



HTML5 приложения за Android

Урок 02

Леон Анави
@leonanavi
leon@anavi.org

С подкрепата на:



Съдържание

- Въведение в JavaScript
- Въведение в Git
- Регистрация и работа с GitHub

JavaScript променливи

- Декларация:

```
var име_на_променлива;
```

- Името трябва да започва с буква или долна черта
- В името може да има букви, цифри или долни черти
- Не може да има интервал в името
- Не може името да съвпада с JS ключова дума
- Има разлика между малки и големи букви

```
var foo = 10;
```

```
var bar = "Hello World";
```

JavaScript примитивни типове

- Числа, например: 10, 3.14
- Булеви стойности, например: true, false
- Низове (т.е. string), например: "Hello World!"
- Променлива без стойност: null
- Недефинирана променлива: undefined

JavaScript масиви

- Структура за съхранение и достъп до множество еднотипни данни
- Наименования по правилата за променливи
- Създаване на масив:

```
var people3 = ["John", "Tom", "Wayne"];
```

- Алтернативно създаване на масив:

```
var people = new Array("John", "Tom", "Wayne");
```

- Друго алтернативно създаване на масив:

```
var people2 = new Array(2);
```

```
people2[0] = "John"; people2[1] = "Tom";
```

JavaScript методи за масиви

- Достъп до елемент от масива: `console.log(people[0]);`
- Определяне на броя на елементите в масив: `.length`
- Премахване на последния елемент: `.pop()`
- Добавяне на елемент в края: `.push()`
- Премахване на първия елемент: `.shift()`
- Добавяне на елемент в началото: `.unshift()`
- Изтриване на елемент по индекс: `delete people[1];`
- Добавяне/изтриване на елементи чрез `.splice()`
- Сортиране: `.sort()`

JavaScript условни конструкции

- Проверяват дали дадено условие е изпълнено

- If-else

```
if (условие) {  
    /* TODO */  
} else {  
    // TODO  
}
```

- Switch

```
switch(foo) {  
    case "John":  
        console.log("foo");  
        break;  
    default:  
        console.log("bar");  
}
```

- Въпросителна ?

```
var foo = (1 > bar) ? 10 : 20;
```

JavaScript цикли

- For
- While
- Do / While
- For / in, например:

```
var obj = {a:1, b:2, c:3};
```

```
for (var prop in obj) {  
  console.log("o." + prop + " = " + obj[prop]);  
}
```

```
// Output:  
// "o.a = 1"  
// "o.b = 2"  
// "o.c = 3"
```


JavaScript функции

- Блок от код, който извършва дадена задача и може да се извиква многократно
- Синтаксис:

```
function Name(параметър1, ..., параметърN) {  
    //код  
}
```

- Пример:

```
function plus(p1, p2) {  
    return p1 + p2;  
}  
console.log('1 + 2 = ' + plus(1, 2));
```

JavaScript Closures

- Анонимни функции, които се изпълняват сами и позволяват създаването на private променливи.
- Пример:

```
var add = (function () {  
    var counter = 0;  
    return function () {  
        return counter += 1;  
    }  
})();  
add();  
add();  
var final = add();  
console.log('final: '+final);
```

Резултат:

final: 3

JavaScript обекти

- Обектът съдържа свойства и методи (функции)
- Наименува се по правилата за имена на променливи
- Могат да се създават чрез функции или обектни инициализатори със синтаксис:
име_на_обект={свойство:стойност}
- Подробна информация:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

JavaScript обекти

```
var person = {  
  firstName:"John",  
  lastName:"Doe",  
  name : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};  
console.log("first name: " + person.firstName);  
console.log("last name: " + person["lastName"]);  
console.log("name: " + person.name());
```

JavaScript обекти

```
function createPerson() {  
  this.firstName = "John";  
  this.lastName = "Doe";  
  this.name = function() {  
    return this.firstName + " " + this.lastName;  
  }  
}
```

```
var person = new createPerson();  
console.log("first name: " + person.firstName);  
console.log("last name: " + person["lastName"]);  
console.log("name: " + person.name());
```

JSON

- Формат за представяне на JavaScript обекти
- Подходящ за сериализация на данни
- Удобен за четене и от хора
- Пример:

```
{"employees":  
  {"firstName":"John", "lastName":"Doe"},  
  {"firstName":"Anna", "lastName":"Smith"},  
  {"firstName":"Peter", "lastName":"Jones"}  
}]
```

JavaScript стриктен режим

"use strict";

- Въвежда ограничения като по този начин предпазва от грешки
- Може да се използва за целия скрипт или за отделна функция
- Не дава да се декларират глобални променливи без `var`
- За останалите ограничения разгледайте:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
var foo = 1;
```

```
if (foo = 2) { console.log('foo is 2'); }
```

```
var bar = 1;
```

```
if (bar == 2) { console.log('bar is 2'); }
```


JavaScript капани

Какво ще стане при изпълнения на следния код?

