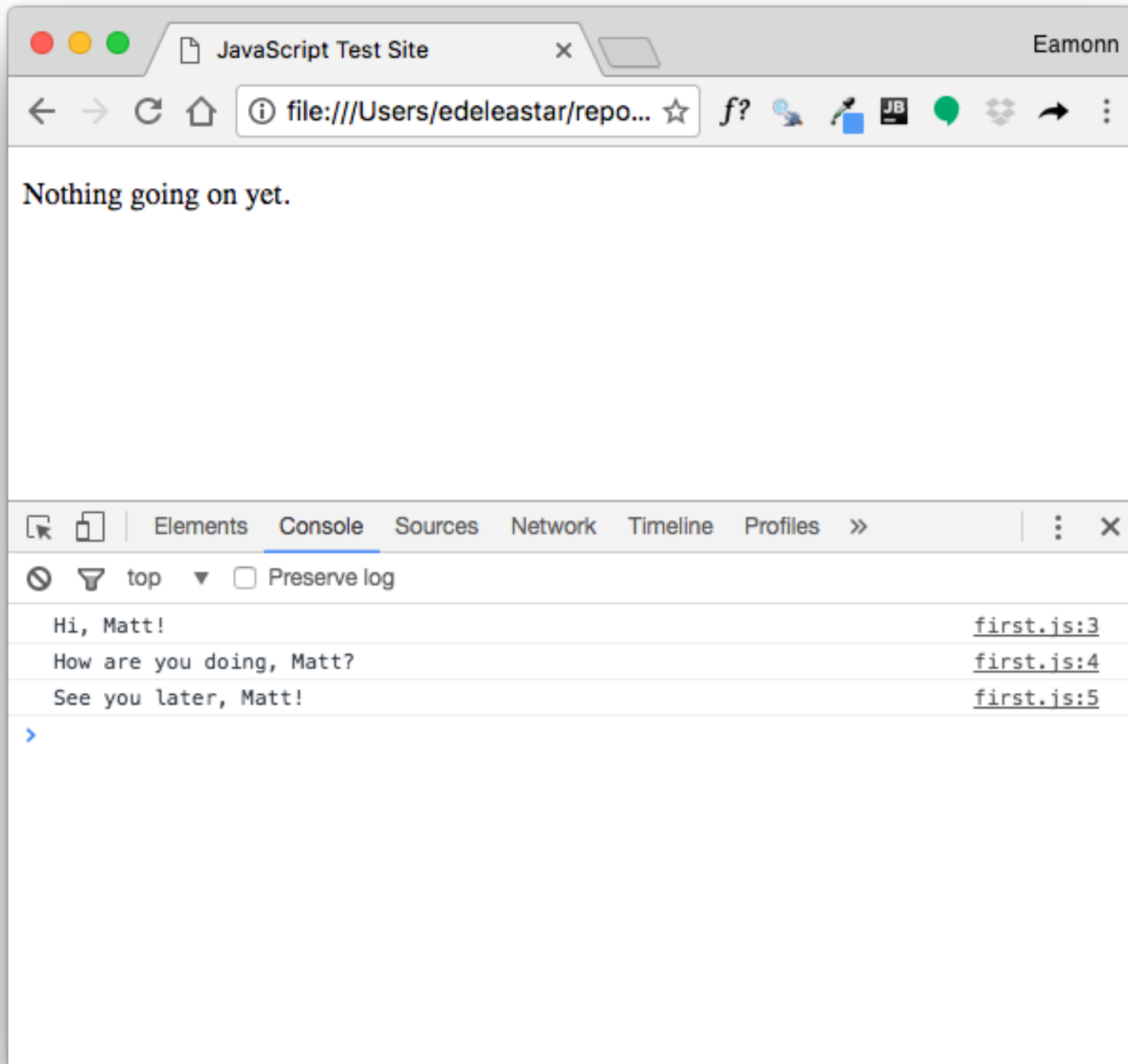# Javascript: Variables & Boolean Logic

Variables

- Initialize and assign variables in JavaScript

- Store variables using the prompt function

- Write comments in your JavaScript code

- List all of the data types in JavaScript

- Compare and contrast primitive data types with objects

Variable Fundamentals

```html
<!DOCTYPE html>
<html>
  <head>
  <title>JavaScript Test Site</title>
    <script src="first.js"></script>
  </head>
  <body>

    <p>Nothing going on yet.</p>

  </body>
</html>
```
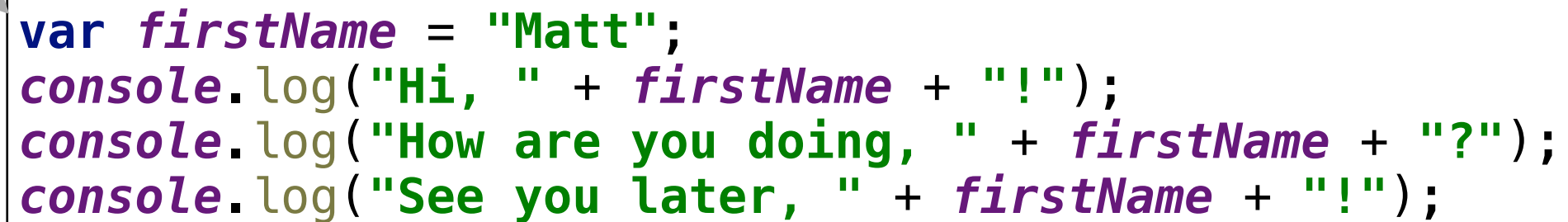
Nothing going on yet.

