TypeScript

Odkaz na učebnici

bit.ly/hacknitypescript

Playlist všech videí na kanálu Hackni svou budoucnost

https://www.youtube.com/playlist?list=PLQ8x_VWW6AkvlMSxICuexWjk_vwfWbqG1

David Šetek

1. Co je to typescript

https://youtu.be/mTt_D5idYWM

Odkaz na prezentaci:
https://docs.google.com/presentation/d/1K33oUIwmphXomSUEkIxq04TrMXcpzsqy/edit?usp=sharing&ouid=101906621946234510159&rtpof=true&sd=true

Z konzole

```
function soucet(cislo1, cislo2){
    return cislo1 + cislo2
}
```

undefined

soucet(5,2)

7

const vysledek = soucet(5,4)

undefined

vysledek

9

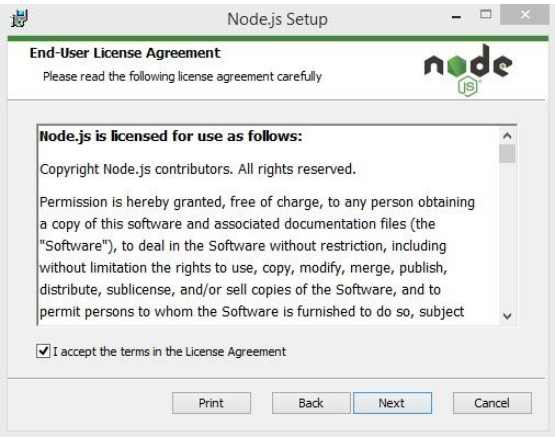
soucet("5",4)

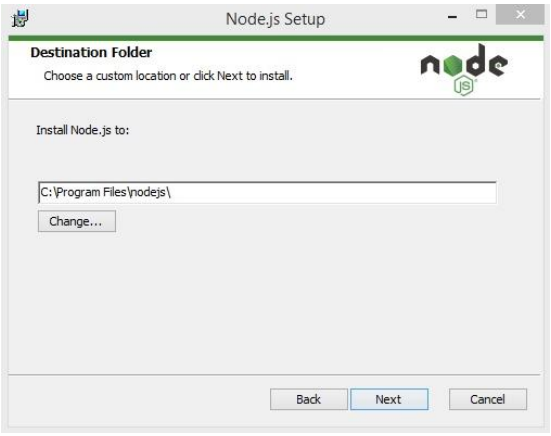
"54"

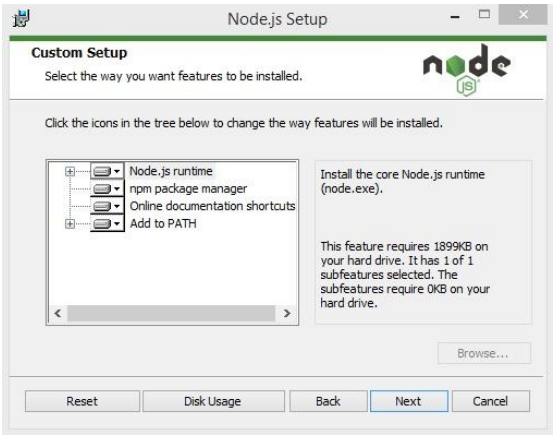
2. Učebnice a zprovoznění typescriptu ve VS code

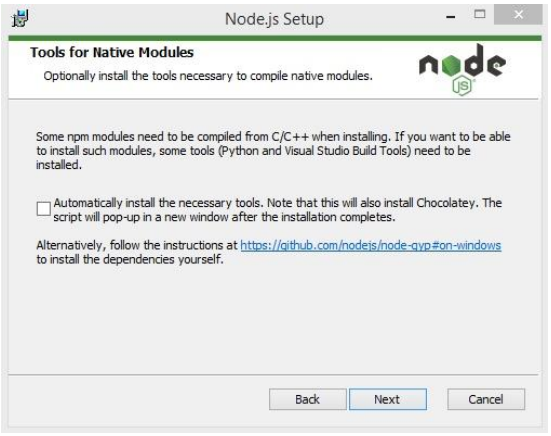
https://youtu.be/CSaNrfA5pF4

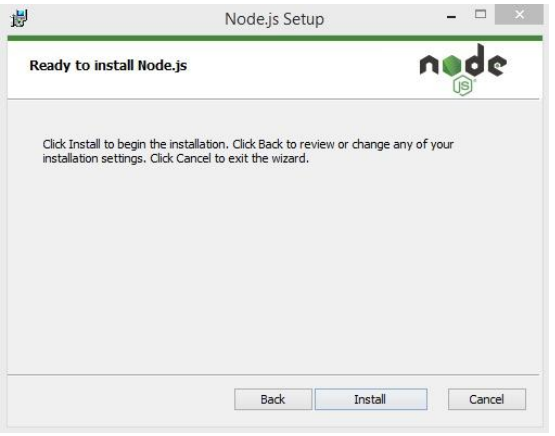
https://nodejs.org/en/
https://www.typescriptlang.org/download

