SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ


BRACE YOURSELF
A LOT OF JAVASCRIPT IS COMING
memegenerator.net

Изрази и оператори - II част

- Миналият път споменахме логическите и аритметичните оператори:
  - те се използват за да извършваме операции като сравнение и събиране върху **прости типове данни**
    - те могат да бъдат унарни (напр: !) или бинарни (напр: &&)
- също така говорихме за групиращия оператор **()** и как с негова помощ да правим композитни изрази като:

```
(wallet.amount >= (item.price * 3)) || (creditCard.balance > 100)
```

Изрази и оператори - II част

- В JavaScript обаче има още цял куп оператори, които можем да ползваме, като:
  - typeof
  - ++ и --
  - += и -=
- https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators

Условен преход (if - else конструкция)

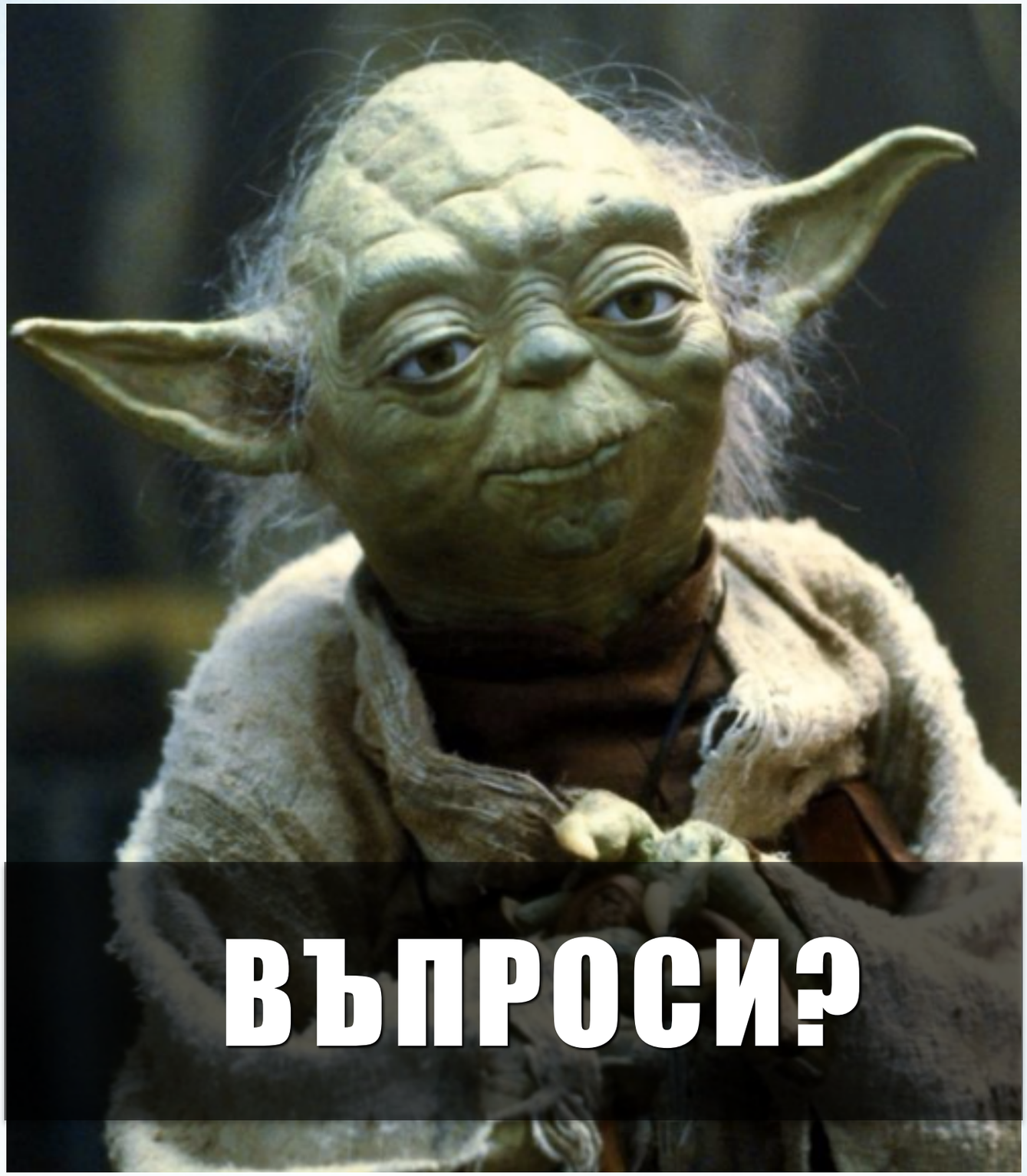
- Често искаме да проверим някакво условие и ако то е вярно да направим едно действие, а ако не е вярно - друго
- За целта има конструкция, която ни позволява да направим точно това
- Синтаксис:

```
if (условие) {
   // действия
} else {
   // действия
}
```

