WEB230: JavaScript 1

Module 1A: Values, Types, and Operators

Values

- Any small bit of data
- Each value has a type

JavaScript has 6 types of values:

- numbers
- strings
- booleans
- objects
- functions
- undefined

Numbers

- only one kind of number

```
13
9.81
2.998e8
```

Arithmetic Operators

- + Addition
- - Subtraction
- * Multiplication
- / Division
- % Modulus (remainder)
- ** Exponent (New)

Arithmetic

- JS has arithmetic operators

```
100 + 4 * 11
```

Special Numbers

- 3 special values considered numbers
- don't behave like numbers - Don't trust these too much:
  - Infinity
  - -Infinity
- If the operation results are not meaningful:
  - NaN - not a number

Strings

- Represent text
- Zero or more characters stored as a single value

```
"Mary's car is red."
'The monkey says "goodbye"'
`Back ticks are called "template literals"`
```

- single or double quotes behave very much the same
  — only difference is in which type of quote you need to escape

Strings Escaping

- some special characters need a backslash
  - newline is "\n", tab is "\t"

```
"This is the first line\nAnd this is the second"
```

will result in:

```
This is the first line
And this is the second
```

Strings Escaping Continued

- if you need to display a special character use "\"

```
"A newline character is written like \"\\n\"."
```

will result in:

```
A newline character is written like "\n".
```

String Operator

- There is only one:
  - `+` Concatenation - Join two strings together

```
"Patch my boat " + "with chewing gum"
```

will result in:

```
"Patch my boat with chewing gum"
```

Template Literals

- Backtick-quoted strings, usually called *template literals*, can do more than single or double quoted strings:
  - span lines
  - embed other values

```
`Strings can
now span
lines`
```

Template Literals Continued

- an expression inside ${} will be evaluated, converted to a string, and included at that position

```
let number = 100;
console.log(`half of ${number} is ${number / 2}`);
```

displays:

```
half of 100 is 50
```

Unary Operators

- operate on a single value
- Some operators are words:
  - typeof - produces a string naming the type
- Others:
  - - negate (number)
  - ! not (bolean)

Boolean Values

- has just two values
- true or false

Comparison

- > and < result in boolean values

```
5 > 2        // true
"abc" > "def"  // false
```

- >= Greater than or equal
- <= Less than or equal
- == Equal
- != Not Equal

Logical Operators

- && AND
- || OR
- ! NOT

Ternary Operator

- takes 3 values

```
true ? 1 : 2  // 1
false ? 1 : 2  // 2
```

Empty Values

- The absense of value
- null
- undefined
- If something does not produce a meaningful result it will produce undefined
- null is an accident of JS design

Automatic Type Conversion

- JavaScript will do it's best to work with what you give it.
- Sometimes it has to convert from one type to another
- called type coercion

```
"one" + 2  // "one2"
"5" * 2    // 10
```

Truthy and Falsy

- If a boolean value is expected
- 0, "", undefined, null, NaN are false
- anything else is true

Precise Compare

- Sometimes we want to make sure they are the same type too!
- === precisely equal (value and type)
- !== precisely not equal
- It is recommended to use these instead of == and !=

```
"2" == 2   // true
"2" === 2  // false
```

Short-circuiting of logical operators

- logical operators && and ||
- The second value is only evaluated if needed

```
true || console.log("Hello")
```

```
true && console.log("Hello")
```