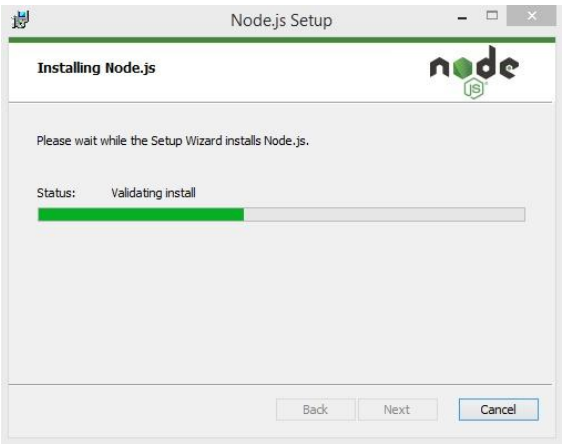


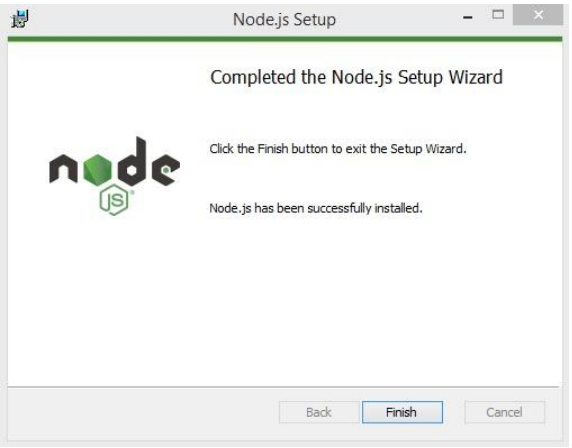












Příkazy z videa

node -v
(mělo by vyhodit číslo = verzi node)

npm install typescript --save-dev
NEBO
sudo npm install typescript --save-dev

Pokud výše dva uvedené příkazy nefungují, tak zkuste ještě
npm install -g typescript
NEBO
sudo npm install -g typescript

tsc script.ts
(mělo by zkompilovat = převést script.ty na script.js - tedy typescript na javascript)

tsc -w script.ts
(bude kompilovat při jakékoliv změně souboru script.ts

3. Ukázka typescriptu v praxi

https://youtu.be/M0DjxaQEiFE

```
function soucet(num1: number, num2: number){
    return num1 + num2
}

const vysledek = soucet(5, 6)
console.log(vysledek)
```

4. Základní datové typy (number, string, boolean)

https://youtu.be/SMjc982g0w4

```
// jednořádkový komentář

/*
Víceřádkový
komentář
*/


/*
    number = 3, 2.1, -5
    string = "pes", 'pes'
    boolean = true, false (pravda, nepravda)
*/

const myName: string = "David"
const age: number = 34
const adult: boolean = true
```

5. Procvičování základních datových typů - realitní kancelář

https://youtu.be/oQXRkGLk_kY

Zadání

```
/* Realitní kancelář
Popisujeme dům. Dům má 4 okna, 1 dveře, barva je bílá,
výšku má 50 metrů, je nový, má garáž.

Vaším úkolem je tyto hodnoty zanést do následujícím
proměnných a napsat k nim vždy správný datový typ:

windows
doors
color
height
isNew (ano, ne)
garage (ano, ne)

Vypište proměnné do konzole

*/
```

Řešení

```
const windows: number = 4
const doors: number = 1
const color: string = "bílá"
const height: number = 50
const isNew: boolean = true
const garage: boolean = true

console.log(windows)
```

```
console.log(doors)
console.log(color)
console.log(height)
console.log(isNew)
console.log(garage)
```

6. Odvození typu typescriptem (type inference) + malé procvičování

https://youtu.be/KxLAWAlPbg0

```
// Type inference = odvození typu

let windows = 4

/* Vylepšení Realitní kanceláře
const windows: number = 4
const doors: number = 1
const color: string = "bílá"
const height: number = 50
const isNew: boolean = true
const garage: boolean = true
*/
```

Řešení

```
/* Vylepšení Realitní kanceláře */
const windows = 4
const doors = 1
const color = "bílá"
const height = 50
const isNew = true
const garage = true
```

7. Datový typ object v typescriptu a jak na něj

https://youtu.be/XPcRvByPP_s

Klasický objekt

```
const person = {
    name: "David",
    age: 15
}
```

Můžeme specifikovat, že je to objekt, ale přestane fungovat person.name

```
const person: object = {
    name: "David",
    age: 15
}

console.log(person.name)
```

Museli bychom specifikovat takto:

```
const person: {
    name: string,
    age: number
} = {
    name: "David",
    age: 15
}

console.log(person.name)
console.log(person.age)
```

V praxi zpravidla píšeme stále takto, protože zápis výše je moc zdlouhavý a kód má zbytečně moc řádků

```
const person = {
    name: "David",
    age: 15,
```

```
    adult: false
}

console.log(person.name)
console.log(person.age)
console.log(person.adult)
```

