JavaScript

T · · Systems

Features

- Top level functions
  - Функции могут возвращать функции, которые возвращают функции и т.д.
- ECMAScript 5-6
  - У JS тоже есть версии
- Наследование через прототипы.
- Strings – во многом схожи с джавовскими. Можно использовать как одинарные кавычки, так и двойные.
- Массивы – можно использовать сразу как стек
- GIGO – Garbage In, Garbage Out

```
> 5 + true
6
> [] + true
"true"
> '' + {}
"[object Object]"
> // to be continued...
```

```
> var myStack = [];
  myStack.push(1);
  myStack.pop();
1
> myStack.pop();
undefined
```

T · · Systems

Features

- Scope
  - If блок не создает область видимости.

```
> var t = 'string in global context';
  function testIf(value){
  if(value === 1) {
   var t = 'string in if block';
  }
  console.log(t);
  }
  testIf(1);
  string in if block
```

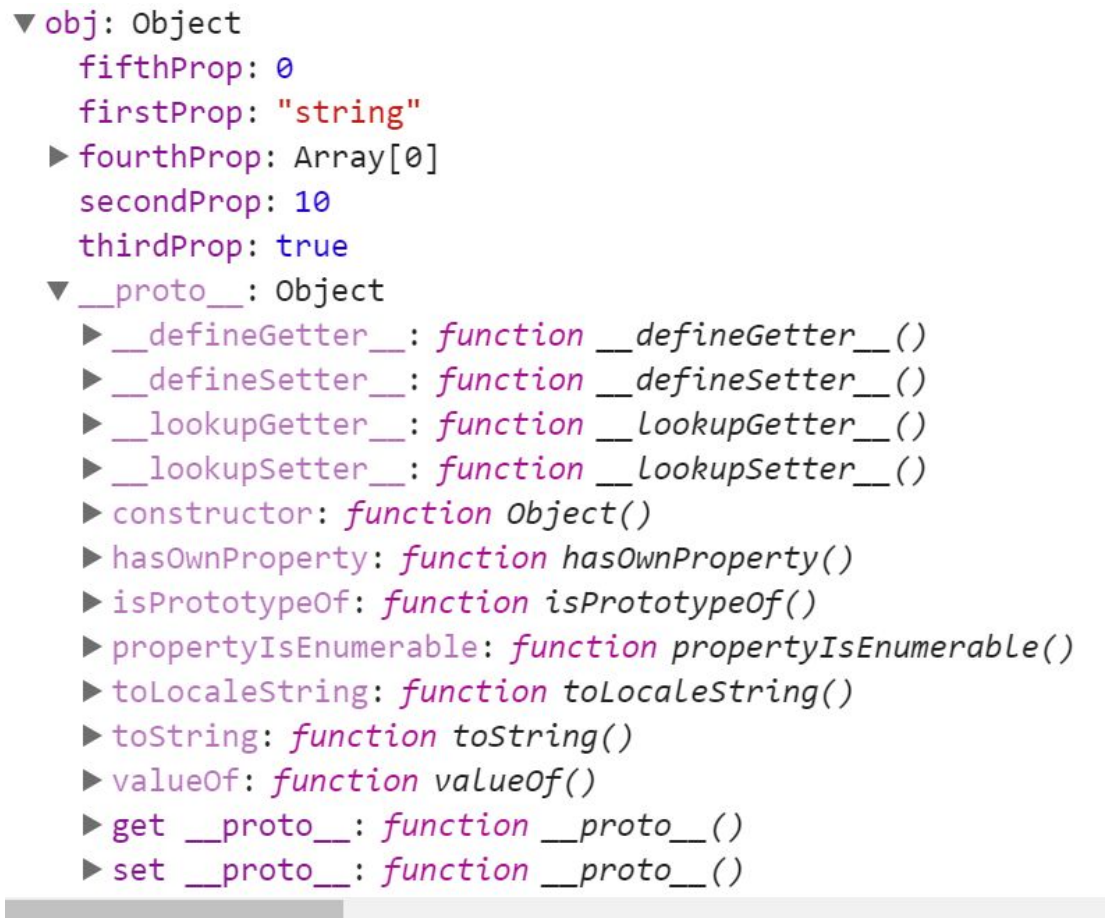
  - Цикл не создает область видимости

```
> var t = 'string in global context';
  function testForScope(){
  for(var i = 0; i < 5; i++){}
  console.log(i);
  }
  testForScope();
  5
```

T · · Systems

Features

- Различные проперти глобального объекта – constructor, toString и т.п.



