

Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
<div>Javascript<ul style="list-style-type: none"><li>What is JavaScript</li><li>Getting started with JS</li><li>Variables</li><li><b>Data types</b></li><li>ASI</li><li>Strict mode</li><li>Iteration</li><li>Conditionals with Truthy / Falsey</li><li>Objects, Arrays + JSON</li><li>Structuring JS Code</li><li>Destructuring</li><li>Scope</li><li>Functions, function expressions and arrow functions</li><li>The ECMAScript 6 Specification</li><li>OOP in JavaScript</li><li>Best Practices</li><li>Closures</li><li>Callbacks and Promises</li><li>Cookies</li><li>Hoisting</li><li>Prototypes</li><li>Query Parameters</li><li>Higher Order Functions</li></ul></div>

## Data types

### Contents

- [Overview](#)
- [Tutorial](#)
  - [declaring variables](#)
- [Exercises](#)

### Overview

In this module we will explore JavaScript data-types.

- In JavaScript we have 8 basic data types:

Type	e.g.	Description
number	123 or 12.3456	for numbers of any kind: integer or floating-point, integers are limited by $\pm 253$ .
bigint	9007199254740991n	In JavaScript, the “number” type cannot represent integer values larger than $(2^{53}-1)$ (that’s 9007199254740991), or less than $-(2^{53}-1)$ for negatives. It’s a technical limitation caused by their internal representation. A BigInt value is created by appending n to the end of an integer:
string	"this is some text"	A string may have zero or more characters, there’s no separate single-character type.
boolean	true or false	
null	special value which represents “nothing”, “empty” or “value unknown”.	for unknown values – a standalone type that has a single value null
undefined	The meaning of undefined is “value is not assigned”.	for unassigned values – a standalone type that has a single value undefined.
object	<code>var car={type:"Audi",model="A5"};</code>	objects are used to store collections of data and more complex entities
symbol	symbol type is used to create unique identifiers for objects	

### Tutorial

- ☐ 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

## declaring variables

To declare variables we use `let` or `const` before giving it a name and then a value. In JavaScript we don't have to explicitly state the type for variables, unlike other programming languages.

```
let myName = "Ollie Tabooger"; //string
let myNumber = 20; //number
let myBigInt = 1234567891234567891234567891234567890n; //BigInt
let myBool = false; boolean
let myAge = null; //null value
let dob; //undefined
let myObject = {firstName:"Felix", lastName:"Cited"}; //object
```

String can be written using **double** or **single** quotes.

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

```
let answer1 = "It's alright";           // Single quote inside double quotes
let answer2 = "He is called 'Johnny'";  // Single quotes inside double quotes
let answer3 = 'He is called "Johnny"';   // Double quotes inside single quotes
```

Extra large or extra small numbers can be written with scientific (exponential) notation.

```
let myBigNumber = 123e5; //12300000
let mySmallNumber = 123e-5; //0.00123
```

We can also use **Arrays** in JavaScript, by using the `[]` square brackets.

```
let cars = ["BMW", "Mercedes", "Audi"];
```

JavaScript has dynamic types, which means that the same variable can be used to hold different data types. This makes it very easy compared to other programming languages as we don't explicitly set a type.

```
let number; //number is now undefined.  
let number = 1234; //number is a number  
let number = true; //number is now a boolean  
let number = "some string"; //number is now a string
```

Finally we can use the `typeof` operator in order to find out the type of the javascript variable.

```
let x = "";
typeof(x); // will return String
let y = "String";
typeof(y) // will return string also.
let z;
typeof(z) //Will have a value of undefined and a type of undefined.
let w = null;
typeof(w); // Will return Object and not of type null as expected.
let myObj = {firstName:"Felix", lastName:"Cited"};
typeof(myObj); //Will return Object.
```

## Exercises

There are no exercises for this module.