type can become a boolean when used with logical operators

Null and Undefined

- **Null** is a value that can be assigned to variables to represent "no value"

```
var occupation = null;
console.log(occupation); // null
```

Null and Undefined

- **Null** is a value that can be assigned to variables to represent "no value"

```
var occupation = null;
console.log(occupation); // null
```

- **Undefined** means that a variable was declared but no value has been assigned

```
var salary;
console.log(salary);  // undefined
```

Summary

- JavaScript variables do not need to have their types specified when they are declared
- Variable types are allowed to change
- Five primitive types: number, string, boolean, null, undefined