COURSEWARE

Professional Skills			
Agile Fundamentals			
Jira			
Git			
Databases Introduction			
Java Beginner			
Maven Testing (Foundation)			
		Java Intermediate	
		HTML	
CSS			
Javascript			
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		
0	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

Conditionals with Truthy / Falsey

Contents

- Overview
- IF Statement
- Checking for equality and type
- <u>Tutorial</u>
- Exercises

Overview

JavaScript variables are loosely / dynamically typed; therefore the language doesn't care how a value is declared nor changed.

Each value has an **inherent boolean value** generally known as either *truthy* or *falsey*.

All values are TRUTHY unless they are defined as FALSEY.

These variables are always FALSEY:

- false
- 0 (zero)
- '' or "" (empty string)
- undefined
- null
- NaN

Everything else is **TRUTHY**:

- '0' (String containing single digit 0)
- 'false' (String containing text 'false')
- [] (An empty array)
- {} (An empty object)
- function(){} (An 'empty' function)

IF - Statement

The below code is a template for having conditionally executed code based on a boolean condition being met.

```
if (condition) { // Boolean operation
    statement;
    // value is TRUTHY if condition is met
} else if () { // Additional OPTIONAL boolean operation
    statement;
} else {
    statement;
    // value is FALSEY if condition is NOT met
}
```

Note: It is acceptable to have many *else if* statements within the block. Each *else if* is checked in order of appearance.

► Ternary If

Checking for equality and type

Types in a type insensitive language can be interesting...

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

Seemingly different values can equate to true when using the '==' operator, for example:

```
1 == "1"; // returns true
1 == [1]; // returns true
"1" == [1]; // returns true
```

This is because JavaScript converts each string representation before comparison - only values are checked!

Now, to get an obvious false result we can use the *strictly equal operator* '===' in this case the *type* and the *value* are both checked!

```
1 === "1"; // returns false
1 === [1]; // returns false
"1" === [1]; // returns false
```

Tutorial

There are no tutorials for this module.

Exercises

1. What are the return values of the following code?

```
let strictA = true;
let strictB = 1;
```

```
console.log(strictA == strictB);
console.log(strictA === strictB);
```

- ► Solution
 - 2. Considering the following code, what will be the results be?

```
console.log(strictA != strictB);
console.log(strictA !== strictB);
```

- ► Solution
 - 3. Create a IF statement that satisfies the following:
 - Declare a variable age
 - Write a condition that checks if age is between 18 AND 65
 - Return a value in each case where the condition is satisfied and not satisfied.

Extra: Consider the case where age is less than 18 - return 'underage'.

- ► Solution
 - 4. Using ternary-if syntax, write code that checks if age is above 50.
- ▶ Solution