8. Procvičování objektů - dům v realitní kanceláři jako objekt

https://youtu.be/s1Uh1LF5czk

Zadání

```
// const windows = 4
// const doors = 1
// const color = "bílá"
// const height = 50
// const isNew = true
// const garage = true

/* Objekt v realitní kanceláři
Vytvořte objekt s názvem house, který bude obsahovat
všechny výše uvedené proměnné. Použijte pro zápis co
nejjednodušší způsob (nemusíte specifikovat, jaká
proměnná je jaký typ)

Vypište všechny vlastnosti do konzole
*/
```

Řešení

```
const house = {
    windows: 4,
    doors: 1,
    color: "bílá",
    height: 50,
    isNew: true,
    garage: true
}

console.log(house.windows)
```

```
console.log(house.doors)
console.log(house.color)
console.log(house.height)
console.log(house.isNew)
console.log(house.garage)
```

9. Pole v typescriptu

https://youtu.be/U_u3HthPNsM

```
// Pole (array)

let employees: any[]

employees = ["David", "Diana", "Harry", 4, true]


console.log(employees[0])
console.log(employees[1])
console.log(employees[2])

for (const oneEmployee of employees){
    console.log(oneEmployee)
}


const person = {
    firstName: "David",
    secondName: "Šetek",
    age: 34,
    hobbies: ["sport", "teaching", "programming"]
}

console.log(person.hobbies[0])
console.log(person.hobbies[1])
console.log(person.hobbies[2])
```

```
for (const oneHobby of person.hobbies){
    console.log(oneHobby)
}
```

10. Procvičování - pole klientů v realitní kanceláři

https://youtu.be/RB_OKrttPCc

```
/* Realitní kancelář a pole klientů
Do následujícího objektu:
const house = {
    windows: 4,
    doors: 1,
    color: "bílá",
    height: 50,
    isNew: true,
    garage: true
}

máte za úkol přidat dvě pole.

První bude mít název vipClients a zde budou uloženy
firmy, které mají o dům zájem. Firmy budou
Build-store, Damage-do, Bum-bum-company.

Druhé pole budou zájemci z řad běžných lidí. Zájemci
jsou Daniel Stavitel, Petr Oknokrad a Dana Parketová.

Dále máte do konzole vypsat vždy prvního a posledního
z obou dvou polí.

*/
```

Řešení

```
const house = {
    windows: 4,
    doors: 1,
    color: "bílá",
    height: 50,
    isNew: true,
    garage: true,
    vipClients: ["Build-store", "Damage-do", "Bum-bum-company"],
    clients: ["Daniel Stavitel", "Petr Oknokrad", "Dana Parketová"]
}

console.log(house.vipClients[0])
console.log(house.vipClients[2])
console.log(house.clients[0])
console.log(house.clients[2])
```

11. Tuples

https://youtu.be/DojCwL4W8g0

```
// Tuples - pole s fixní délkou

const array: string[] = ["David", "Harry", "Ron"]
const tuples: [string, number] = ["David", 5]

const employee: {
    name: string,
    age: number,
    hobbies: string[],
    department: [number, string]
} = {
    name: "David Šetek",
    age: 34,
    hobbies: ["sport", "teaching", "programming"],
    department: [2, "it developer"]
}

// výpis pomocí console.log
console.log(employee.department[0])
console.log(employee.department[1])

// výpis pomocí cyklu
for (const x of employee.department){
    console.log(x)
}
```

12. Procvičování - Tuple v realitní kanceláři

https://youtu.be/L8PqHQaUnFg

```
/* Oblast domu v realitní kanceláři

Máte realitní kancelář ve městě, které se dělí do pěti částí. Tyto části jsou očíslovány (1 až 5). Každá část se dělí ještě na sever, jih, západ, východ a střed. Přidejte do objektu house tuple s názvem location, které má vždy dva údaje - Číslo části a určenou oblast. Tento konkrétní dům bude v 5. části a v jižní oblasti.

const house = {
    windows: 4
}

Vypište tuple do konzole pomocí console.log a for cyklu
*/

const house: {
    windows: number,
    location: [number, string]
} = {
    windows: 4,
    location: [5, "jih"]
}

// výpis pomocí console.log
console.log(house.location[0])
```

```
console.log(house.location[1])

// výpis pomocí cyklu
for(const i of house.location){
    console.log(i)
}
```

13. Enum

https://youtu.be/r4zLHuzFGI0

```
// Enum

enum Role {ADMIN = "admin", AUTHOR = 200, READ_ONLY =
300}

const employee = {
    name: "David Šetek",
    age: 34,
    role: Role.ADMIN
}

console.log(employee.role)

/*
Role:
    0 - admin
    1 - author
    2 - read_only
*/
```

14. Procvičování enum - stav domu v realitní kanceláři

https://youtu.be/ZswFUwECwHE

```
/* Realitní kancelář a stav domu

Máte následující objekt:

const house = {
    windows: 4
}

Mimo objekt vytvoříte enum s názvem House_condition
(stav domu) a naplníte ho třemi hodnotami:
habitable (obyvatelný),
normal (normální),
uninhabitable (neobyvatelný).

Do objektu house přidáte vlastnost condition a
přiřadíte z enum, že je neobyvatelný.

Vypište tuto vlastnost do konzole pomocí console.log -
do konzole se vypíše jen číslo

*/

enum House_condition {HABITABLE, NORMAL,
UNINHABITABLE}

const house = {
    windows: 4,
```

```
    condition: House_condition.UNINHABITABLE
}

console.log(house.condition)
```

