# JavaScript Specials

# Code structure

- Statements are delimited with a semicolon
- Usually, a line-break is also treated as a delimiter, so that would also work
  - That’s called “automatic semicolon insertion”. Sometimes it doesn’t work
  - Most codestyle guides agree that we should put a semicolon after each statement.
- Semicolons are not required after code blocks {...} and syntax constructs with them like loops

# Strict mode

- fully enable all features of modern JavaScript, we should start scripts with `"use strict"`.
- The directive must be at the top of a script or at the beginning of a function body.
- Without "use strict", everything still works, but some features behave in the old-fashion, “compatible” way.
  - We’d generally prefer the modern behavior.

# Variables

- Can be declared using:
  - `let`
  - `const` (constant, value can't be changed)
  - `var` (old-style, avoid)
- A variable name can include
  - Letters and digits, but the first character may not be a digit.
  - Characters `$` and `_` are normal, on par with letters.
  - Non-Latin alphabets and hieroglyphs are also allowed, but commonly not used.
- Variables are dynamically typed. They can store any value.

# Data Types

- There are 8 data types:
  - `number` for both floating-point and integer numbers,
  - `bigint` for integer numbers of arbitrary length,
  - `string` for strings,
  - `boolean` for logical values: `true`/`false`,
  - `null` – a type with a single value `null`, meaning “empty” or “does not exist”,
  - `undefined` – a type with a single value `undefined`, meaning “not assigned”,
  - `object` and `symbol` – for complex data structures and unique identifiers, we haven’t learnt them yet.

# The `typeof` operator

- The `typeof` operator returns the type for a value, with two exceptions

```
typeof null == "object" // error in the language

typeof function(){} == "function" // functions are treated specially
```

# Interactions

- In browser environment we have inbuild basic UI functions like `prompt(message,[default])`, `confirm(message)` and `alert(message)`
- For Node.js environment we use external module prompt-sync to get similar console-based input functionality and `console.log(message)` for printing out on the console.

# Operators

- JavaScript supports the following operators:
  - Arithmetical
    - Regular: `*` `+` `-` `/`, also `%` for the remainder and `**` for power of a number.
    - The binary plus + concatenates strings if either or both operands are strings by converting the non string operand into a string first.
  - Assignment
    - There is a simple assignment: `a = b` and combined ones like `a *= 2`.
  - Relational operators (comparisons)
    - Equality check `==` for values of different types converts them to a number (except `null` and `undefined` that equal each other and nothing else)
    - Values `null` and `undefined` are special: they equal `==` each other and don’t equal anything else.
    - The strict equality operator === doesn’t do the conversion: different types always mean different values for it.
    - Other comparisons covert to a number as well.
    - Greater/less comparisons compare strings character-by-character, other types are converted to a number.

# Operators cont.

- More operators
  - Logical operators
    - Logical AND `&&` and OR `||` perform short-circuit evaluation and then return the value where it stopped (not necessarily `true`/`false`).
    - Logical NOT `!` converts the operand to boolean type and returns the inverse value.

# Loops

- While
  - do while – executes at least once
  - Sentinel and counter variants
- The variable declared in `for(let...)` loop is visible only inside the loop.
  - But we can also omit let and reuse an existing variable.
- Directives `break`/`continue` allow to exit the whole loop/current iteration.

# The “switch” construct

- The “switch” construct can replace multiple `if` checks. It uses `===` (strict equality) for comparisons.

```
let age = prompt('Your age?', 18);

switch (age) {
  case 18:
    alert("Won't work"); // the result of prompt is a string, not a number
    break;

  case "18":
    alert("This works!");
    break;
}
```

# Functions

- We covered three ways to create a function in JavaScript:
  - Function Declaration: the function in the main code flow
  - Function Expression: the function in the context of an expression
  - Arrow functions
- Functions may have local variables: those declared inside its body. Such variables are only visible inside the function.
- Parameters can have default values:
  - `function sum(a = 1, b = 2) {...}`.
- Functions always return something. If there's no `return` statement, then the result is `undefined`.

# References

- JavaScript specials