Тринарен оператор и switch

- if и else са запазени думи от езика и сформират конструкция
- подобен ефект има и конструкцията switch - case (https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/switch)
- от зората на програмирането е останала и една short-hand конструкция или т.нар. тринарен оператор, който е кратък запис за if - else израза
- Синтаксис:

```
condition ? true_actions : false_actions
```


ВЪПРОСИ?

f(js)

функции в JavaScript

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Front-End Development

Функции

- Това е едно от най-фундаменталните неща в JavaScript (и в повечето езици за програмиране)
- Използват се за съхраняване на код, който ще се изпълнява при всяко извикване на функцията
- Функциите могат да приемат входящи стойности (аргументи)
- Функциите прилагат кода, който съхраняват върху получените при извикването си аргументи
- Функциите могат да връщат стойност (резултата от прилагането на кода в тях върху аргументите)

Синтаксис

```
function myFunctionName(myParam1, myParam2, …) {
  var result = "set initial result value";
  // do some calculations
  return result;
}
// calling (executing) the function with arguments:
console.log(myFunctionName(42, 'param 2 value'));
```

The return statement

- За да върнем стойност с нашата функция, използваме запазената дума **return**
- Когато изпълним функцията, резултатът от нея ще бъде стойността върната с return
- Ако функцията не изпълни return, то резултатът от нея ще бъде undefined
- Във функцията можем да имаме повече от един return, като изпълнението й ще приключи при достигането до първият от тях или при достигането до края на тялото на функцията (което се случи по-напред)

Вградени функции

- JavaScript предлага известен брой 'вградени' (built-in) функции, които са включени в езика по подразбиране и могат да се ползват наготово
- Примери за built-in функции са Array.indexOf(), Math.round() и Date.now(), където Array, Math и Date са глобални обекти (ще стигнем и до там)
- http://www.tutorialspoint.com/javascript/javascript_builtin_functions.htm

Извикване на функции

- След като сме си запазили някаква последователност от изчисления във функция, бихме могли да я изпълним във всеки момент, в който си поискаме
- действието, което кара функцията да се изпълни, се нарича "извикване"
- извикването на функцията става, като изпишем името ѝ, последвано от кръгли скоби: **fn()**
- извикването може да бъде със или без аргументи: **fn(x)**


I DON'T OFTEN DO FUNCTIONAL PROGRAMMING
BUT WHEN I DO, IT'S ALWAYS IN JAVASCRIPT
memegenerator.net

