Javascript

- Vysoký skriptovací jazyk, interpretovaný (nepotřebuje kompilaci)
- Široké využití (například pomocí frameworků):
    - Webové aplikace na straně klienta (nejčastější)
        - Manipulace s HTML
        - Manipulace s CSS
        - Grafika
        - Validace vstupů
        - Matematické operace
    - Webové aplikace na straně serveru
    - Práce s databázemi
    - Desktopové aplikace
    - Mobilní aplikace

Proměnné

- Jsou dynamické – nemusí se deklarovat, interpreter automaticky převádí jejich typ
- *var* – deklarace proměnné
- Typy:
    - Číslo – celá čísla, desetinná čísla, vědecký zápis čísel
    - Boolean – true/false
    - String – textové řetězce
- Rozsah – odkud můžeme k proměnné přistupovat
    - Globální – všechny proměnné deklarované mimo funkce; můžeme k nim přistupovat odkudkoli z programu (vně i uvnitř funkcí)
    - Lokální – proměnné deklarované uvnitř funkcí; můžeme k ní přistupovat pouze zevnitř funkce (pokud je funkce ve funkci, pak tuto proměnnou může využívat také)

Pole

- Pole proměnných s proměnnou délkou
- Proměnné v poli mohou být různých typů
- Zápis: *var arr = [“foo“, “bar“, 1];*

Funkce

- Definice pomocí *function Name()*
- Funkce je objekt → má atributy a může být předána

Anonymní funkce

- Funkce uvnitř funkcí
- Jsou vytvořeny při zavolání funkce nadřazené
- Mají přístup k proměnným funkce vnější

Objekty

- Objekt je soubor proměnných a metod
- Základ pro OOP
- Zápis pomocí složených závorek

- Každá hodnota má klíč, název, kterým se identifikuje
- Podobají se slovníkům v Pythonu
- Téměř vše v JavaScriptu je objekt (pole, funkce, …)
- Kontext – objekt, do kterého funkce patří; pojí se s klíčovým slovem *this*

Konstruktor

- V jiných jazycích se definuje třída a podle ní se vytváří jednotlivé objekty
- JavaScript neobsahuje definici tříd
- Dá se obejít pomocí funkcí

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
}
```

- Tento postup nemá vlastnosti klasických tříd jako třeba dědičnost
- Konstruktor se po vytvoření nedá měnit

Prototyp

- Každý objekt má prototyp
- Objekty dědí proměnné a metody z prototypu
- Prototyp umožňuje přidávat nové metody a proměnné do konstruktoru

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.nationality = "English";
```

- Objektu můžeme přiřadit prototyp jiného objektu a tím umožnit dědičnost

HTML DOM

- HTML Document Object Model
- Je to rozhraní pro práci s HTML
- Slouží k přístupu a měnění HTML elementů
- Z HTML elementů vytvoří stromovou strukturu
- Díky tomu můžeme:
  - Měnit HTML elementy
  - Měnit atributy elementů
  - Měnit CSS
  - Mazat elementy
  - Přidávat elementy a atributy
  - Reagovat na existující elementy