JS Datatypes

Datatypes

- Javascript Data Types

- There is a handful of basic building blocks, known as primitive data types available in JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures).

- Use typeof to read the datatype.

Boolean

- Is a binary data type which can have only two possible values- true or false.

- Used typically in control structures for comparison.

- All values which convert and will work like false:
    - undefined
    - null
    - 0
    - -0
    - NaN
    - ""

- All other values, including all objects (and arrays) convert to, and work like, true.

- Truthy/Falsy

Null

- This type has only one possible value – null
- It means that there is no value
- If this type is used in boolean operations it is intrepret as falsy
- See also typeof null -> object.

Undefined

- Any variable, which has not been assigned a value is undefined.

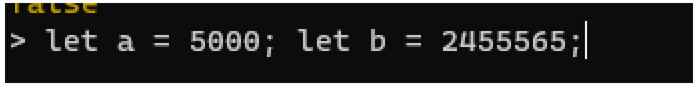
- The equality operator == considers them to be equal. (Use the strict equality operator === to distinguish them.)

Number

- used to represent integers and to approximate real numbers
- Internal representation uses the 64-bit floating-point format defined by the IEEE 754 standard
  - ±1.7976931348623157 × 10^308 and as small as ±5 × 10^−324
  - integers between −9,007,199,254,740,992 and 9,007,199,254,740,992
    - Bigger integers will loose precision
    - 52 bits available for integer numbers
- Numeric literal
  - Number which appears directly in JS code

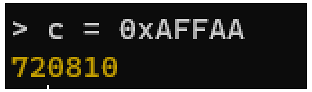
Integer Numbers– different bases

- Base 10



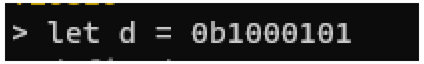
```
false
> let a = 5000; let b = 2455565;
```

- Hexadecimal (base-16)



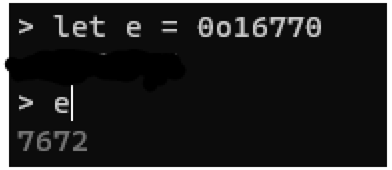
```
> c = 0xAFFAA
720810
```

- Binary (base-2)



```
> let d = 0b1000101
```

- Octal (base-8)



```
> let e = 0o16770
[redacted]
> e
7672
```

See Number.MAX_SAFE_INTEGER

Try:

```
const value1 = 9007199254740991 + 1
const value2 = 9007199254740991 + 2
```

```
Compare
value1 === value2
```

Floating Point Numbers

- Floating-point literals can have a decimal point

- Floating-point literals may also be represented using exponential notation
  - a real number followed by the letter e (or E), followed by an optional plus or minus sign, followed by an integer exponent.
  - This notation represents the real number multiplied by 10 to the power of the exponent.

```
3.14
2345.6789
.3333333333333333333
6.02e23 // 6.02 × 10²³
1.4738223E-32 // 1.4738223 × 10⁻³²
```

Floating point number exercise

- Write a program, which declares two variables, x and y.
- X is initialized with calculation operation 0.4 - 0.3
- Y is initialized with calculation operation 0.5 - 0.4
- Print the values of X and Y
- Compare if the values of X and Y are equal and print the result

BigInt

- a numeric type whose values are integers
- allow the representation of 64-bit integers (remember the Number only has 52 bits)
- Notice that BigInt cannot be used with methods in the built-in Math object and cannot be mixed with instances of Number in operations
- Standard operators work with BigInt

BigInt

- BigInt literals are written as a string of digits followed by a lowercase letter n
  - Default is base 10, but you can use the `0b`, `0o`, and `0x` prefixes for binary, octal, and hexadecimal BigInts

- `1234n                // A not-so-big BigInt literal`
- `0b111111n            // A binary BigInt`
- `0o7777n              // An octal BigInt`
- `0x8000000000000000n  // => 2n**63n: A 64-bit integer`

String

- A string is an immutable ordered sequence of 16-bit values, each of which typically represents a Unicode character
- *length* of a string is the number of 16-bit values it contains
- Zero based indexing, same as array
- Empty string has length of zero

String Literal

- Double quote literals
  - "Hello World"
- Single quote literals
  - 'Hello World'
- Backtick quote literals
  - `Hello World`
  - Template literals

- Multiple lines by backslash \ (double and single quote) or directly on multiple lines in backtick
- Backslash also for escape character

String exercises

1. **Concatenation:** Create two variables, `firstName` and `lastName`, and concatenate them to form a full name. Then, display the full name.
2. **String Length:** Find and display the length of a string "Hello World"
3. **Name printout:** Declare your name as a string in one variable called *name* and then create another variable, which uses the value of the *name* variable to print "My name is Max", where Max is your name from the *name* variable.
4. **Substring:** Use string "Javascript is fun" and use a suitable string method to extract word "is" from it.
5. **Uppercase and Lowercase:** Convert a string to uppercase and lowercase. Use "MiXeD CaSe" as the initial string to convert.
6. **String Replace:** Replace a specific word in a sentence with another word with a suitable method. Use "I love JavaScript!" As initial string and replace word "JavaScript" with word "coding".

Type Conversions

- JS is quite loose with mixing types.
- Consider for example code:
  - 44 + " is my shoe size"
  - "10" * "2"
- Conversions and equality
  - == equals
  - === strict equals