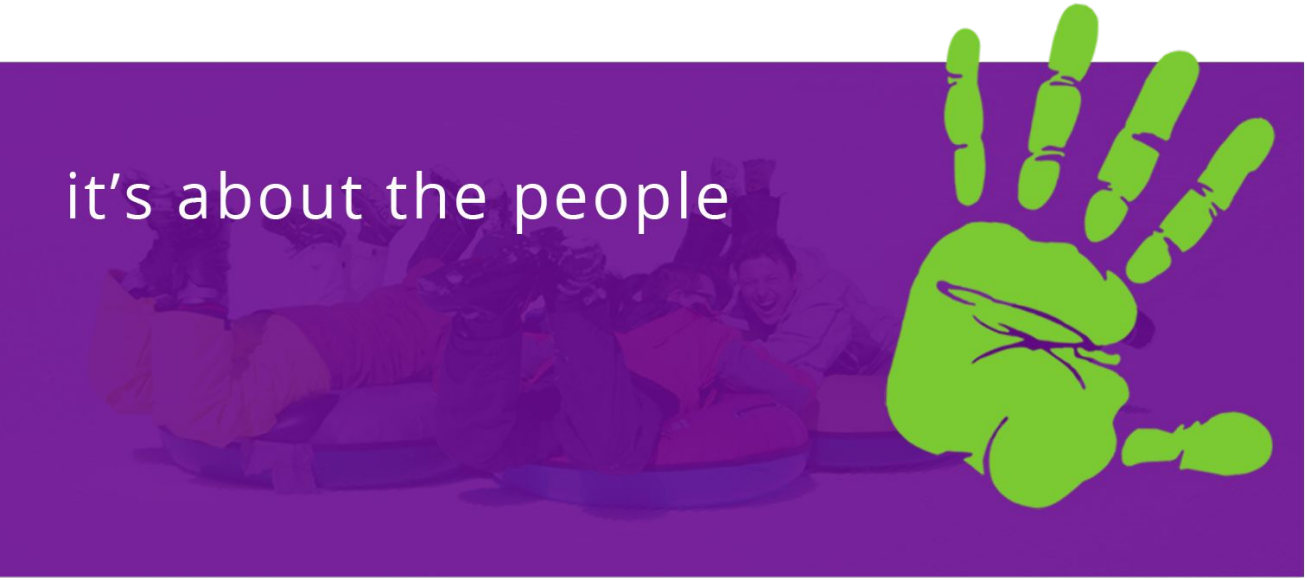


inventi

Frontend Akademie 1
Javascript 1

Jakub Ferenc

17.4.2018

Obsah

- Trochu kontextu JS (historie, současnost, „JS fatigue”)
- První a rychlý debugging kódu
- Hrátky s proměnnými a strukturami
- Bez funkcí to nefunguje
- Lexikální scope, closure, callback a this: kameny JS úrazů
- „O DOM to je“: DOM, HTML elementy a události
- Úvod do objektů
- Domácí úkol

@jakub_ferenc

Kde jsem “dělal front-end a UX”

- Mangoweb, Hypoteční banka, Ústav pro studium totalitních režimů, assemblage

Věčný student

- Informační technologie, UK
- Nová média a filosofie, FF UK
- Kognitivní informatika, VŠE
- Etnografie a filosofie, Universität Bonn
- Diplomová práce: *Postkognitivistické HCI: Interface jako sociotechnický vztah*

inventi

Otázky na úvod


I DON'T ALWAYS CODE
BUT WHEN I DO, I PREFER JAVASCRIPT
quickmeme.com

Otázky na úvod

- Nejzajímavější informace z historie Front-endu a JS?
- Hoisting?
- Scope a Closure?
- Proč na JS funkcionálně?

inventi

Trochu kontextu JS

Trochu kontextu JS

- Války prohlížečů
- JS versus ECMAScript
- JS implementace a prostředí
- ES6/ES2015
- „JavaScript fatigue“
- Koho sledovat?
- Proč JavaScript?

inventi

První a rychlý debuging kódu

První a rychlý debugging kódu

- **funkce** alert()
- **metoda** console.log()
- **operátor** typeof
- https://codepen.io/
- https://jsfiddle.net/

inventi

Hrátky s proměnnými a strukturami

Hrátky s proměnnými a strukturami

- Operátory
  - Matematické: +,-,*,/,%,
  - Porovnání: <,>,=,==,<=,=>
  - Logické: &&, ||, !
  - Ternární: ?
  - Speciální: typeof, this, instanceof

