SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

JavaScript

SWIFTACADEMY

НИЕ ВЯРВАМЕ ВЪВ ВАШЕТО БЪДЕЩЕ

Front-End Development

Интерпретативни езици за програмиране

- Всяка програма, написана на език за програмиране представлява последователност от команди (инструкции), изчислителни и логически операции

- JavaScript е интерпретативен език, което означава, че командите от нашата програма, се интерпретират директно от браузъра (без предварително да се компилират) до получаването на конкретен резултат

- Разликата между интерпретативните езици и тези, които изискват предварителна компилация (C++, Java, etc..), е че interpreted езиците позволяват да има грешки в кода, като тяхното изпълнение ще спре при първата грешка

- При компилаторните езици, ако има грешка, изпълнението е невъзможно, тъй като грешката възниква още по време на компилация (компилацията е процес, който преобразува source кода в изпълним код)

Интерпретатори

- Браузърът интерпретира JavaScript кода "ред по ред" и така на всяка стъпка прави някакво изчисление или изпълнява логическа операция

- Т.е. браузърът има вграден JavaScript интерпретатор

- Същият този интерпретатор се използва и от JavaScript конзолата на браузъра

- Именно конзолата ни позволява да въвеждаме нашата програма (набор от инструкции) ред по ред (т.е. инструкция по инструкция), като за всеки ред ще получаваме междинните резултати от изпълнението

One does not simply learn JavaScript

Синтаксис

- Както обикновенните езици, така и тези за програмиране, си имат синтаксис
- Това са правилата за формиране на думите и изразите от езика в разбираема за интерпретаторите последователност
- Синтаксиса на JS ще взимаме в движение, но като за начало можем да кажем следните 2 основни неща:
  - всяка команда (ред) завършва с ; (точка и запетая)
  - всяка "дума", която не е наименование, дефинирано от нас, не е дума от езика и не е коментар - води до грешка

Коментари

- Както във всеки друг език, така и в JavaScript, можем да пишем прозиволен текст като коментар, който се игнорира от интерпретатора

- Има два вида коментари:
  - едноредови:

```
alert('hi!'); // от тук до края на реда, всичко се игнорира
```

  - многоредови:

```
/* Този коментар би могъл да бъде поставен навсякъде
   както и да се разпростира на няколко реда */
```

Типове данни

- Програмата работи с данни. Те са обикновенно *числа*, *текст* или пък са по-сложни структури като *списъци* и *обекти*

- Данните са това, което програмата ползва за да направи съответните изчисления:
    - Например ако въведа някъде рожденната си дата - програма може да изчисли на каква възраст съм

- Данните също могат да се използват за логически операции:
    - Например ако изчислената възраст е под 18 години, може да ми бъде отказан достъп до някакво съдържание

Числа (Numbers)

- Цели числа: например 2 или -10
- Числа с плаваща запетая: 3.5
- С числата можем да извършваме основните аритметични операции:
  - събиране: 2+3
  - изваждане: 10-4
  - умножение: 2*3
  - деление: 10/5

Числа - особености

- операции със скоби - както в математиката

- при деление резултата понякога е число с плаваща запетая, ако искаме да го направим цяло, можем да използваме методът **round** на класа **Math**, ето така:

```
Math.round(8/3)   // => 3
```

- Понякога искаме да разберем какъв е остатъка при деление (модул), това става по следния начин:

```
8%3 // => 2
```

Текстов низ (String)

- Всеки набор от символи, заграден в двойни или единични кавички е String
  - "hello"
  - "1"
  - "Some very long text inputs are also strings"
  - 'a'
  - ""

Основни операции със текст

- Съединяване (concat):

```
"hello" + ", world!" // "hello, world!"
```

- Дължина:

```
"hello, world!".length // 13
```

- Символ на определена позиция:

```
"hello, world!".charAt(4) // 'o'
```

- Индекс (позицията) на определен символ:

```
"hello, world!".indexOf('l') // ‘2'
"hello, world!".indexOf('x') // ‘-1'
```

Списъци (Arrays)