JavaScript Test Site

file:///Users/edeleastar/repo...

Eamonn

Elements Console Sources Network Timeline Profiles »

top ▼ ☐ Preserve log

Hi, Matt!   first.js:3
How are you doing, Matt?   first.js:4
See you later, Matt!   first.js:5

first.js

```javascript
console.log("Hi, Matt!");
console.log("How are you doing, Matt?");
console.log("See you later, Matt!");
```

- what if we want to change the person's name from "Matt" to something else?

var keyword

declare a variable

```
var firstName = "Matt";
console.log("Hi, " + firstName + "!");
console.log("How are you doing, " + firstName + "?");
console.log("See you later, " + firstName + "!");
```

use a variable

Primitive Data Types

- 6 Primitive Data Types

- JavaScript is known as a "weakly" typed language.

- This means is that when you create variables and assign them to values, you do not have to specify the type of data you are working with.

```javascript
// String
var greeting = "hello";

// Number
var favoriteNum = 33;

// Boolean
var isAwesome = true;

// undefined
var foo;
var setToUndefined = undefined;

// null
var empty = null;
```

Strings

```
// a string is a set of characters enclosed in quotes.
//A string can be defined using double quotes:
var greeting = "Hello Whiskey";

// or using single quotes:
var greeeting = 'Hello World';

// if We want quotes in a string, we can mix them, keeping them balanced:
var phrase = 'Matt said, "I have not been to Chile", the other day.';
```

Numbers

```javascript
//JavaScript numbers can be positive:
var num = 5;

// or negative:
var num = -25;
```

Decimal Numbers

```javascript
var piApproximation = 3.14159265;

var x = 1 + 3;
var a = 4.5;
var b = 5.9;
var c = Math.sqrt(a * a + b * b);
var studentTeacherRatio = 4 / 1;
```

Boolean

```
// A boolean type can only be in one of two states,
// true or false.

var pizzaIsGood = true;
var pizzaIsBad = false;
```

Undefined

```javascript
// Any variable that is created in JavaScript
// that is not assigned a value is undefined:
var noValue;  // The value here will be undefined


//You can also explicitly set a variable to undefined:
var favoriteFood = "Candy";


// Changed your mind
var favoriteFood = undefined;
```

Null

```
// Null is not the same as undefined.
It signifies an intentional absense of data.
var secondEmailAddress = null;
```

- It is important to remember that null and undefined are different types in JavaScript

- This can be a confusing feature of JavaScript, even for people who know other programming languages.

- The distinction can seem somewhat arbitrary when you're first learning the language, but as you get more comfortable the distinction will become clearer.

Figuring out a variable's type

- In JavaScript, we have a keyword called typeof that returns the type of the variable.

```
typeof "";          // - "string"
typeof 5;           // - "number"
typeof false;       // - "boolean"
typeof undefined;   // - "undefined"
typeof null;        // this is not what we expect,
                    // it returns "object"!
```

Converting to a string: toString

- The toString method will convert any value which is not undefined or null into a string

```
var num = 5;
var bool = true;

num.toString(); // "5";
bool.toString(); // "true";
```

Converting to a number using *parse*

- There are several ways you can convert a value to a number.

- One way is to parse the number, using parseInt or parseFloat:

- Each function will look at a string from left to write and try to make sense of the characters it sees as numbers.

```
parseInt("2"); // 2
parseFloat("2"); // 2
parseInt("3.14"); // 3
parseFloat("3.14"); // 3.14
parseInt("2.3alkweflakwe"); // 2
parseFloat("2.3alkweflakwe"); // 2.3
parseInt("w2.3alkweflakwe"); // NaN (not a number)
parseFloat("w2.3alkweflakwe"); // NaN (not a number)
```

Converting to a number using *Number*

- This doesn't parse, it simply tries to convert the entire string directly to a number

```
Number("2"); // 2
Number("3.14"); // 3.14
Number("2.3alkweflakwe"); // NaN
Number("w2.3alkweflakwe"); // NaN
```

Converting to a number using +

- This doesn't parse, it simply tries to convert the entire string directly to a number.

```
+"2"; // 2
+"3.14"; // 3.14
+"2.3alkweflakwe"; // NaN
+"w2.3alkweflakwe"; // NaN
```