15. Any

https://youtu.be/Xf3GCHhqYdM

```
const myArray: any[] = [true, 5, "David"]
```

16. Union type

https://youtu.be/m0cm6WIP5Dg

```
// Union type

function combination(input1: string | number, input2:
string | number){
    let result: number | string
    if(typeof(input1) === "number" && typeof(input2) === "number"){
        result = input1 + input2
        return result
    } else {
        result = input1.toString() + input2.toString()
        return result
    }
}

console.log(combination(5, 6))

console.log(combination("David", "Šetek"))
```

17. Literal type

https://youtu.be/P90vWFXDPVE

```
// Literal type

function combination(input1: string | number, input2:
string | number, resultType: "as-number" | "as-text"){
    let result: number | string
    if(typeof(input1) === "number" && typeof(input2)
=== "number" || resultType === "as-number"){
        result = +input1 + +input2
        return result
    } else {
        result = input1.toString() + input2.toString()
        return result
    }

    // if(resultType === "as-number"){
    //     result = +result
    //     return result
    // } else {
    //     return result.toString()
    // }
}

console.log(combination(5, 6, "as-number"))

console.log(combination("David", "Šetek", "as-text"))

console.log(combination("10","8","as-number"))
```

```
console.log(combination("10", 3, "as-number"))
```

18. Type alias neboli custom type

https://youtu.be/3tLYjPjCEEQ

V bílé části jsme nic neměnili

```
// Type alias / custom type
type Combination = string | number
type resultType = "as-number" | "as-text"

function combination(input1: Combination, input2:
Combination, resultType: resultType){
    let result: Combination
    if(typeof(input1) === "number" && typeof(input2)
=== "number" || resultType === "as-number"){
        result = +input1 + +input2
        return result
    } else {
        result = input1.toString() + input2.toString()
        return result
    }
}

console.log(combination(5, 6, "as-number"))

console.log(combination("David", "Šetek", "as-text"))

console.log(combination("10","8","as-number"))

console.log(combination("10", 3, "as-number"))
```

19. Return a void

https://youtu.be/K7jsz1r0NrA

```
// Return a void

function sum(n1: number, n2: number): number {
    return n1 + n2
}

function sum2(n1: number, n2: number): string {
    return n1.toString() + n2.toString()
}

function writeResult(num: number){
    console.log("Result: " + num)
    // console.log(`Result: ${num}`)
}

writeResult(30)
```

20. Procvičování funkcí a výpisu

https://youtu.be/EN36Y0N5I30

```
/*

Vezměte si následující objekt, který jsme již dříve vytvořili:

const house = {
    windows: 4,
    doors: 1,
    color: "bílá",
    height: 50
}

Vaším úkolem je sestrojit funkci, do které když pošlu tento objekt, tak se vypíše věta:
"Tento dům má 4 okna. Počet dveří je 1. Barva domu je bílá. Výška domu je 50 metrů."

Napovím, že obecný předpis funkce bude mít jeden parametr, který si jakýmkoliv způsobem pojmenujete. Až budete funkci volat, tak tam jen za tento parametr dosadíte objekt house.

Pro vypsání věty můžeme z předchozího videa použít buď znaménko plus (+) nebo tzv. template string ${něco}. Výběr je na vás.

*/
```

Řešení

```
const house = {
    windows: 4,
    doors: 1,
    color: "bílá",
    height: 50
}

function houseDescription(myObject){
    console.log(`Tento dům má ${myObject.windows}
okna. Počet dveří je ${myObject.doors}. Barva domu je
${myObject.color}. Výška domu je ${myObject.height}
metrů.`)
}

houseDescription(house)
```

21. Funkce jako typ

https://youtu.be/gA5OjmKBu4U

```
// Funkce jako typ

function sum(num1: number, num2: number) {
    return num1 + num2
}

function test(description: string){
    return description
}

let myNumber = 5
let myString = "David"

// let myFunction: Function
let myFunction: (x: number, y: number) => number
myFunction = sum
// myFunction = test

console.log(myFunction(10, 40))
```

22. Procvičování funkce jako typ

https://youtu.be/Eeu4IzRFiQk

```
// Funkce jako typ - procvičování

function test1(num1: number, num2: number, num3:
number) {
    return num1 + num2 + num3
}

// let myFunction1:


function test2(description: string){
    return description
}

// let myFunction2:



function test3(myString: string, myNumber: number){
    return myString
}

// let myFunction3:
```

Řešení:

```
// Funkce jako typ - procvičování

```

```
function test1(num1: number, num2: number, num3:
number) {
    return num1 + num2 + num3
}

let myFunction1: (a: number, b: number, c: number) =>
number
myFunction1 = test1



function test2(description: string){
    return description
}

let myFunction2: (x: string) => string
myFunction2 = test2



function test3(myString: string, myNumber: number){
    return myString
}

let myFunction3: (m: string, n: number) => string
myFunction3 = test3
```

