Основи на JavaScript

II-ра част

25.10.2016

Георги Христанов
georgi.hristanov@sap.com

Public

Съдържание

- ✓ Алгоритъм за конвертиране
- ✓ Масиви
- ✓ Обекти
- ✓ Функции

Преговор

Кои са типовете данни в JavaScript?

- ✓ Boolean, Null, Undefined, Number, String, Object , *Symbol

Кои са „falsy“ стойностите в JavaScript?

- ✓ false, 0, undefined, null, “”/’’(празен низ), NaN

Алгоритъм за конвертиране

Алгоритъм за конвертиране
Въведение

```
1 if ("potato") {
2   console.log("potato" == false);
3 }
```

?

false

```
1 if ("potato") {
2   console.log("potato" == true);
3 }
```

?

false

ЗАЩО?

“Coercion/Abstract Equality Comparison Algorithm”

(Алгоритъм за конвертиране)

Алгоритъм за конвертиране

Описание оператор за сравняване ==; 1/2

X == Y ?

Тип на X	Тип на Y	Резултат
еднакви типове		Както ===
null	Undefined	true
Undefined	null	true
Number	String	x == toNumber(y)
String	Number	toNumber(x) == y
Boolean	(any)	toNumber(x) == y
(any)	Boolean	x == toNumber(y)
String or Number	Object	x == toPrimitive(y)
Object	String or Number	toPrimitive(x) == y
Във всички останали случаи…		false

Undefined е равен на null. Всичко друго се конвертира почти винаги до число преди да се извърши сравнение.

Алгоритъм за конвертиране

Описание оператор за сравняване ==; 2/2

toNumber(Z)

Тип на Z	Резултат
Undefined	NaN
Null	+0
Boolean	**1** при **true**, **0** при **false**
Number	Z (няма конверсия)
String	Както new String (str) “777” / „777FMI“ -> 777 “FMI” / “FMI777” - > NaN
Object	Прилагане на следните стъпки: 1. primitiveValue = ToPrimitive(Z) 2. Връщаме toNumber(primitiveValue)

toPrimitive(Z)

Тип на Z	Резултат
Object	1) valueOf връща примитив -> valueOf 2) toString връща примитив->toString В противен случай – хвърля се грешка
Други	Връща Z

Алгоритъм за конвертиране

Описание оператор за сравняване ===

Тип X	Стойности	Резултат
Типа на X се различава от типа на Y		false
Undefined или Null		true
Number	x е със същата стойност като y (но не и NaN)	true
String	x и y са идентични символи	true
Boolean	x и y са и двете true или и двете false	true
Object	x и y реферират един и същи обеки	true
В противен случай…		false

Алгоритъм за конвертиране
Събиране и изваждане на стрингове

Израз	*Резултат*
“hello” + “world”	“hello world”
“5” + “5”	“55”
“5” + 5	“55”
“5” - “5”	0
“5” - 5	0

Substract operator rules:

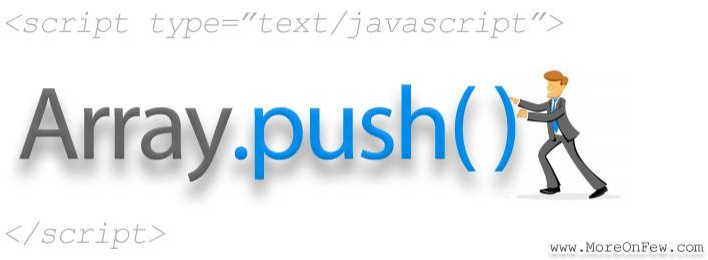
If the two operands are numbers, perform arithmetic subtract and return the result.
If either number is NaN, the result is NaN.
If Infinity is subtracted from Infinity, the result is NaN.
If –Infinity is subtracted from –Infinity, the result is NaN.
If –Infinity is subtracted from Infinity, the result is Infinity.
If Infinity is subtracted from –Infinity, the result is –Infinity.
If +0 is subtracted from +0, the result is +0.
If –0 is subtracted from +0, the result is –0.
If –0 is subtracted from –0, the result is +0.
If either of the two operands is not a number, the result is NaN.