- Hoisting
  - Загрузка функций и переменных в память происходит на первом этапе, поэтому можно использовать переменные до их объявления

  Считается не очень хорошей практикой

```
> notExistsYet();

  function notExistsYet(){console.log(myVar);}

  var myVar = 10;
  10
```

- Не работает при использовании strict mode

Features

- IIFE – Immediately invoked function expression

```
> // will be called immediately
  (
    function(arg){
    var inner = 'inner';
    return arg + ' and ' + inner + ' are returned from IIFE'}
  )('outer argument')
"outer argument and inner are returned from IIFE"
> inner
▶ Uncaught ReferenceError: inner is not defined(…)
```

T-Systems

Features

- Замыкания
  - Использование «потерянного» контекста

```
> var closures = [];
  function generateRandomString() {
  return 'Generated ' + Math.floor(Math.random() * 10);
  }
  function addClosure(){
  var local = generateRandomString();
  closures.push(function(){ console.log(local + ' is not lost!') })
  }
<· undefined
> addClosure();
  addClosure();
<· undefined
> closures[0]();
  closures[0]();
  closures[1]();
  closures[0](); // context differs
  Generated 9 is not lost!
  Generated 9 is not lost!
  Generated 4 is not lost!
  Generated 9 is not lost!
<· undefined
> local; // no such value, undefined
✖ ▶ Uncaught ReferenceError: local is not defined(…)
```

T · · Systems

Features

- Еще один пример замыкания. Что будет на выходе?

```
> var closures = [];
  function generateRandomString() {
  return 'Generated ' + Math.floor(Math.random() * 10);
  }
  function addManyClosures(){
  var local = generateRandomString();
  for(var i = 0; i < 3; i++){
  closures.push(function(){ console.log('local: ' + local + ', i: ' + i) })
  }
  }
<· undefined
> addManyClosures();
  // guess the output?
  closures[0]();
  closures[1]();
  closures[2]();
```

- На выходе у нас:

```
local: Generated 7, i: 3
local: Generated 7, i: 3
local: Generated 7, i: 3
```

Features

- Как сделать так, чтобы переменная i каждый раз была такой, какой мы хотим ее видеть? Нужно поместить ее значение в новый контекст (выделено красным).

```
> var closures = [];
function generateRandomString() {
return 'Generated ' + Math.floor(Math.random() * 10);
}
function addManyClosures(){
var local = generateRandomString();
for(var i = 0; i < 3; i++){
closures.push(
(function(i){
    return function(){
            console.log('local: ' + local + ', i: ' + i) }
          })(i))
    }
}
addManyClosures();
<· undefined
> closures[0]();
closures[1]();
closures[2]();
local: Generated 7, i: 0
local: Generated 7, i: 1
local: Generated 7, i: 2
```

T · · Systems

Features

- Способы создания объекта
  - Конструктороподобный

```
function Person(firstName, lastName){
this.firstName = firstName;
this.lastName = lastName;
}
undefined
new Person()
Person {firstName: undefined, lastName: undefined}
// careful, code below returns underfined!
Person()
undefined
```

  - Object.create(), ему даем (опционально) предка

```
var p = 'never called';
var newObj = Object.create({p:5});
newObj.p;
5
```

