09 JSDOC, undefined, BOOLEAN

toto je povedzme špecialitka: kolegyňa v staršej verzii kurzu narazila na nasledovný problém - ak si vytvorím premennú typu string v hlavnom programe:

```
let firstName = 'balls';
```

tak pri `firstName.` mi VSCode krásne dopĺňa všetky metódy, ktorými môžem pracovať so stringom. ale keby mám funkciu `sayHello` by to nefungovalo. v jej vnútri pri `firstName.` nevie, čo má dopĺňať. ale ty môžeš VSCodu pomôcť, ak pridáš tzv. JSDoc komentár:

```
/**
 * POZDRAVIME CLOVEKA (a bude to krasne)
 * @param {String} firstName
 * @param {String} lastName
 */
function sayHello(firstName, lastName) {
  console.log('Hello ' + firstName + ' ' + lastName + '!!');
}
```

toto je tzv. JSDoc komentár. doc akože dokumentácia. podobne ako MDN dokumentuje vstavané funkcie, ty si môžeš zdokumentovať vlastnú funkciu. teraz VSCode chápe, že meno bude typu String a vie dopĺňať kód. ak si počul o **TYPESCRIPTE**, tak typescript je toto, len komplikovanejšie.

znova: často veľmi užitočná vec pri väčších projektoch a pri práci v tíme. ak ja napíšem funkciu a vyplním tieto VSCode komentáre a kolegyňa bude používať moju funkciu, textový editor jej bude radiť.

ABSENCIA HODNOTY

pracujeme s hodnotami. nonstop. ale často sa stretneme s **absenciou** hodnoty. ako som spomínal, keď píšeme kód, nepoznáme reálne údaje, ktoré budú do našej appky prichádzať. a naša úloha je napísať kód tak, aby fungoval, aj keď user veci vyplní blbo. alebo nevyplní vôbec. vráťme sa k pozdravu:

```
sayHello('Vašimír');
```

v konzole zasvieti **undefined**. funkcie chce 2 veci, ale poslali sme len jednu. undefined je absencia hodnoty. undefined je, že hodnota neexistuje. nebola definovaná.

napríklad ak `score` je nula a a vypíšem score:

```
let score = 0;
console.log(score);
```

tak chill. ale ak score vytvorím ale nezadám hodnotu:

```
let score;
```

`undefined`, nemáme, nevieme, hodnota nebola definovaná...

PODMIENKY, BOOLEAN

často teda dáva zmysel kód spúšťať **iba ak** máme hodnoty:

```
if (score) {
  console.log('nahral/a si: ' + score);
}
```

na ulohú skúste podobné podmienky zapracovať do všetkých vašich funkcií, ktoré očakávajú údaje. pri podmienkach ešte chcem spomenúť `&&` a `||`:

```
// táto podmienka platí AK máme firstName A ZÁROVEŇ (a současně) máme aj
lastName
if (firstName && lastName)

// táto platí ak máme firstName ALEBO lastName
if (firstName || lastName) {}
```

`&&` znamená AND, **a**
`||` znamená OR, **alebo**

výsledok podmienky je `true` alebo `false`. podmienka buď platí (true) alebo neplatí (false). v kurze JavaScript a ES6 nájdeš doplňujúce detaily. alebo si môžeš v dokumentácii dohľadať koncept tzv. Truthy https://developer.mozilla.org/en-US/docs/Glossary/Truthy a Falsy https://developer.mozilla.org/en-US/docs/Glossary/Falsy hodnôt.

UNDEFINED, NULL

`undefined` je často chyba. ale absencia hodnoty nemusí byť nutne chyba. existuje dátový typ `null`.

`null` keď ty ako programátor vyložene povieš, že vec nemá hodnotu. `undefined` keď hodnota proste neexistuje.

undefined nikdy nezadávaš ty, to javascript vyrobí automaticky.

často, keď ti konzola vypíše `undefined`, tak vieš, že nastala chyba. pretože väčšinou chceš pracovať s hodnotami. a `undefined` znamená, že vec neexistuje. premenná nemá žiadnu hodnotu: asi chyba. snažíš sa zavolať funkciu, ktorá neexistuje: asi chyba.

ale aha, napríklad ak tu https://mdbootstrap.com/docs/standard/extended/bootstrap-address-form/ nevyplníš políčko "company", to neni chyba. ak niečo objednávaš z eshopu, nemusíš to objednávať na firmu a nemusíš vždy nutne vyplniť absolútne všetky inputy formulára.

takže ak toto pole zostane prázdne, programátor môže použiť hodnotu `null`.

`null` znamená, že hodnota neexistuje, ale úmyselne. `undefined` znamená, že hodnota neexistuje.