POWERING
POTENTIAL

Operators

JavaScript fundamentals

INTRODUCTION

In this module, you will learn about:

Operators

- Using operators
- Type conversion

Operators – Assignment and Arithmetic

Operators allow us to work with types in tasks such as

- Mathematic operations
- Comparisons

They include

- Assignment:

Assignment	=
Shorthand Assignment	+=, -=, *=, /=, %=

- Arithmetic:

Arithmetic	
Addition, subtraction	+ , -
Multiplication, division, modulus	* , / , %
Negation	-
Increment, decrement	++ , --
Power	**

Operators – Relational and Boolean

Relational and Boolean operators evaluate to true or false

- Relational:

Relational	
Less than, greater than	**< , >**
Less than or equal, greater than or equal	**<= , >=**
Equals, not equals	**==, ===, !=**

- Boolean:

Boolean	
AND, OR	**&& , ||**
NOT	**!**

The Boolean logical operators short-circuit

- Operands of && and || are evaluated strictly left to right and are only evaluated as far as necessary

Type checking

JavaScript is a loosely-typed language

```
let a = 2;
let b = "two";
let c = "2";
alert(typeof a);// alerts "number"
alert(typeof b);// alerts "string"
alert(typeof c);// alerts "string"
```

JavaScript types can mutate and have unexpected results

```
alert(a * a);// alerts 4
alert(a + b);// alerts 2two
alert(a * c);// alerts 4
alert(typeof (a * a));// alerts "number"
alert(typeof (a + b));// alerts "string"
alert(typeof (a * c));// alerts "number"
```

Quick exercise – checking for equality and type

Type in a type insensitive language can be ‘interesting’

```
let a = 2;
let b = "2";
let c = (a == b);
```

What is the value of c? true or false?

```
let a = 2 ;
let b = "2";
let c = (a === b); //returns ?
```

There is a strict equality operator, shown as ===

```
let a = true; let b = 1;
alert(a == b); // ???
alert(a === b); // ???
alert(a != b); // ???
alert(a !== b); // ???
```

Type conversion

Implicit conversion is risky – better to safely convert

You can also use explicit conversion

- eval( ) evaluates a string expression and returns a result
- parseInt( ) parses a string and returns an integer number
- parseFloat( ) parses a string, returns a floating-point number

```
let s = "5";
let i = 5;
let total = i + parseInt(s); //returns 10 not 55
```

You can also check if a value is a number using isNaN()

```
isNaN(s); // returns true
!isNaN(i); //returns true
```

QuickLab 2 - Operators

- Exploring operators and types
- Arithmetic types
- Relational operators
- Assignment operations
- Type mismatching and conversion

REVIEW

Operators

- You use operators to manipulate data including its type