Boolean Logic

- Write conditional logic using boolean operators

- List all of the falsey values in JavaScript

- Use if/else and switch statements to include conditional logic in your JavaScript code

- Explain the difference between == and === in JavaScript

- Convert between data types explicitly in JavaScript

Conditional Logic

- An essential part of writing programs is being able to execute code that depends on certain conditions. For example:

  - You want the navigation bar on your website to look different based on whether or not someone is logged in

  - If someone enters their password incorrectly, you want to let them know; otherwise, you want to log them in

  - You're building a tic-tac-toe game, and want to know whether it's X's turn or O's turn

  - You're building a social network and want to keep person A from seeing person B's profile unless the two of them are friends

```
var instructor = 'Brenda';

// we begin with an "if" statement
// followed by a condition in ()
// and a block of code inside of {}
if (instructor === 'Brenda') {
  console.log('Yes!');
} else {
  console.log('No');
}
```

Always true

- Notice that we used a === instead of =.

- Anytime that we use more than one equals operator (we can either use == or ===) we are doing a comparison (comparing values).

- When we use a single equals operator =, we are doing an assignment (setting a variable equal to some value).

```javascript
var favoriteFood = prompt('What\'s your favorite food?');

if (favoriteFood === 'pizza') {
  console.log('Woah! My favorite food is pizza too!');
} else {
  console.log('That\'s cool. My favorite food is pizza.');
}
```

- In this version, the boolean expression will be true/false depending on the value entered in 'prompt'

Difference between "==" and "==="

- Two different operators for comparison: the double and triple equals.

- Both operators check whether the two things being compared have the same value, but there's one important difference.

  - == allows for **type coercion** of the values,

  - === does not.

- To understand the difference between these operators, we first need to understand what is meant by **type coercion**.

Type Coercion 1

- Add a number and a string.

- In a lot of programming languages, this would throw an error, but JavaScript is more accommodating

```
5 + 'hi'; // '5hi'
```

- It evaluates the expression 5 + "hi" by first coercing 5 into a string, and then interpreting the "+" operator as string concatenation.

- So it combines the string "5" with the string "hi" into the string "5hi"

Type Coercion 2

- JavaScript expects the values inside of parentheses that come after the keyword if to be booleans.

- If you pass in a value which is not a boolean, JavaScript will coerce the value to a boolean according to the rules for **_truthy/falsey_** values (more on this later)

```javascript
if ('foo') {
  console.log('this will show up!');
}

if (null) {
  console.log('this won\'t show up!');
}
```

Type Coercion 3

- A very common way to coerce a stringified number back into a number.

- By prefacing the string with the plus sign, JavaScript will perform a coercion on the value and convert it from a string value to a number value.

```
+'304'; // 304
```

"==" Vs "===" again

== ➡️ loose   === ➡️ strict

```
5 == '5'; // true
'true' == true; // false
true == 1; // true
undefined == null; // true
```

```
5 === '5'; // false
'true' === true; // false
true === 1; // false
undefined === null; // false
```

- == allows for coercion while === doesn't.

- *If you don't want to have to think about coercion in your comparisons, stick to ===.*

```
var x = 4;
if (x <= 5) {
    console.log('x is less than or equal to five!');
} else {
    console.log('x is not less than or equal to five!');
}
```

Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Falsey Values

- Some values (aside from false) are actually false as well, when they're used in a context where JavaScript expects a boolean value

- Even if they do not have a "value" of false, these values will be translated (or "coerced") to false when evaluated in a boolean expression.

6 Falsey Values in Javascript

```
0
""
null
undefined
false
NaN // (short for not a number)
```

Logical Operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

If-Else

- Sometimes you may have more than two conditions to check.

- In this case, you can chain together multiple conditions using else

```javascript
if (number >= 1000) {
  console.log('Woah, thats a big number!');
} else if (number >= 0) {
  console.log('Thats a cool number.');
} else {
  console.log('Negative numbers?! Thats just bananas.');
}
```

Switch

- Another way to write conditional logic is to use a switch statement.

- While these are used less frequently, they can be quite useful when there are multiple conditions that can be met.

- Notice that each case clause needs to end with a break so that we exit the switch statement.

```
switch (feeling) {
  case 'happy':
    console.log("Awesome, Im feeling happy too!);
    break;
  case 'sa':
    console.log('Thats too bad, I hope you feel better soon.');
    break;
  case 'hungry':
    console.log('Me too, lets go eat some pizza!');
    break;
  default:
    console.log('I see. Thanks for sharing!');
}
```

Modulus Operator

```javascript
5 % 3 === 2 // true (the remainder when five is divided by 3 is 2)

var num = prompt('Please enter a whole number');
if ( num % 2 === 0 ) {
  console.log('the num variable is even!')
} else if ( num % 2 === 1) {
  console.log('the num variable is odd!')
} else {
  console.log('Hey! I asked for a whole number!');
}
```