Функционално програмиране

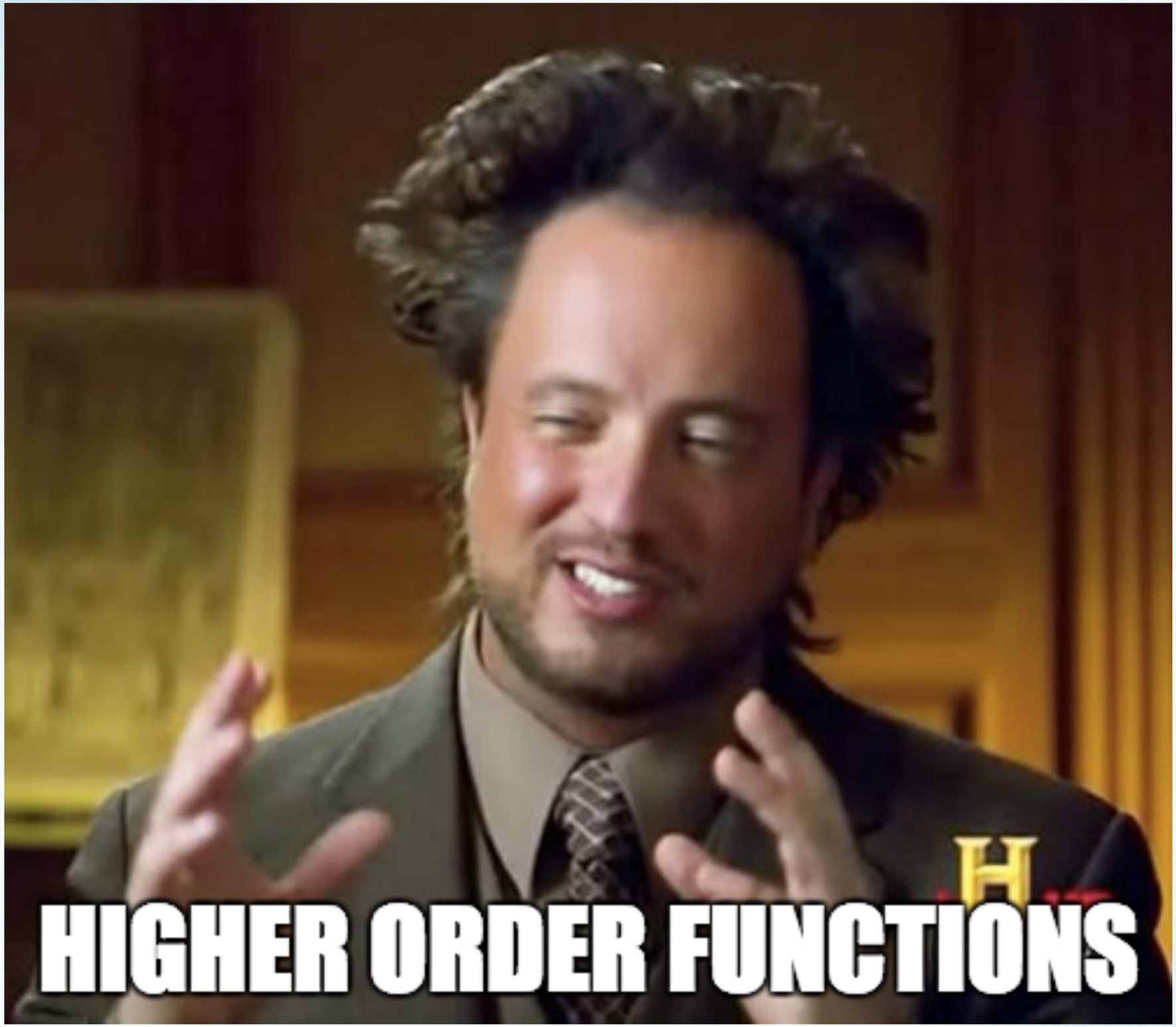
- Всяка функция прави едно конкретно нещо
- Функцията работи единствено с параметри (без да ползва променливи, които са декларирани извън нея)
- Функцията връща стойност
- За всяко действие създаваме функция, като се стараем тя да е generic (да може да се ползва универсално)
- Използване на функции от по-висок ред (higher-order functions) - т.е. композиране на функции

Да започнем с примери

- for цикълът:

```
for (var i = 1; i < 10; i++) {
  console.log('current index: ', i);
}
```

- търсене в списък (find)
- прилагане на едно и също действие върху всички елементи от списъка (map)
- редуциране на списък (reduce)
- филтриране на списък (filter)
- сортиране на списък (sort)


HIGHER ORDER FUNCTIONS

Higher order functions - warming up

- Какво е всъщност функцията в JavaScript ?
- Функциите могат да се присвояват (задават като стойност) на променливи
- Функциите могат да се подават на други функции като параметри
- И накрая - функциите могат да бъдат връщани като резултат от други функции
- **Заключение**: можем да боравим с функцията по същият начин както и с обикновенна стойност!

Higher order functions

- Функция, която приема като аргумент друга функция и след това я използва за изчисляването на резултата си, се нарича Функция от по-висок ред
- Функцията, която подаваме като аргумент, се нарича callback функция
- Функциите, които създаваме и съхраняваме в променлива или подаваме като аргументи, могат да бъдат анонимни (без наименование)
- https://www.youtube.com/watch?list=PL0zVEGEvSaeEd9hlmCXrk5yUyqUag-n84&v=BMUiFMZr7vk


I need to implement some HOF to get this script working!
Why not using many nested loops instead?

Въпроси?

Полезни връзки

- **JavaScript best practices**:
  https://www.w3.org/wiki/JavaScript_best_practices
  (това е библията на програмиста и трябва да се знае наизуст !!)

- W3schools
  http://www.w3schools.com/jsref/jsref_if.asp
  http://www.w3schools.com/jsref/jsref_function.asp
  http://www.w3schools.com/jsref/jsref_return.asp

- Интересни (и малко шумни) видео уроци:
  https://www.youtube.com/channel/UCO1cgjhGzsSYb1rsB4bFe4Q/videos

Примери

http://zenlabs.pro/courses/lessons/lesson12/examples.zip

https://repl.it/CMz4

Домашно

https://github.com/zzeni/swift-academy-homeworks/blob/fe-03/tasks/L12