COURSEWARE

Professional Skills		
Agile Fundamentals		
Jira		
Git		
Databases Introduction		
Java Beginner		
Maven		
Java Intermediate		
HTML		
CSS		
	ascript ascript	
0	What is JavaScript	
0	Getting started with JS	
0	Variables	
0	Data types	
0	ASI	
0	Strict mode	
0	Iteration	
0	Conditionals with Truthy / Falsey	
0	Objects, Arrays + JSON	
0	Structuring JS Code	
0	Destructuring	
0	Scope Functions, function expressions and arrow functions	
0	The ECMAScript 6 Specification	
0	OOP in JavaScript	
0	Best Practices	
0	Closures	
0	Callbacks and Promises	
0	Cookies	
0	Hoisting	
0	Prototypes	
0	Query Parameters	

Higher Order Functions

Variables

Contents

- Overview
- <u>Declaring variables</u>
- Rules for creating variables
 - JavaScript types
 - Data types
 - Primitive and Object types
- <u>String Concatenation and Interpolation</u>
- <u>Tutorial</u>
- Exercises
 - How to get started
 - How to find JavaScript types
 - <u>Use template literals in order to dynamically change the sentence</u>

Overview

JavaScript variables are used to hold a data value and can be changed anytime.

A JavaScript variable is a name for a storage location. In JavaScript there are three ways to declare variables:

- let a block-scoped variable (don't worry we'll discuss what block-scoped means later).
- const the same as let, but must be initialised at declaration and cannot be changed.
- var a function-scoped variable whose declaration is hoisted and can lead to confusing code! To be avoided now that we have let and const.

Declaring variables

Using the new ES6 standard we use let or const to declare variables.

x = 10;	Implicit - DO NOT USE
let y;	Explicit without assignment
let z = 10;	Explicit with assignment
const W = 10;	Const with assignment

Rules for creating variables

Variable names:

- Starts with a letter, '_' or '\$'
- Can also include digits
- Are case sensitive
- Cannot use reserved words
 - e.g. int, else, case
- Best Practice is to use camelCase for variable names
 - e.g. thisIsCamelCase

JavaScript types

Web Storage DOM Manipulation Handling Events and Timed Events Asynchronous Programming HTTP-Requests **XMLHttpRequests** Fetch API Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB **Express NodeJS** React **Express-Testing** Networking Security Cloud Fundamentals **AWS Foundations AWS Intermediate** Linux **DevOps** Jenkins Introduction Jenkins Pipeline Markdown

IDE Cheatsheet

- Dynamically typed
 - Data types not declared and not known until runtime
 - Variable types can mutate
- Interpreted
 - Stored as text
 - Interpreted into machine instructions and stored in memory as the program runs

Data types

- Primitive data types
 - Boolean
 - Number
 - BigInt
 - String
 - Undefined
 - Null
 - Symbol
- Object

Primitive and Object types

JavaScript can hold two types

- Primitives
 - o Primitive values are immutable pieces of data.
 - Their value is stored in the location the variable accesses.
 - They have a fixed length.
 - Quick to look up.
- Object
 - Objects are collections of properties.
 - The value stored in the variable is a reference to the object in memory.
 - Objects are mutable.

```
let age;
typeof(age); // Returns undefined
```

- A variable that has not been declared will also be undefined
 - The typeof operator does not distinguish between the two

```
let userID = null;
console.log(userID); //returns null
```

- null and undefined are different concepts in JavaScript
 - undefined variables have never been initialised
 - null is an explicit keyword that tells the runtime it is 'empty'

```
let userID = null;
console.log(userID == undefined); //returns true
```

- There is a foobar to be aware of with null:
 - undefined is the value of an uninitialised variable
 - null is a value we can assign to represent objects that don't exist

```
var loggedOn = false;
console.log(loggedOn); //returns false
```

Booleans can hold one of two values – true or false These are reserved words in the JavaScript.

- false is evaluated as 0.
- true is evaluated as 1.
- ▶ Number types

▶ String types

String Concatenation and Interpolation

- Adding 2 (or more strings) is an expensive operation due to the memory manipulation required
- To concatenate a string the + operator is used

```
let str1 = "5 + 3 = ";
let value = 5 + 3;
let str2 = str1 + value
console.log(str2); // 5 + 3 = 8
```

• Template literals (introduced in ES2015) allow for strings to be declared with JavaScript expressions that are evaluated immediately using \${ } notation.

```
let str2 = `5 + 3 = ${5 + 3}`;
console.log(str2); // 5 + 3 = 8
```

► String functions

Tutorial

There are no tutorials for this module.

Exercises

How to get started

- 1. Open Google Chrome
- 2. Press F12 to access developer tools.
- 3. Click console and then start each of the activities.

How to find JavaScript types

```
let a;
let b = "12345";
let c = 12344;
let d = true;
let e = {a:"JavaScript"};
```

▶ Solution

Use template literals in order to dynamically change the sentence

```
let totalMoney = 4000;
let moneyPaidSoFar = 2348;
let totalLeftToPay;

`The total bill is £4000 the remaining amount of money to be paid is £1652`
```

▶ Solution