23. Callback function v TypeScriptu

https://youtu.be/z7EdHPGutfM

```
// Callback funkce - funkce jako parametr jiné funkce

const y = (nejakeCislo: number) =>
console.log(nejakeCislo)
y(10)
y(60)


function sum(n1: number, n2: number, callBackFun: (n3:
number) => void){
    const result = n1 + n2
    callBackFun(result)
}

sum(5, 30, (x) => console.log(x))
```

24. Unknown type v TypeScriptu

https://youtu.be/OuQSeL9xaMc

```
// Unknown type

let test: unknown
let result: string

test = 5
test = "David"

if (typeof test === "string"){
    result = test
}
```

25. Never type v TypeScriptu

https://youtu.be/wngVEqFPW28

```
// Never type

function test(myValue: string){
    return myValue
}

function generateError(errorText: string, errorNumber:
number): never {
    throw {message: errorText, errorCode: errorNumber}
}

generateError("Závažná chyba", 688)
```

26. Compiler a compilování více souborů najednou

https://youtu.be/ZEV1KVU-1uY

1. založte si druhý soubor např. test.ts
2. napojite soubor test.js do index.html
3. v terminálu dejte příkaz tsc --init (vytvoří se soubor tsconfig.json)
4. nyní můžete použít pro kompilaci více souborů příkaz **tsc** v terminálu
5. pokud chcete automatické ukládání, tak použijte příkaz **tsc -w**

27. tsconfig.json - exclude, include, files

https://youtu.be/ZzMb494hEwM

tsconfig.json

```
,
  "exclude": [
    "test.ts",
    "*.dev.ts", //wild-card
    "node_modules"
  ],
  "include": [
    "script.ts"
  ],
  "files": [
    "script.ts"
  ]
```

28. tsconfig.json - target, knihovny (lib)

https://youtu.be/XmLvJlpa6XY

index.html

```
<div class="my-test">Toto je jen test</div>
```

script.ts

```
const myTest = document.querySelector(".my-test")
```

tsconfig.json

target: es5 (Ctrl + mezerník)

```
"target": "ES6",
```

tsconfig.json

```
"lib": [
      "dom",
      "es6",
      "dom.iterable",
      "scripthost"
    ],
```

29. tsconfig.json - sourceMap a typescriptové soubory v developerské consoli

https://youtu.be/msNsy7EBFC4

```
"sourceMap": true,
```

30. STŘIH U HTML tsconfig.json - outDir a rootDir (děláme pořádek v souborech)

https://youtu.be/q3Ynl19Oa-E

tsconfig.json

```
"outDir": "./dist",
"rootDir": "./src",
```

index.html

```
<script src="dist/script.js"></script>
<script src="dist/test.js"></script>
```

31. tsconfig.json - removeComments a noEmitOnError

https://youtu.be/11HyKP_fYQM

```
"removeComments": true,
```

```
"noEmitOnError": true,
```

32. tsconfig.json - Strict mode

https://youtu.be/NZq9MTYGDI4

```
 "strict": true,
```

33. tsconfig.json - Additional Check (dodatečná kontrola)

https://youtu.be/cD5RIa5j0x8

```
// noUnusedLocals
// if(5 > 3){
//     let result = "Je to tak"
// }


// noUnusedParameters
// function sum(n1: number, n2: number){
//     console.log(n1)
// }


// noImplicitReturns
// function sum2(n1: number, n2: number){
//     if(n1 + n2 >= 0){
//         return n1 + n2
//     }
//     return "Došlo k chybě"
// }
```

34. Nový javascript a typescript - const, let, var

https://youtu.be/_FzLiKuoamc

```
// const, let, var

// const userName = "David"
// userName = "Harry"

let userName = "David"
userName = "Harry"

var userName2 = "David"
userName = "Harry"


function sum(n1: number, n2: number){
    var result
    result = n1 + n2
    return result
}

// console.log(result)

if(5 > 3){
    let result2 = true
}

// console.log(result2)
```

35. Nový javascript a typescript - arrow function

https://youtu.be/RuCSciK0_uk

```
// Arrow function
function sum(n1: number, n2: number){
    return n1 + n2
}
const result = sum(5, 8)
console.log(result)


const mySum = (n1: number, n2: number) => n1 + n2
console.log(mySum(5, 10))
```

36. Nový javascript a typescript - default parameters

https://youtu.be/LpgtWAmwiGw

```
// Default parameters

const mySum = (n1: number, n2: number = 3) => n1 + n2
console.log(mySum(5))
```

37. Nový javascript a typescript - spread operator

https://youtu.be/u7d4NvzyX2U

```
// Spread operator

const hobbies = ["teaching", "reading", "cinema"]
const activeHobbies = ["running"]

// activeHobbies.push(hobbies)
// activeHobbies.push(hobbies[0], hobbies[1],
hobbies[2])
activeHobbies.push(...hobbies)

console.log(activeHobbies)
```