- Наричат се също масиви
- Това е колекция от елементи, например числа и текст
- записва се, като елементите на колекцията се изреждат между квадратни скоби, разделени със запетайки:

```
['a', 1, 'w']  // => ["a", 1, "w"]
```

- Използваме ги, за да групираме няколко елемента (най-често еднотипни), с цел - да можем да изпълняваме определени действия върху всички елементи от списъка

Списъци - особености

- Дължина:

```
[1, 2, 3].length // 3
```

- Достъп до елемент

```
["hello", "world"][0] // "hello"
```

- Индекс (позицията) на определен елемент:

```
["hello", “world"].indexOf("world") // 1
```

- *Първият елемент на string или на масив, е винаги с индекс 0!*

Булеви стойности (Boolean)

- **true** и **false**
- Използват се за да кажат дали нещо е вярно или невярно
- Използват се и се получават при изпълнение на логически операции
- Примери:

```
1 > 2                  // false
true == false          // false
"hello" != "world" // true
```

Обекти

- Обектите са, така да се каже, "яката работа" в JavaScript
- Това което представляват те, е не просто стойност
- Те са съвкупност от характеристики и могат да имат собствено "поведение" (функционалност)
- Засега просто ще кажем, че обектът се записва в къдравите скоби по следния начин:

```
{ first_name: "Chuck", last_name: "Norris", age: 76 }
```

42

Number

"Yesterday, all my troubles seemed so far away .."

String

```
{
  race:   "minion",
  colour: "yellow",
  height: "30cm",
  iq:      3
}
```

Object

Променливи

- Променливите са именовани *указатели*, които използваме за да съхраняваме в тях данни
- Както се подразбира от името, те могат да имат най-различни стойности, които не са постоянни
- Представяйте си ги като имена или етикети на дадени стойности/обекти
- Синтаксиса за създаване на променлива е следният:

```
var pi = 3.14;
```

- Може и така:

```
var myVar;
```

Именуване

Приета е следната конвенция за именуване на променливи в JavaScript:

1. имената да са максимално екплицитни (описателни)
2. използват се думи от английския език

3. ако името на променливата се състои от няколко думи, се изписват "слято", като първата дума започва с малка буква, а всяка следваща с главна (CamelCase):

```
var myAwesomeVar = 1;
```

Undefined

- Когато нещо няма стойност:

```
var myVar;     // undefined
[1, 2, 3][4]; // undefined
```

- Дефинирани са всички, променливи, на които сме задали конкретна стойност

- Когато една променлива не е дефинирана, нейната булева стойност е **false**

- *Използването на недефинирани променливи, може да доведе до грешка!*

Логически и аритметични операции

- Аритметичните операции са тези, които се извършват върху числови данни и резултатът от тях е пак число (+, -, *, /, % и т.н.)
- Логическите операции се използват за сравнение на данни или за формиране на условие и те могат да се използват за всякакви видове данни. При тях резултата е **true** или **false**
- Логически оператори: <, >, ==, !=, ===, !==, !, &&, || и т.н.
- Логически изрази:
  - `(2 > 1) && (3< 5)  // true`
  - `(age > 18) || (parent !== undefined)`