  - Просто пишем {} (Object literal notation)

```
var obj = {
firstProp : 'string',
secondProp : 10,
thirdProp : true
};
// we can set props also like this:
obj.fourthProp = [];
obj["fifthProp"] = 0;
0
obj
▶ Object {firstProp: "string", secondProp: 10, thirdProp:
true, fourthProp: Array[0], fifthProp: 0}
```

T · · Systems

Features

- «Асинхронная» работа – таймауты, коллбэки и т.п.
  - Работа с event-queue
- Strict mode (с ES5)

```
> p = 5;
<· 5
> p
<· 5
> 'use strict'
  b = 7;
✕ ▶ Uncaught ReferenceError: b is not defined(…)
>
```

T · · Systems

Features

- Функция – особый тип объекта, который имеет у себя «вызываемый» код
  - К ней можно добавлять проперти и использовать как объект

```
> function emptyFunc(){}
< undefined
> emptyFunc.a = 10; emptyFunc.b = 'String!';
< "String!"
> emptyFunc.printIt = function(){return a + ' ' + this.b;}
< function (){return a + ' ' + this.b;}
> emptyFunc.printIt()
✕ ▶ Uncaught ReferenceError: a is not defined(…)
> emptyFunc.printIt = function(){return this.a + ' ' + this.b;}
< function (){return this.a + ' ' + this.b;}
> emptyFunc.printIt()
< "10 String!"
>
```

Features

- Псевдомассив arguments
  - Дает текущие аргументы, даже если их больше чем заявлено

```
> function printArgs() {
  var result = '';
  for(var idx = 0; idx < arguments.length; idx++){
   result += arguments[idx];
  }
  return result;
  }
< undefined
> printArgs(1, 2, 3, {}, 'string')
< "123[object Object]string"
```

- Если аргументов меньше чем надо, то недостающие берутся как undefined

```
> function takeSeveralArgs(first, second, third)
  {
    return first + ' ' + second + ' ' + third;
  }
< undefined
> takeSeveralArgs(1)
< "1 undefined undefined"
>
```

Features

- Точки с запятой, особенности парсера – лучше ставить вручную

```
> function tricky(){
  return
   {p:10}
  }
undefined
> var value = tricky();
undefined
> value.p
Uncaught TypeError: Cannot read property 'p' of undefined(…)
```

- Триксы и приколы
  - void 0
  - default value
  - typeOf – самое веселое

```
> var check = undefined == void 0;
  check;
true
```

```
> function returnArgOrDefault(arg){
  return arg || 200; // 'default' value
  }
  returnArgOrDefault();
200
```

T · · Systems

Features

- Вспомогательные методы – call, bind, apply
- Bind – создание нового объекта с зафиксированным параметром (-ами)

```
> function multiply(multiplier, value){
    return multiplier * value;
  };
  multiply(10,2);
<· 20
> var multByTwo = multiply.bind(multiply,2);
<· undefined
> multByTwo(2);
<· 4
```

- apply и call – указать, на что будет указывать this и вызвать с аргументами (массив)
- call аналогично, только аргументы через запятую

```
> function printThis(){return this;}
<· undefined
> printThis()
<· ▶ Window {external: Object, chrome: Object
     SpeechSynthesis…}
> printThis.apply(
  // this will point to new object
  {first:'string', second:5}
  )
<· Object {first: "string", second: 5}
```

T · · Systems

Features

- JSON != object notation
- Для конвертации одного в другое есть удобные функции у объекта JSON : stringify и parse

```
> var myNotJsonObj = {first : 1, second : 'second'};
< undefined
> JSON.stringify(myNotJsonObj);
< "{"first":1,"second":"second"}"
> JSON.parse('{"first":1,"second":"second"}');
< Object {first: 1, second: "second"}
```

Useful links

- https://learn.javascript.ru/ Различные обучающие материалы
- https://www.udemy.com/understand-javascript/ Интересные видеокурсы по JS (осторожно, рутрекер)
- https://habrahabr.ru/post/305900/ Новое в ES6
- http://www.w3schools.com/js/default.asp туториал
- Ctrl + Shift + J dev tools в хроме