38. Nový javascript a typescript - rest parameters

https://youtu.be/JRuyIwNjspw

```
// Rest parameters

const sum = (...myNumber: number[]) => {
    return myNumber.reduce((result, value)=>{
        return result + value
    })
}

console.log(sum(5, 3, 5))
console.log(sum(10, 3, 4, 2, 4.2, 10))
```

39. Nový javascript a typescript - array destructuring a object destructuring

https://youtu.be/lCU7Xjh0trY

```
// Array destructuring
const employees: string[] = ["David", "Harry",
"Hermiona", "Ron"]

const [employee1, employee2, ...otherEmployees] =
employees

console.log(employee1)
console.log(employee2)
console.log(otherEmployees)
console.log(employees)

// Object destructuring
const person = {
    firstName: "David",
    age: 34,
    hobby: "teaching"
}

const {firstName, age} = person

console.log(firstName)
console.log(age)
```

40. Nový javascript a typescript - závěrem k této části

https://youtu.be/hRuk9-N0ZLA

tsconfig.json

```
"target": "es6",
```

tsconfig.json

```
"target": "es5",
```

41. Objektově orientované programování (OOP) - class, constructor, objects

https://youtu.be/DqRdFFxV5Do

```
// Objektově orientované programování (OOP)

class Department {
    name: string
    number: number

    constructor(na: string, nu: number){
        this.name = na
        this.number = nu
    }
}

const HRdep = new Department("Human resources", 100)
const MAdep = new Department("Marketing", 200)
const FIdep = new Department("Finance", 300)

console.log(HRdep)
console.log(MAdep)
console.log(FIdep)

console.log(HRdep.name)
console.log(MAdep.name)
console.log(FIdep.name)

console.log(HRdep.number)
console.log(MAdep.number)
console.log(FIdep.number)
```

42. Objektově orientované programování (OOP) - Procvičování class, objekty, constructor

https://youtu.be/zlZZiFyHNZc

```
// Procvičování OOP - realitní kancelář a domy

/*
Vytvořte obecný předpis (class) s názvem House. Bude mít vlastnosti street, number, floors. Tyto vlastnosti se budou zadávat již při vytvoření (constructor)

Vytvořte 3 libovolné objekty podle classy a uložte do proměnných. Následně vypište všechny tři ulice, všechna tři čísla a všechna tři podlaží do konzole.
*/
```

Řešení

```
class House {
    street: string
    number: number
    floors: number

    constructor(str: string, num: number, flo: number){
        this.street = str
        this.number = num
        this.floors = flo
    }
}

const house1 = new House("Modrá", 26, 4)
const house2 = new House("Červená", 5, 2)
```

```
const house3 = new House("Zelená", 32, 1)

console.log(house1.street)
console.log(house2.street)
console.log(house3.street)

console.log(house1.number)
console.log(house2.number)
console.log(house3.number)

console.log(house1.floors)
console.log(house2.floors)
console.log(house3.floors)
```

43. Objektově orientované programování (OOP) - metody

https://youtu.be/Cd9tsxNhnyk

```
// OOP a metody

class Department {
    depName: string
    number: number

    constructor(depN: string, nu: number){
        this.depName = depN
        this.number = nu
    }

    describe() {
        console.log("Oddělení " + this.depName + " má
číslo " + this.number)
    }
}

const HRdep = new Department("Human resources", 100)
const MAdep = new Department("Marketing", 200)
const FIdep = new Department("Finance", 300)

HRdep.describe()
MAdep.describe()
FIdep.describe()
```

44. Objektově orientované programování (OOP) - procvičování metod

https://youtu.be/7EhH9yd8JXg

```
// Procvičování OOP - realitní kancelář, domy a popis

/*

Vytvořte class House, který bude mít vlastnosti street, number, floors, state (state = stav, který bude nový nebo starý). Vytvořte se všemi vlastnosti constructor.

Do classy přidejte metodu describe, která bude vypisovat např. takovýto text: "Jedná se o nový dům. Nachází se v ulici Modrá 26. Má 4 podlaží." Části street, number, floors a state budou doplněny pomocí this a budou u výpisu každého objektu (domu) jiné.

Podle classy vytvoříte tři objekty, u kterých zavoláte metodu describe a vypíšete tři texty do konzole.

*/

class House {
    street: string
    number: number
    floors: number
    state: string
```

```
    constructor(str: string, num: number, flo: number,
sta: string){
        this.street = str
        this.number = num
        this.floors = flo
        this.state = sta
    }

    describe() {
        console.log("Jedná se o " + this.state + "
dům. Nachází se v ulici " + this.street + " " +
this.number + ". Má " + this.floors + " podlaží")
    }
}

const house1 = new House("Modrá", 26, 4, "nový")
const house2 = new House("Červená", 5, 2, "nový")
const house3 = new House("Zelená", 32, 1, "starý")

house1.describe()
house2.describe()
house3.describe()
```