Demo

Public

Масиви

- ✓ Деклариране и инициализация
- ✓ Достъпване на елементи
- ✓ Динамичност
- ✓ Сортиране на масиви



Масиви

Преглед

- **Деклариране и инициализация**
  - new Array(elements)
  - new Array(length)
  - var array = […]
- **Достъпване на елементи**
- **Динамичност**
  - Добавяне на последен елемент – push
  - Премахване на последен елемент – pop
  - Вмъкване на пръв елемент – unshift
  - Премахване на пръв елемент – shift
- **Сортиране**
  - Array.sort()

Масиви

Overview

- **Други полезни методи при масиви**
  - array.reverse()
    - Връща нов масив с елементи в обратен ред
  - array.concat(elements)
    - Добавя подадените елементи към края на масив и връща нов масив
  - array.join(separator)
    - Връща нов низ с елементите на масива разделени със стойността на *separator*
  - array.filter(condition)
    - Връща нов масив с елементи, които удовлетворяват условието
  - array.forEach(function(item){})
    - Итерира по елементите на масивва
  - array.indexOf(element)
    - Връща индекса на първия елемент, който е равен на element или -1 ако не е намерен такъв
  - array.lastIndexOf(element)
    - Връща последният елемент, който е равен на element

Demo

Public

Обекти

- ✓ Какво са обектите?
- ✓ Обектите в JavaScript
- ✓ **J**ava**S**cript **O**bject **N**otation (JSON)
- ✓ Асоциативни масиви
  - ✓ речници и мапове в JavaScript

Обекти

Overview

- **Какво са обектите?**
  - Абстракция, която симулира реални обекти от живота
  - Обектите както в реалният живот, така и в програмирането имат състояния и поведение
  - Създаден обект от определен тип се нарича **инстанция**

- **Обекти в JavaScript**
  - В JavaScript почти всичко е обект
  - Стойностите в обектите се представят като асоциация име и стойност
  - Създават се по няколко начина

Обекти

JSON обекти и асоциативни масиви

- Какво е JSON
  - Стандартен начин за дефиниране на обекти
  - Често се използва за пренос на данни между сървър и клиент
  - Често данните в JSON формат представяват масив от обекти
- В JavaScript обектите могат да се ползват и за асоциативни масиви

```
1  var person = {
2      'firstName': 'Martin',
3      'lastName': 'Hristov',
4      sayHello: function() {
5          return 'Hello I am ' + this.firstName + ' ' + this.lastName;
6      }
7  }
```

Обекти

Вградени обекти

Built-in JavaScript Objects – Date, Arrays, RegExp, etc..

Demo

Public

Функции

- ✓ Деклариране и викане на функции
- ✓ Функции с аргументи
- ✓ Връщани стойности
- ✓ Scope на функция
- ✓ Overloading на функция

$$f(x, y) = x + y$$

Функции
Overview

- ✓ **Какво са функциите?**
  - ✓ Парчета от код
  - ✓ Решават даден проблем
  - ✓ Могат да бъдат викани
  - ✓ Могат да приемат параметри и да връщат стойност
- ✓ **Използване на функции**
  - ✓ Разделяне на проблема на малки парчета
  - ✓ Организация на кода
  - ✓ Подобряване на четимостта на кода
  - ✓ Избягване на повтаряне на код
  - ✓ Кодът става преизползваем

Функции

JavaScript функции

- ✓ **Какво са функциите в JavaScript**
  - ✓ Всяка функция има **име (освен анонимните, които нямат)**
    - ✓ Използва се за викане на функцията
    - ✓ Описва целта на функцията
  - ✓ Функциите в JavaScript **нямат тип**, който трябва да се върне, но могат да **връщат резултат**
  - ✓ Ако не сме казали на функцията какво да върне, тя връща **undefined**
  - ✓ Функциите в JavaScript могат да имат **параметри**
  - ✓ Функциите в JavaScript имат **тяло**
    - ✓ Тялото съдържа кода за изпълнение
    - ✓ Загражда се от { }

```
1 function printNumbers(n) {
2     for (var i = 1; i <= n; i++) {
3         console.log(i);
4     }
5 }
```

Функции

Деклариране и използване

- ## Как се декларират функции в JavaScript?
  - Чрез конструктор за деклариране на функция

```
1  var print = new Function('console.log("Hello")');
```

  - Чрез декларатор за функция

```
1  function print() { console.log('Hello') };
```

  - Чрез израз за функция

```
1  var print = function() { console.log('Hello') };
2
3  var print = function printFunc() { console.log('Hello') };
```