Hrátky s proměnnými a strukturami

- Deklarace proměnných
  - let
  - const
  - var (je dobré znát, ale již nepoužívat)
  - bez operátoru (nepoužívat)
- Jak se liší?
  - let: platnost v rámci bloku (block-scoped)
  - const: jako let, plus nelze znovu přiřadit hodnotu
  - var: globální nebo scope ve funkci
- Block-scope?
  - vše uvnitř “{}”, relevantní pouze pro let a const

Hrátky s proměnnými a strukturami

- V JS jsou proměnné dynamické
- Typy proměnných
  - string
  - number
  - boolean
  - array
  - object
  - undefined, null

Hrátky s proměnnými a strukturami

- String (řetězec)
  - kombinace řetězců a šablony v ES6
  - mnohařádkové řetězce v ES6
  - každý string je objekt a má metody
  - Nativní objekt String
  - užitečné metody: String.prototype.includes, String.prototype.indexOf

Hrátky s proměnnými a strukturami

- Number (číslo)
  - JS má pouze jeden číselný datový typ
  - double-precision 64-bit binary format IEEE 754 (hodnoty $-(2^{53} -1)$ až $2^{53} -1$)
  - Čísla jsou také objekty a mají metody
  - Nativní objekt Number (nabízí např. metody isInteger())

Hrátky s proměnnými a strukturami

- Boolean
  - dvě hodnoty: true, false
  - Nativní objekt Boolean

Hrátky s proměnnými a strukturami

- Array
  - vytvoření pomocí “[]” (new Array() konstrukce není třeba, naopak dělá problémy)
  - Přístup k poli pomocí indexu pole[index]
  - Nativní objekt Array
  - Užitečné metody Array.prototype.forEach, Array.prototype.keys, Array.from, Array.prototype.length

Hrátky s proměnnými a strukturami

- Object
  - v JS je vše objekt
  - vytvoření pomocí {} uvozovek (není důvod preferovat new Object() konstrukci)
  - Nativní objekt Object
  - Užitečné metody Object.assign(), Object.keys(), Object.create()

Hrátky s proměnnými a strukturami

- Undefined, null
  - Undefined: existuje deklarovaná proměnná, ale nemá hodnotu
  - Null: symbolizuje absenci hodnoty

Hrátky s proměnnými a strukturami

- Hoisting
  - **Pouze** deklarace proměnných s “var” a celé funkce se posouvací při interpretaci kódu zcela nahoru. Jedna z “weird” vlastností JavaScriptu, tak bacha na undefined proměnné!

Hrátky s proměnnými a strukturami

- Řídicí struktury stejné jako v jiných jazycích
  - Cykly
    - For, while, for...in, for...of
  - Podmínky
    - If
    - ternární operátor
    - switch

inventi

Bez funkcí to nefunguje

Bez funkcí to nefunguje

- V JS jsou funkce „first-class citizens“
- Deklarace a inicializace
  - function nazev () {}
  - var anonymni_funkce = function() {}
  - const nazev_funkce = () => {}
- argumenty a return
- scope a closure
- this
- callback funkce

Bez funkcí to nefunguje

- V JS jsou funkce „first-class citizens“
  - hodnota proměnné
  - argument funkce
  - hodnota vracená funkcí
  - anonymní (nepojmenovaný)
  - funkce jako každá jiná proměnná
  - https://en.wikipedia.org/wiki/First-class_citizen