```
var foo = 1;
```

```
if (2 == foo) { console.log('foo is 2'); }
```

```
if (2 = foo) { console.log('foo is 2'); }
```

Резултат:

Uncaught Reference Error: Invalid left-hand side in assignment

Извод: добра професионална практика е при сравнения константите да са от ляво.

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
var foo = 1;  
var bar = true;  
if (foo == bar) {  
    console.log("foo == bar");  
}  
if (foo === bar) {  
    console.log("foo === bar");  
}
```

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
function printCity() {  
    var city = "Plovdiv";  
    console.log(city);  
}
```

```
var city = "Sofia";  
printCity();  
console.log(city);
```

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
function foo(bar) {  
    console.log("bar: "+bar);  
}  
  
foo();  
foo(10);
```

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
if (!0) { console.log("0 is false"); }
```

```
if (!false) { console.log("false is false"); }
```

```
if (!"") { console.log("'' is false"); }
```

```
if (undefined) { console.log('undefined is false'); }
```

```
if (!null) { console.log('null is false'); }
```

```
if (!0/0) { console.log('NaN is false'); }
```

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
function returnObject() {  
  return  
    { name: "John" };  
}  
var person = returnObject();  
console.log(person);
```

JavaScript капани

Какво ще стане при изпълнения на следния код?

```
var varA = {  
  prop: 37,  
  f: function() {  
    return this.prop;  
  }  
};  
var varB = {prop: 37};  
function f() {  
  return this.prop;  
}  
console.log(varA.f());  
console.log(this.varA.f());  
console.log(varB.f());
```

JavaScript капани

```
var foo = 0.1 * 0.2;
```

```
console.log("foo: " + foo);
```

Резултат:

```
foo: 0.020000000000000004
```

```
var bar = 1 + "2" + 3.50;
```

```
console.log("bar: " + bar);
```

Резултат:

```
bar: 123.5
```

```
var foobar = 1 + parseInt("2") + 3.50;
```

```
console.log("foobar: " + foobar);
```

Резултат:

```
foobar: 6.5
```


JavaScript Callback Hell

```
fs.readdir(source, function(err, files) {  
  if (err) {  
    console.log('Error finding files: ' + err)  
  } else {  
    files.forEach(function(filename, fileIndex) {  
      console.log(filename)  
      gm(source + filename).size(function(err, values) {  
        if (err) {  
          console.log('Error identifying file size: ' + err)  
        } else {  
          console.log(filename + ' : ' + values)  
          aspect = (values.width / values.height)  
          widths.forEach(function(width, widthIndex) {  
            height = Math.round(width / aspect)  
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)  
            this.resize(width, height).write(destination + 'w' + width + '_' + filename, function(err) {  
              if (err) console.log('Error writing file: ' + err)  
            })  
          }.bind(this))  
        }  
      })  
    })  
  }  
})
```

Идеи за избягване на: <http://callbackhell.com/>

JavaScript полезни връзки

- Mozilla Developer Network: JavaScript

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

- w3schools: JavaScript Tutorial

<http://www.w3schools.com/js/>

- Уеб дизайн, ФМИ към СУ

<http://www.fmi.uni-sofia.bg/econtent/web-design>

- ES6: The Good Parts, Борис Симандов, OpenFest 2014

<https://www.youtube.com/watch?v=-CirKivxCwo>

- JavaScript основи - видео уроци, Telerik Academy

<http://academy.telerik.com/student-courses/web-design-and-ui/javascript-fundamentals/video>

ПОЧИВКА

Git

- Система за контрол на версиите на кода
- Децентрализирана
- Open source
- Създадена от Линус Торвалдс за кода на Linux kernel



GitHub

- Онлайн система, базирана на Git
- Безплатни публични Git репота
- Допълнителни услуги като bug tracking, wiki и други
- Развива се и като социална мрежа за програмисти :)



Основни Git команди

- Сваляне локално на кода от Git репо с `clone`:
`git clone https://github.com/leon-anavi/html5-android-course`
- Сваляне на промени във вече съществуващо локално репо на проекти с `pull`:
`git pull`
- Смяна на Git клон (т.нар. branch) с `checkout`:
`git checkout foo`
- Отказ и изтриване на всички локални промени:
`git reset --hard HEAD`

Основни Git команди

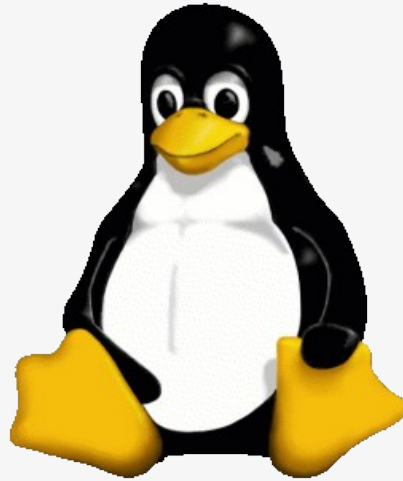
- Списък на променените файлове със **status**:
`git status`
- Преглеждане на промени по файл с **diff**:
`git diff foo.txt`
- Маркиране на файл за добавяне с **add**:
`git add bar.txt`
- Изтриване на файли с **remove**:
`git remove foobar.txt`

Основни Git команди

- Запазване на промените с локалното репо с `commit`:
`git commit -m "Информация за промените"`
- Предаване на запазените промени от локалното репо към други репота със `push`:
`git push`
- Показване на хеш на текущия `commit` в репо:
`git rev-parse HEAD`

Упражнения

- Регистрация в GitHub.
- Създаване на репо с open source лиценз и README.
- Напишете JavaScript функция, която изписва числата от 1 до 100. Всяко число, което се дели на 3 трябва да бъде заменено с Fizz. Всяко число, което се дели на 5 да се замени с Buzz. Всяко число, което се дели и на 3, и на 5, трябва да се замени с FizzBuzz.
- Публикувайте решението в GitHub.



**KEEP CALM
AND
SUPPORT
FOSS**