# Variables

- What is a variable?
- Declaration and Initialisation
- Naming rules
- Data types
- Constants
- **var** vs **let** vs **const**
- null and undefined
- Expressions

# Variables - What is a variable?

- A variable is a container for a value
- A value can be as simple as a number, a string, a boolean (True or False), arrays (represent arrays/lists of values), or objects like dates, a person or a car or a house
- Examples
  - Numbers: 1, 10, 12.5, 13.843
  - Strings: ‘Hello’, “How are you?”
  - Dates: a timestamp 1696800750, a Date “2020-05-09”
  - Boolean: true, false
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables#what_is_a_variable

# Variables - Declaration and Initialisation

- A variable is declared and then initialised. It can be declared and initialised at the same time.
  - Declaration: let myVariable;
  - Initialisation: myVariable = 1
  - Declaration + Initialisation: let otherVariable = 1;
- References
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

# Variables - Naming rules

- Recommended to use Latin characters (0-9, a-z, A-Z) and the underscore character
- Recommended convention to use is lower camel case
  - myVariable
  - someVariableWithALongName
- Do not use numbers or underscores at the beginning of variable names
  - Numbers at the beginning of the name are not allowed
  - Underscores are allowed but they can cause confusion with JavaScript's own internal definitions
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables#an_aside_on_variable_naming_rules

# Variables - Data types - Numbers

- A data type for integer and decimal values.
- Integers: floating-point numbers with no fraction. They can be positive or negative.
- Floating point numbers (floats): they have decimal points and decimal places. They can be positive or negative.
- Doubles: greater precision than floats. They can hold more information. They can be positive or negative.
- Common operations
  - Addition (+), subtraction(-), multiplication (*), division (/), remainder (%), increment (++), decrement(--), comparison (==, ===, !=, !==, <, >, <=, >=,
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Math

# Variables - Data types - Strings

- A data type for sequences of characters
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Strings

# Variables - Data types - Boolean

- A data type representing one of two possible values: True or False
- Usually used to test a condition

# Variables - Data types - Arrays

- Containers of zero, one or more values.
- Used to represent lists, stacks or queues and other data structures
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays

# Variables - Data types - Objects

- Containers for data in the form or key-value pairs
- Can hold one or more name-value pairs, also referred to as “properties”. You may also hear them referred to as “attributes”.
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics

# Variables - **var**, **let** and **const**

- var, let and const are all JavaScript key words
- They are used to declare variables
- var
  - When JS was created, it was the only way to declare variables
  - Its design can be confusing and is error-prone
  - Variables can be declared with **var** multiple times
  - Variables declared with **var** are visible in all scopes (we will discuss scopes shortly)
  - Variables could be declared with **var** after they have been used
- let
  - Declares a variable and corrects the behaviour described for **var**
  - Variables declared with let can be updated
- const
  - Declares a constant which are like variables but they need to be initialised as soon as they are declared and they cannot be assigned a new value after they have been initialised.
  - Principle: "*Use const when you can, and use let when you have to.*"
- References
  - https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Variables

# Variables - null and undefined

- null
  - null is a value that expresses the intentional absence of value in a variable or constant
- undefined
  - undefined is the type of any variable that has not been assigned a value i.e. that has not been initialised
- References
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/null
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined

# Expressions

- This section covers common expression and operators used in the JavaScript language. We will focus on the items in bold:
  - **Primary expressions**
  - Left-hand-side expressions
  - **Increment and decrement**
  - **Unary operators**
  - **Arithmetic operators**
  - **Relational operators**
  - **Equality operators**
  - Bitwise shift operators
  - Binary bitwise operators
  - **Binary logical operators**
  - Conditional (ternary) operator
  - **Assignment operators**
  - Yield operators
  - Spread syntax
  - Comma operator
- References
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators

# Expressions - Primary expressions

- **Literals**
  - numbers, string literals, boolean values, the **null** value
- **[ ]**
  - Array initialiser
- **{ }**
  - Object initialiser
- **function**
  - Defines a function expression
- **( )**
  - Grouping operator. Useful to group other expressions to show clear intent and logic.
  - For example: 1 + 2 * 5 is the same as 1 + (2 * 5) but it is different than (1 + 2) * 5
- **`string`**
  - Template literal
  - For example:
    - let name = “John Doe”
    - `Have a nice day ${name}!!!` would result in the string “Have a nice day John Doe!!!”

# Expressions - Left-hand-side expressions

- **new**
  - Creates an instance of a constructor
  - For example:
    - let today = new Date()
    - let john = new Person()
    - let dog = new GoldenRetriever()
    - let supermaket = new Business()

# Expressions - Increment and Decrement

- **++A**
  - Prefix increment. Increment number variable A by 1. Increment happens first, then the value of A is used.
- **A++**
  - Postfix increment. Increment number variable A by 1. The value of A is used first, then increment happens.
- **--A**
  - Prefix decrement. Decrement number variable A by 1. Decrement happens first, then the value of A is used.
- **A--**
  - Postfix decrement. Decrement number variable A by 1. The value of A is used first, then decrement happens.

# Expressions - Unary operators

- **typeof**
  - Determines the type of a given object
  - For example:
    - let student = new Person()
    - typeof studend
- **+**
  - Converts its operant to Number type
- **-**
  - Converts its operant to Number type and negates it
  - For example:
    - let n = 5
    - -n // would result in the value -5
- **!**
  - Logical not operator. Inverts boolean values or the boolean result of a given expression
  - For example
    - let isGreater = 6 > 3 // results in isGreater having the value **true**
    - let isNotGreater = ! isGreater // results in isNotGreater having the value **false**

# Expressions - Arithmetic operators

- **
  - Exponentiation operator
  - For example: 2 ** 3 = 2 * 2 * 2 = 8
- *
  - Multiplication operator
- /
  -
- %
  - Remainder operator
  - For example: 10 % 7 = 3 // 10 divided by 7 is 1 with a remainder of 3
- +
  - Addition operator
- -
  - Subtraction operator

# Expressions - Relational operators

- <
  - Less than operator
- >
  - Greater than operator
- <=
  - Less than or equal operator
- >=
  - Greater than or equal operator
- instanceof
  - Determines whether an object is an instance of another object
  - Keep in mind that literal types can be checked with the ‘typeof’ unary operator
- in
  - Determines whether an object has a given property
  - For example:
    - let dog = { “name”: “Max” }
    - “name” in dog // this result in the boolean value true

# Expressions - Equality operators

- **==**
  - Equality operator
- **!=**
  - Inequality operator
- **===**
  - Strict equality operator
- **!==**
  - String inequality operator

# Expressions - Binary logical operators

- &&
  - Logical AND. Operates normally with boolean values. It will only return the boolean value true when both of the operated values are the boolean value true
  - For example:
    - true && true = true
    - false && true = false
    - true && false = false
    - false && false = false
- | |
  - Logical OR. Operates normally with boolean values. It will return the boolean value true when either of the operated values are the boolean true
  - For example:
    - true || true = true
    - false || true = true
    - true || false = true
    - false || false = false
- ??
  - Nullish coalescing operator. When the left side value is null, the right side value is returned
  - For example:
    - let a = null ?? “Greetings” // results in a being assigned the value “Greetings”
    - let b = “Hello” ?? “default value” // results in b being assigned the value “Hello”

# Expressions - Assignment operators

-