JavaScript Lab 2

1 Control flow

1.1 if , if else, if else if

https://www.javascripttutorial.net/javascript-if/

https://www.javascripttutorial.net/javascript-if-else/

https://www.javascripttutorial.net/javascript-if-else-if/

Files to use: `if-else-if-basic.js`

1.2 Ternary / conditional operator

https://www.javascripttutorial.net/javascript-ternary-operator/

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

Files to use: `ternary-operator.js`

1.3 switch case

https://www.javascripttutorial.net/javascript-switch-case/
Files to use: `switch-basic.js`

1.4 Loops

1.4.1 while, do - while

https://www.javascripttutorial.net/javascript-while-loop/

https://www.javascripttutorial.net/javascript-do-while/

Files to use: `while-do-while.js`

1.4.2 for loop

https://www.javascripttutorial.net/javascript-for-loop/

Using comma operator in loops
https://www.javascripttutorial.net/javascript-comma-operator/

Files to use: `for-loop.js`

1.4.3 for..of loop

https://www.javascripttutorial.net/es6/javascript-for-of/
Topic 1 - 3

Files to use: `for-of-loop.js`

1.4.4 break

https://www.javascripttutorial.net/javascript-break/

Files to use: `break-loop.js`

1.4.5 continue

https://www.javascripttutorial.net/javascript-continue/

Files to use: `continue-loop.js`

2 Implicit and explicit type coercion / conversion

Explicit type conversion / coercion is where the specified type to be converted to is explicitly specified programmatically via code. For e.g. explicitly converting a number to a string, or vice versa. Implicit coercion is where the type conversion is performed automatically in the background by the JavaScript engine.

2.1 Explicit type conversion

To convert a string to a number
https://dev.to/sanchithasr/7-ways-to-convert-a-string-to-number-in-javascript-4l

Files to use: `convert-string-number.js`

To convert a number to a string
https://medium.com/theleanprogrammer/type-conversion-and-coercion-8974afe03b85
Topic: To String Conversion

Files to use: `convert-number-string.js`

2.2 Implicit type conversion

https://www.freecodecamp.org/news/js-type-coercion-explained-27ba3d9a2839/

Key points: For the case of implicit Boolean conversion, the result of using the Boolean function to convert any particular value will result in either true or false. Thus, if a given value when converted via Boolean gives a true, it is said to have a truthy value, and a falsy value otherwise.

This is the basis of the truthy and falsy values in the table explained in:
https://www.sitepoint.com/javascript-truthy-falsy/

Topic: Truthy vs Falsy Values in JavaScript

If you are using a non-boolean operand directly within a logical context (such as within an if , if else, if else if statement without any comparison operators involved) or when Logical operators are used, for e.g.

```
if (x) {
}

if (x || y)
}
```

then implicit coercion of the operand to a boolean value occurs

For guidelines on truthy / falsy values
https://www.sitepoint.com/javascript-truthy-falsy/
Topic: Recommendations for Working with Truthy or Falsy Values

Files to use: `implicit-type-conversion.js`

3 Operators

3.1 Arithmetic operators

https://www.javascripttutorial.net/javascript-arithmetic-operators/

Files to use: `arithmetic-operator.js`

3.2 Remainder operator

https://www.javascripttutorial.net/javascript-remainder-operator/

Files to use: `remainder-operator.js`

3.3 Unary operators

https://www.javascripttutorial.net/javascript-unary-operators/
There is an error in the table for unary operators: Increment operator is ++ and decrement operator is -- (not a single - as shown)

Files to use: `unary-operator.js`

3.4 Assignment operator

https://www.javascripttutorial.net/javascript-assignment-operators/

Files to use: `assignment-operator.js`

3.5 Logical operators

https://www.javascripttutorial.net/javascript-logical-operators/
The explanation for how the && and || operator works is not very clear, the better explanation is found below.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_NOT
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_OR
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_AND

Key points:
All non-Boolean operands have a truthy or falsy value (due to implicit type conversion) when used with logical operators. The double negation (!!) used on non-Boolean operand with a truthy value returns true, and the same likewise for one with a falsy value.

When the || and && operator are used in an expression with non-Boolean operands, we use the operand's truthy or false value. The final result returned is the actual operand value itself, rather than a Boolean value.

https://dmitripavlutin.com/javascript-and-or-logical-operators/
This demonstrates the use of && for optimal evaluation when attempting to access a property or nested property at any arbitrary level within an object. If at any particular point in the evaluation either the object or one of its referenced properties is null or undefined, then the evaluation ends there, otherwise we will return with the value of the last operand - which is the actual property that we wish to access.

At any point in the evaluation of this expression, if any of the operands have the value `null` or `undefined`, then the evaluation ends and returns with that value. Otherwise, if all the operands have some valid value that evaluates to truthy, then the expression returns with the value of the final operand, which is the actual property we wish to access. There is shorter form of doing this, and that is the: Optional chaining operator ?

On the same idea, we can use || to provide default value of an object property in the event that the property does not exist. However, using || when an object does have the property required, but its value is either 0 or an empty string will return the default value instead (since either of those values are falsy). In that case, we can use the: Nullish coalescing operator ??

Files to use: `logical-operator.js`

3.6 Logical assignment operators

https://www.javascripttutorial.net/es-next/javascript-logical-assignment-operators/

Files to use: `logical-assignment-operator.js`

3.7 Comparison operator

https://www.javascripttutorial.net/javascript-comparison-operators/

Files to use: `comparison-operator.js`

3.7.1 Loose vs strict equality comparison operators

https://medium.com/jspoint/loose-vs-strict-equality-in-javascript-3e5136720b00
https://javascript.plainenglish.io/javascript-comparison-operators-loose-equality-vs-strict-equality-explained-w-3d4004625c7f

Working with strict equality comparison (===) operators

With a strict equality comparison operator (===), the result is immediately false if the two operands being compared are of different type. There is no implicit coercion performed as in the case of loose equality operators. If they are of the same type and the same value, then the result is true otherwise it is false.

Working with loose equality (==) comparison operators

This is sometimes known as abstract equality.

If you are comparing two operands which happen to have any of the values listed in the table below, then the result is immediately obvious.
https://www.sitepoint.com/javascript-truthy-falsy/
Topic: Loose Equality Comparisons with ==

For all other cases, one of the two operands is converted to the same type as the other using implicit type conversion and then their values are compared. The specific algorithm to decide how to decide which operand to convert is outlined below:

https://medium.com/@ajmeyghani/javascript-double-equals-and-coercion2-21339af9841a

The use of implicit type coercion in double equality operators allows the writing of more terse and efficient code, but it requires clear understanding of implicit type coercion. If there is no clear understanding, it s better to use strict equality comparison operators

3.8 Optional chaining operator ?

https://www.javascripttutorial.net/es-next/javascript-optional-chaining-operator/

Files to use: `optional-chaining-operator.js`

3.9 Nullish coalescing operator ??

https://www.javascripttutorial.net/es-next/javascript-nullish-coalescing-operator/

https://www.freecodecamp.org/news/javascript-optional-chaining/

3.10 in operator

https://www.freecodecamp.org/news/the-javascript-in-operator-explained-with-examples/