45. Objektově orientované programování (OOP) - private a public

https://youtu.be/Z8MGuPKzcZM

```
// Public a private

class Department {
    depName: string
    number: number
    private employee: string[] = []

    constructor(depN: string, nu: number){
        this.depName = depN
        this.number = nu
    }

   describe() {
       console.log("Oddělení " + this.depName + " má číslo " + this.number)
   }

    addEmployee(oneEmployee: string){
        this.employee.push(oneEmployee)
    }

    printEmployee(){
        console.log(this.employee)
    }

    printAllEmployee(){
        for(const oneEmployee of this.employee){
            console.log(oneEmployee)
```

```
        }
    }

}

const HRdep = new Department("Human resources", 100)
const MAdep = new Department("Marketing", 200)
const FIdep = new Department("Finance", 300)

HRdep.addEmployee("David Šetek")
HRdep.addEmployee("Hermiona Grangerová")
HRdep.addEmployee("Harry Potter")

HRdep.printEmployee()
HRdep.printAllEmployee()

// HRdep.employee[3] = "Ron Weasley"

HRdep.printEmployee()



const HRdep = new Department("Human resources", 100)
const MAdep = new Department("Marketing", 200)
const FIdep = new Department("Finance", 300)

HRdep.addEmployee("David Šetek")
HRdep.addEmployee("Hermiona Grangerová")
HRdep.addEmployee("Harry Potter")
```

```
HRdep.printEmployee()
HRdep.printAllEmployee()

// private a public přístup
console.log(HRdep.depName = "Nový název oddělení")
console.log(HRdep.number)
// HRdep.employee[3] = "Ron Weasley"

// console.log(HRdep.employee)

console.log(HRdep.depName)
```

46. Objektově orientované programování (OOP) - zkrácený zápis v constructoru

https://youtu.be/pB80MRfOUsM

```
class Department {
    // public depName: string
    // private number: number
    private employee: string[] = []

    constructor(public depName: string, private number: number){
        // this.depName = depN
        // this.number = nu
    }
    ...
```

47. Objektově orientované programování (OOP) - readonly

https://youtu.be/TnPZyCVwHsA

```
// Readonly

class Department {
    // public readonly depName: string
    // private number: number
    private employee: string[] = []

    constructor(public readonly depName: string, private number: number){
        // this.depName = depN
        // this.number = nu
    }
```

48. Objektově orientované programování (OOP) - inheritance neboli dědění

https://youtu.be/tJxierdhYI8

```
class ITDepartment extends Department {

    constructor(number: number){
        super("IT", number)
    }
}

const frontEndDepartment = new ITDepartment(600)
frontEndDepartment.describe()
```

49. Objektově orientované programování (OOP) - inheritance neboli dědění 2. část

https://youtu.be/GE1XbW55rZU

Přidali jsme bílé části do předchozího kódu

```
class ITDepartment extends Department {

    constructor(number: number, public admins: string[]){
        super("IT", number)
        this.admins = admins
    }
}

const frontEndDepartment = new ITDepartment(600, ["David",
"Harry"])
frontEndDepartment.describe()

console.log(frontEndDepartment)
```

50. Objektově orientované programování (OOP) - přepsání metod a protected

https://youtu.be/ISWI1XDYAV4

```
class Department {
    // public readonly depName: string
    // private number: number
    protected employee: string[] = []
```

```
 addEmployee(name: string){
        if(name === "David" || name === "Harry"){
            return "Už mají přístup"
        } else {
            this.employee.push(name)
        }
    }
```

```
frontEndDepartment.addEmployee("Harry")
```

51. Objektově orientované programování (OOP) - getters, setters, throw Error

https://youtu.be/55yRXWg54jQ

Přidali jsme bílé části kódu

```
class ITDepartment extends Department {
    public mainAdmin: string

    constructor(number: number, public admins: string[]){
        super("IT", number)
        this.admins = admins
        this.mainAdmin = admins[0]
    }

    // Getter - musí mít return
    get leadAdmin(){
        if (this.mainAdmin){
            return this.mainAdmin
        }
        throw new Error("Hlavní admin nenalezen")
    }

    // Setter - musíte do něj poslat hodnotu
    set leadAdmin(value: string){
        if(this.mainAdmin){
            this.mainAdmin = value
        } else {
            throw new Error("Hlavní admin nenastaven")
        }
    }
```

```
    addEmployee(name: string){
        if(name === "David" || name === "Harry"){
            return "Už mají přístup"
        } else {
            this.employee.push(name)
        }
    }
}

const frontEndDepartment = new ITDepartment(600,
["David", "Harry"])
console.log(frontEndDepartment.leadAdmin)
frontEndDepartment.leadAdmin = "Hermiona"
console.log(frontEndDepartment.mainAdmin)
```

52. Objektově orientované programování (OOP) - statické metody a statické vlastnosti

https://youtu.be/FJ8BAFgIM50

```
console.log(Math.PI)
```

```
    static createEmployee(name: string){
        return name
    }
```

```
const department1 = new Department("Marketing", 300)
const employee1 = department1.createEmployee("Ron")
console.log(employee1)
```

```
static currentYear: number = 2021
console.log(employee1, Department.currentYear)
```

```
    describe() {
        console.log("Oddělení " + this.depName + " má
číslo " + this.number + " " + this.currentYear)
    }
```

```
    describe() {
        console.log("Oddělení " + this.depName + " má
číslo " + this.number + " " + Department.currentYear)
    }
```