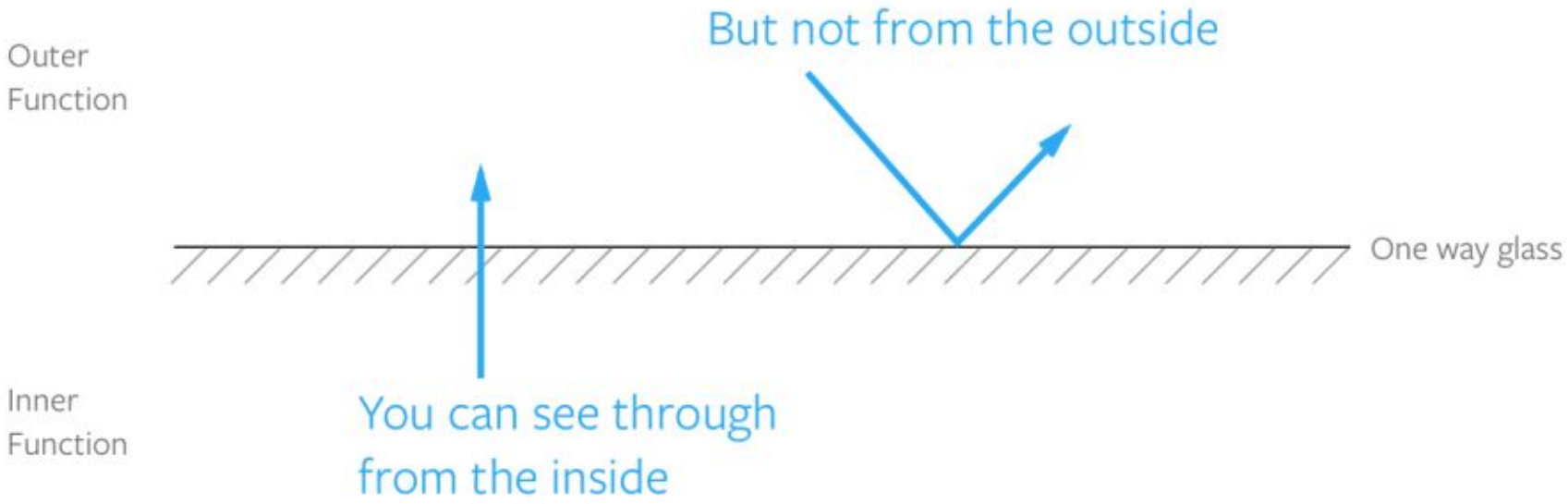
Bez funkcí to nefunguje

- Deklarace a inicializace
  - function() {}
  - () => {}
    - “Arrow funkce” v ES6
    - nevytváří nové “this” (později si ukážeme)

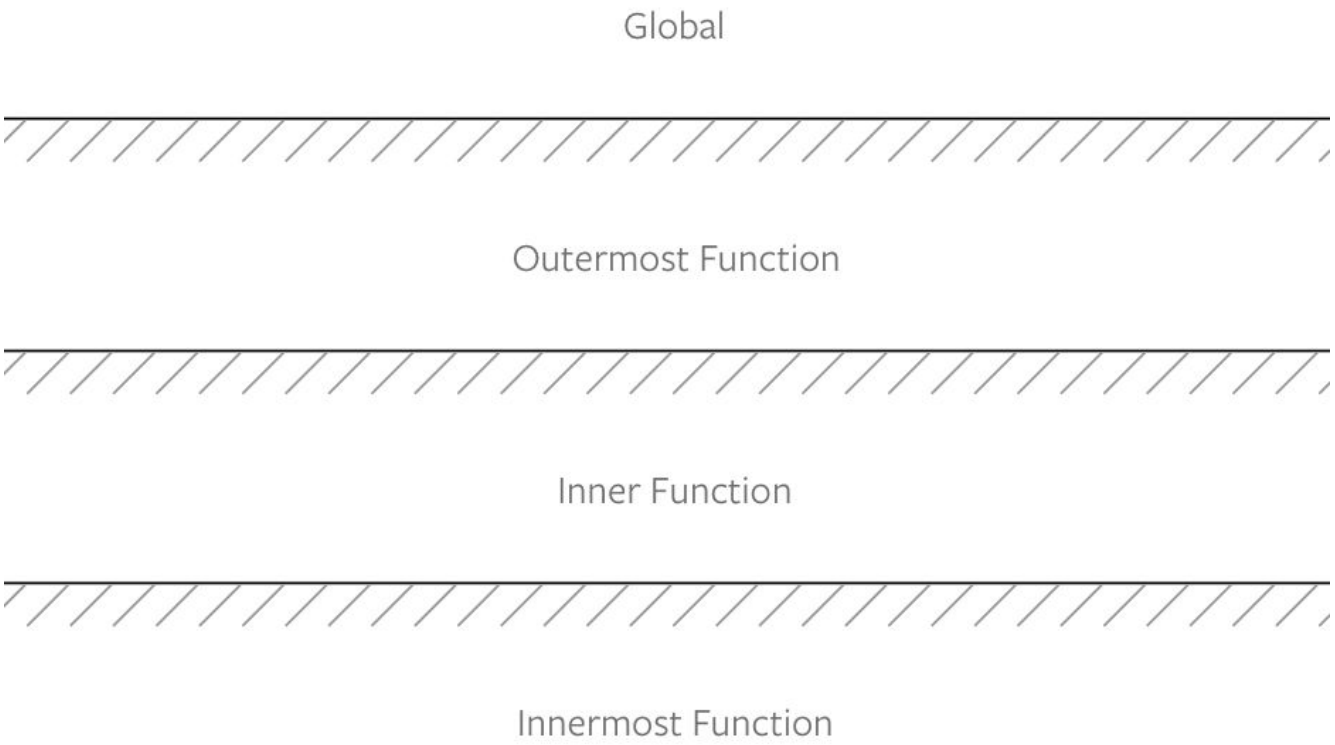
Bez funkcí to nefunguje

- Scope
  - Typy
    - Globální
    - Lokální, vytvořené funkcí nebo “{}”
  - Představme si scope jako zatemněná skla: vidíme ven, ale ne dovnitř
  - Mohou se do sebe vrstvit