Логически сравнения и оператори

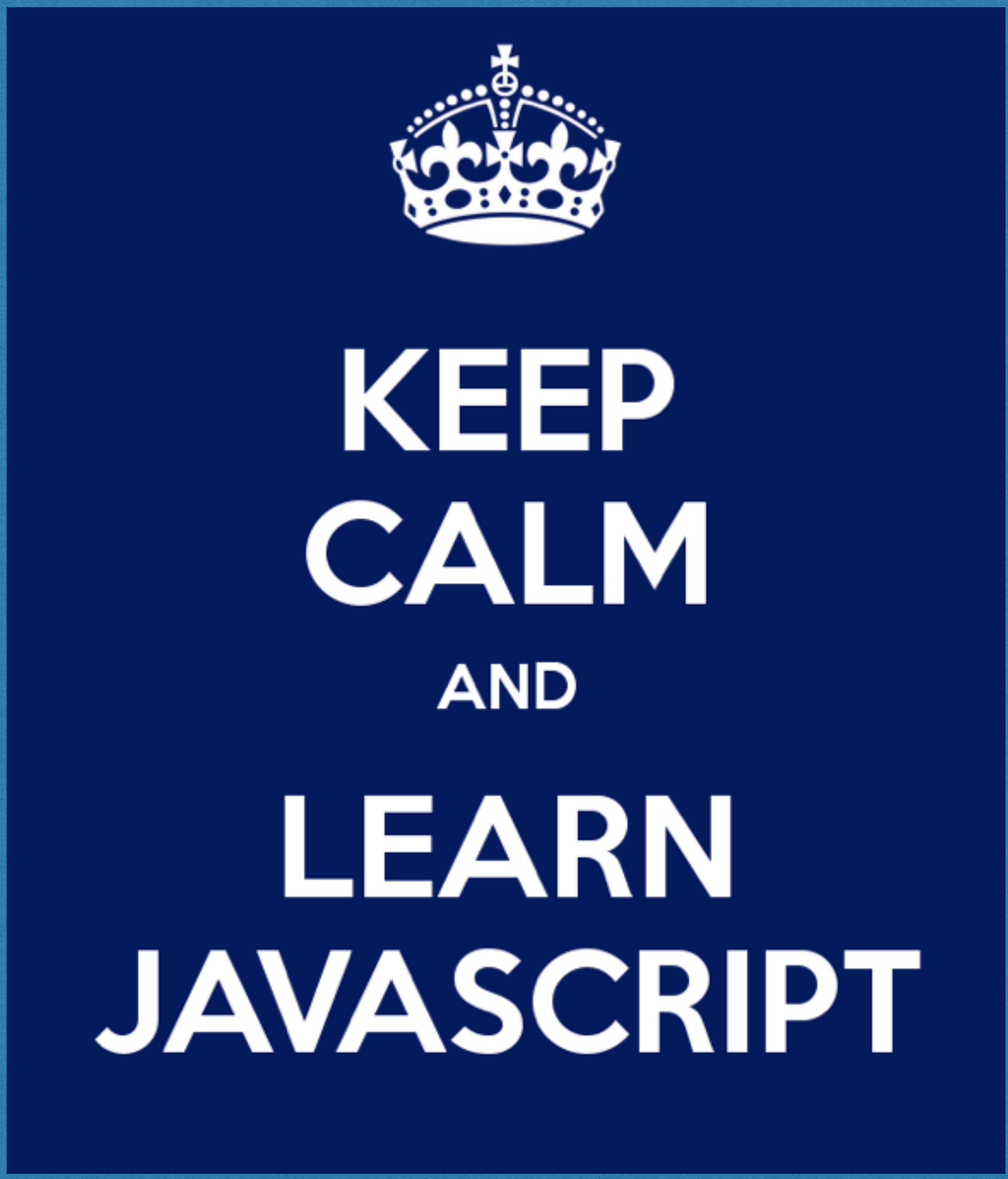
> , >=	"по-голямо", "по-голямо или равно" (за числа)
<, <=	"по-малко", "по-малко или равно" (за числа)
&&	логическо "и"
||	логическо "или"
!	логическо отрицание

Логически сравнения и оператори

==	сравнение за "еднаквост" по стойност
===	сравнение за "еднаквост" по стойност и по тип
!=	сравнение за "различност" по стойност
!==	сравнение за "различност" по стойност и по тип

http://www.w3schools.com/js/js_comparisons.asp

Въпроси?


KEEP
CALM
AND
LEARN
JAVASCRIPT

Полезни връзки

- Codecademy: https://www.codecademy.com/learn/javascript
- Интерактивен онлайн интерпретатор: https://repl.it/languages/javascript
- Загубалка на време: https://blockly-demo.appspot.com/static/demos/interpreter/index.html

Примери

http://swift-academy.zenlabs.pro/lessons/lesson12/examples/download.zip

https://repl.it/CKZe/2

Домашно

http://swift-academy.zenlabs.pro/lessons/lesson12/homework