53. Objektově orientované programování (OOP) - abstract class (abstraktní classa)

https://youtu.be/pFtbXKm4bWs

```
abstract class Department {
    name: string
    number: number

    constructor(na: string, num: number){
        this.name = na
        this.number = num
    }

    abstract describe(): void
}

// const marketing = new Department("Marketing", 100)
// marketing.describe()

class ITdepartment extends Department {

    constructor(depName: string, depNum: number){
        super(depName, depNum)
    }

    describe(){
        console.log("Text metody describe")
    }

}
```

```
const ITdepCzech = new ITdepartment("CzechIT", 900)
ITdepCzech.describe()
```

54. Objektově orientované programování (OOP) - Singletons a private constructor

https://youtu.be/EDjy_UIL5Uo

```
class ITdepartment extends Department {
    // Přidáme
    private static myObject: ITdepartment

    // Přidali jsme private
    private constructor(depName: string, depNum: number){
        super(depName, depNum)
    }

    describe(){
        console.log("Text metody describe")
    }

    // Přidali jsme
    static getObject(){
        if(this.myObject){
            return this.myObject
        }
        this.myObject = new ITdepartment("IT", 999)
        return this.myObject
    }
}

// const ITdepCzech = new ITdepartment("CzechIT", 900)
// ITdepCzech.describe()
```

```
// Přidali jsme
const IT = ITdepartment.getObject()
const IT2 = ITdepartment.getObject()

console.log(IT)
console.log(IT2)
```

55. Objektově orientované programování (OOP) - Interfaces

https://youtu.be/kwc_90eLZM0

```
// Interface

interface Person {
    name: string
    age: number

    greet(phrase: string): void
}

let person1: Person
person1 = {
    name: "David",
    age: 34,

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }
}

person1.greet("Ahoj, já jsem")
```

56. Objektově orientované programování (OOP) - Interface a class

https://youtu.be/iQLIvubd9EQ

```
// Interface

interface IamGreeting {
    name: string

    greet(phrase: string): void
}

class Person implements IamGreeting {
    name: string

    constructor(n: string){
        this.name = n
    }

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }

    sayHi(){
        console.log("Hi, hi, hi")
    }
}

let person2 = new Person("David")
console.log(person2)
```

```
person2.greet("Ahoj já jsem")
person2.sayHi()

// let person1: Person
// person1 = {
//     name: "David",
//     age: 34,

//     greet(myPhrase: string){
//         console.log(myPhrase + " " + this.name)
//     }
// }

// person1.greet("Ahoj, já jsem")
```

57. Objektově orientované programování (OOP) - K čemu jsou interfaces?

https://youtu.be/fxjyzheCfSo

```
// Interface
interface IamGreeting {
    name: string

    greet(phrase: string): void
}

class Person implements IamGreeting {
    name: string

    constructor(n: string){
        this.name = n
    }

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }
}

let person2: IamGreeting
person2 = new Person("David")
console.log(person2)
person2.greet("Ahoj já jsem")
// umazali jsme metodu sayHi()
```

58. Objektově orientované programování (OOP) - Interface a readonly

https://youtu.be/tC76iuWxXEM

```
// public, private není možné u interfacu
// readonly je možné

interface IamGreeting {
    readonly name: string

    greet(phrase: string): void
}

class Person implements IamGreeting {
    name: string

    constructor(n: string){
        this.name = n
    }

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }
}

let person2: IamGreeting
person2 = new Person("David")
// person2.name = "Harry"
console.log(person2)
person2.greet("Ahoj já jsem")
```

59. Objektově orientované programování (OOP) - Interfaces a extends

https://youtu.be/Js_8WEtNz94

```
// Jedna class více interfaces
// Nebo jeden interface je rozšířený o další interface

interface Name {
    readonly name: string
}

interface IamGreeting extends Name {
    greet(phrase: string): void
}

class Person implements IamGreeting {
    name: string

    constructor(n: string){
        this.name = n
    }

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }
}

let person2: IamGreeting
person2 = new Person("David")
console.log(person2)
person2.greet("Ahoj já jsem")
```

60. Objektově orientované programování (OOP) - Interface jako funkce

https://youtu.be/fcjKl4GXz18

```
// Interface jako funkce

interface AddFunction {
    (a: number, b: number): number
}

let sum: AddFunction

sum = (number1: number, number2: number) => {
    return number1 + number2
}
```

61. Objektově orientované programování (OOP) - Volitelné parametry a metody

https://youtu.be/3qrk0Ia8xNE

Přidali jsme bílé části

```
// Volitelné parametry a metody


interface Name {
    name: string
    age?: number
}
```

```
class Person implements IamGreeting {
    name: string
    age?: number

    constructor(n: string, ag: number = 30){
        this.name = n
        // if(ag){
        //     this.age = ag
        // }
        this.age = ag
    }

    greet(myPhrase: string){
        console.log(myPhrase + " " + this.name)
    }
}
```

62. Objektově orientované programování (OOP) - Překlad interface do javascriptu

https://youtu.be/hCQxZjGIqog