Bez funkcí to nefunguje



Bez funkcí to nefunguje



Bez funkcí to nefunguje

- Closure
  - Funkce si s sebou “nesou” nejen vlastní scope, ale i scope svého rodiče a globální scope
  - Tomuto “balíku”, který si funkce nese, se říká closure
  - Použití: například simulace privátních proměnných

Bez funkcí to nefunguje

- This
  - operátor odkazující ke kontextu objektu, ve kterém je volán
  - závisí na tom, jak, kdy a kde je volán

Bez funkcí to nefunguje

- Callback funkce
  - Každá funkce, která je předávána jako argument jiné funkce, je **call back funkce**
  - Většinou volána po dokončení nějaké předchozí funkce
  - Používána například pro asynchronní volání nebo zpracování DOM eventů

Funkcionální aspekty JS

- Co to znamená “funkcionální”
- Výhody a nevýhody
- Konkrétní příklady
  - Pure funkce, nemění stav programu
  - Immutability, vždy vrací novou hodnotu
  - Imperativní versus deklarativní styl

inventi

„O DOM to je“: HTML elementy, události

„O DOM to je“: HTML elementy, události

- Co je Document Object Model (DOM)?
- Jak pracovat elementy?
  - Rozdíl mezi getElementsByClassName a querySelector/querySelectorAll
  - Manipulace s DOM elementy http://youmightnotneedjquery.com/
  - createElement, appendChild, innerHTML, getAttribute, setAttribute aj.
- Události
  - událost jako atribut
  - addEventListener, removeEventListener
  - callbacks (znovu!)
- Styly
  - getComputedStyle, atribut style

inventi

Úvod do objektů

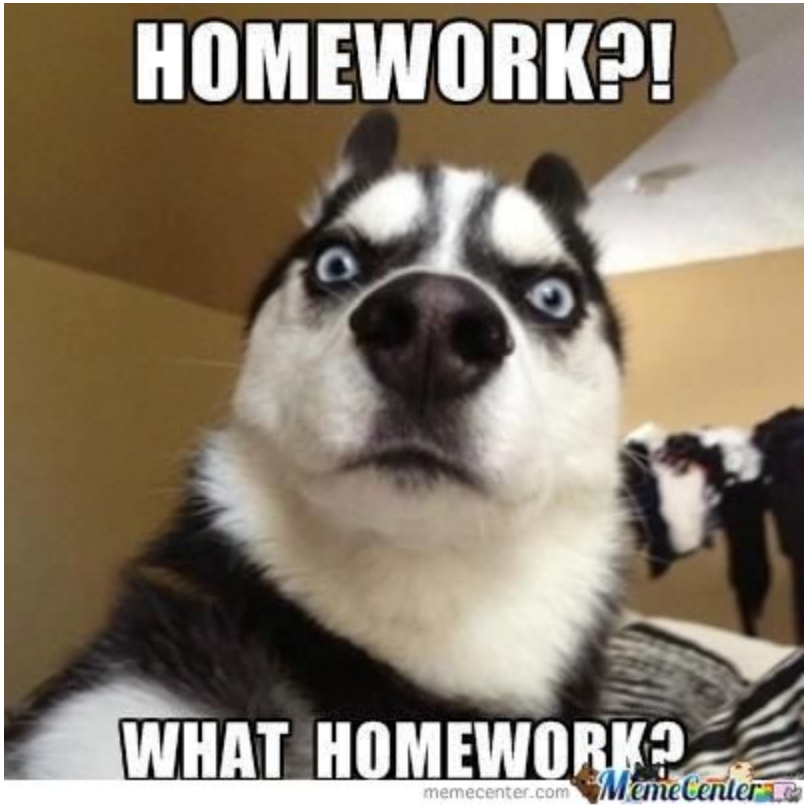
Úvod do objektů

- Deklarace
  - funkce
  - {}
  - new Object()
  - Object.create
- Objekty v JS vs další jazyky
- Zbytek příště

Příště

- Podrobněji funkcionální prvky JS
- Ponoření do objektů
- Třídy
- Moduly
- Hraní si s GUI peněženky a aplikace znalostí

Domácí úkol?!



Domácí úkol?!

- Zadání
  - Vytvoř si pole několika objektů, kde každý objekt bude symbolizovat platbu a bude obsahovat atributy typ účtu (EUR, CZK), popis útraty a číselnou částku
  - Projděte pole objektů pomocí forEach a přidejte každému objektu nový atribut “castkaPrint”, který bude příkazem return vracet částku a symbol měny.
  - V HTML stránce dle svého uvážení zobrazte všechny platby a jejich atributy
  - V HTML vypíšete celkovou útratu v CZK i EUR vypočítanou pomocí metod “filter” a “reduce”