  - В ES6 чрез стрелка (fat arrow function/lambda expression). Обвързани са със скоупа в който са дефинирани.

```
var print = () => console.log('Hello');
```

Функции

Извикване на функции и функции с параметри

- ✓ **Как се извикват функции в JavaScript?**
  - ✓ Име на функцията
  - ✓ Параметри (ако функцията има такива)
  - ✓ Точка и запетая =)

```
1  function printNumber(n) {
2      for (var i = 1; i <= n; i++) {
3          console.log(i);
4      }
5  }
6
7  printNumber(10);
```

```
1
2
3
4
5
6
7
8
9
10
<· undefined
```

Какво ще направи тази функция и какво ще върне ?

Функции

Извикване на функции и функции с параметри

- ✓ Функциите могат да бъдат викани от други функции
- ✓ Или от самите себе си ... (рекурсия)

```
1  function printNumber(n) {
2      if (n == 0) {
3          return;
4      }
5      console.log(n);
6      printNumber(n - 1)
7  }
8
9  printNumber(10);
```

Внимавайте с рекурсията =)

Функции

Аргументи на функции & arguments

- ✓ Аргументите на една функция могат да са много и от всякакъв тип
  - ✓ Number
  - ✓ Object
  - ✓ Array
  - ✓ etc..
  - ✓ Дори **Function**

```
1  function car(owner, color) {
2      console.log('Type of the owner parameter is:  ' + typeof(owner));
3      console.log('Type of the color parameter is: ' + typeof(color));
4
5      return owner.firstName + ' ' + owner.lastName +
6          ' drives ' + color + ' car.';
7  }
8
9  car({ firstName: 'Martin', lastName: 'Hristov' }, 'red');
```

Функции

Обектът *arguments*

- ✓ Обектът **arguments**
  - ✓ Всяка функция има такъв обект
  - ✓ Съдържа списък от подадените аргументи
  - ✓ Няма нужда да се подава като аргумент
  - ✓ Function overloading

```
function introduceYourself() {
    console.log(arguments);

    return 'Hello I am ' + arguments[0] + ' ' + arguments[1] +
        ' and I am ' + arguments[2] + ' years old from ' + arguments[3];
}

introduceYourself('Martin', 'Hristov', 20, 'Sofia');
```

Функции

Обхват на функция и променливи 1/2

- ✓ Променливите имат обхват
  - ✓ Обхвата на една променлива показва къде може да се достъпи тя
  - ✓ Глобални и локални променливи
  - ✓ Променлива декларирана без ключовата дума **var** става **глобална**

```
1  function fnScope() {
2      var a = 5; //local
3      b = 15 // global
4
5      if (true) {
6          var c = 20 // lives even after if scope
7      }
8
9      console.log(a);
10     console.log(b);
11     console.log(c);
12 }
```

Функции

Обхват на функция и променливи (hoisting) 2/2

```
console.log(noSuchVariable);
```

```
console.log(declaredLater);
var declaredLater = "Now it's defined!";
console.log(declaredLater);
```

```
1 isItHoisted();
2 definitionNotHoisted();
3
4 function isItHoisted() {
5     console.log("Yes!");
6 }
7 var definitionNotHoisted = function () {
8    console.log("Definition not hoisted!");
9 };
```

Функции

Overloading на функция

- Какво е overloading на функция?
  - Няколко функции с едно и също име, но с променлив брой аргументи
  - Function overloading в JavaScript **няма**
  - Всяка следваща функция със същото име като някоя минала презаписва старата
  - Но ... както почти всичко друго което липсва в JavaScript така и Overloading – а може да се **постигне**

- Съществува няколко вида *възможности за overloading*
  - Променлив брой аргументи
  - Променлив тип аргументи
  - Незадължителни аргументи

Функции от по-висок ред

- ✓ Какво е функция от по-висок ред?
  - ✓ Функция която може да приема като параметър друга функция
  - ✓ Функция която връща като резултат функция

```
// Higher order function (that returns another function as a result)
let times = function (x){
  return function (y){
    return x * y;
  }
}
```

Demo

Public

alert("Благодаря Ви !");

Контакти:

Георги Христанов
georgi.hristanov@sap.com

Димо Петров
dimo.petrov@sap.com

SAP Labs Bulgaria
София, бул.Цар Борис III, 136А