JAVASCRIPT

ООП. Наследование. Цепочка прототипов

OBJECT

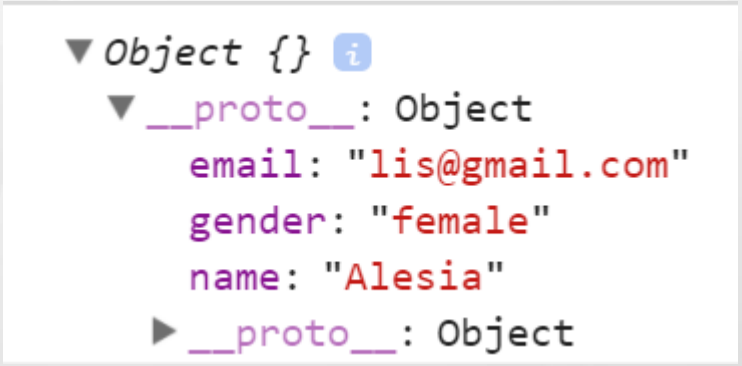
Цепочки прототипов

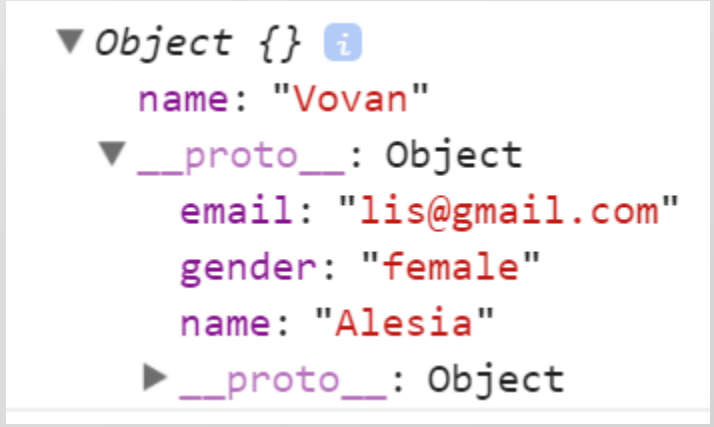
```
var o = {a: 1};
// o ---> Object.prototype ---> null


var a = ["yo", "whadup", "?"];
// a ---> Array.prototype ---> Object.prototype ---> null


function f(){
  return 2;
}
// f ---> Function.prototype ---> Object.prototype ---> null
```

OBJECT

Создание объектов

```
var user = {
    name: 'Alesia',
    email: 'lis@gmail.com',
    gender: 'female'
};
// user ---> Object.prototype ---> null

var admin = Object.create(user);
// admin ---> user ---> Object.prototype ---> null

console.log(admin);
console.log(admin.hasOwnProperty('name')); // false

admin.name = 'Vovan'; // own property
console.log(admin.hasOwnProperty('name')); //true

delete admin.name;
console.log(admin.hasOwnProperty('name')); //false
```





OBJECT

```
var user = {
    name: 'Alesia',
    email: 'lis@gmail.com',
    gender: 'female'
};

var admin = Object.create(user);

console.log('name' in admin); // true
console.log(admin.age); // undefined
console.log('age' in admin); // false

admin.age = 25;
console.log('age' in admin); // true
admin.age = undefined;
console.log('age' in admin); // true
```

OBJECT

Методы как свойства объекта

```
var user = {
    name: 'Alesia',
    email: 'lis@gmail.com',
    gender: 'female',
    login: function (){
        console.log(user.name + ' is logged in');
    }
};

user.login();
```

```
var login = function (){
    console.log(user.name + ' is logged in');
};

user.login = login;
user.login();
```

OBJECT

Ключевое слово - this

```
var login = function (){
    // this указывает на тот объект (scope, context),
    // у которого вызвана функция
    console.log(this.name + ' is logged in');
};

var user = {
    name: 'Alesia',
    email: 'lis@gmail.com',
    gender: 'female',
    login: login
};

user.login(); // Alesia is logged in

var secondUser = {
    name: 'Jone Doe',
    login: login
};
secondUser.login(); // Jone Doe is logged in
```

```
var login = function (){
    // this указывает на тот объект (scope, context),
    // у которого вызвана функция
    console.log(this + ' is logged in');
};

login(); // вызвана в глобальном контексте window
// значит внутри функции this будет указывать на window

console.log(this); // тоже window
```

OBJECT

Методы call и apply

```
var login = function (){
    console.log(this.name + ' is logged in');
};

var user = {
    name: 'Alesia',
    login: login
};

var secondUser = {
    name: 'Jone Doe',
    login: login
};

user.login.call(secondUser);
//Jone Doe is logged in
user.login.apply(secondUser);
// Jone Doe is logged in
```

```
var login = function (greeting){
    console.log(this.name + ' is logged in. ' + greeting);
};

var user = {
    name: 'Alesia',
    login: login
};

var secondUser = {
    name: 'Jone Doe',
    login: login
};

user.login.call(secondUser, 'Hi there!');
//Jone Doe is logged in. Hi there!
user.login.apply(secondUser, ['Good Morning']);
// Jone Doe is logged in. Good Morning
```

OBJECT

Метод bind

```
var userBound = login.bind(user);
// bind не вызывает функцию
// а возвращает функцию с зафиксированным контекстом

userBound();
```

OBJECT

Свойства. Аксессоры

```
var user = {
    name: 'Alesia',
    _age: null,
    get age(){
        return this._age;
    },
    set age(value){
        this._age = value;
    }
};
```

```
Object.defineProperty(user, 'isAdmin', {
    value: true,
    writable: false,
    enumerable: false,
    configurable: false
});

Object.defineProperties();

Object.isExtensible(user);
Object.preventExtensions(user);
```

```
console.log(Object.getOwnPropertyDescriptor(user, 'name'));
console.log(Object.getOwnPropertyDescriptor(user, 'age'));
```

OBJECT

Прототипы

```
var user = {
    name: 'Alesia',
    email: 'lis@gmail.com',
    gender: 'female',
    login: function () {
        console.log(this.name + ' is logged in');
    }
};
```

```
var User = {
    constructor: function (name, email, gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
        return this;
    },
    logIn: function () {
        console.log(this.name + ' is logged in');
    },
    logOut: function () {
        console.log(this.name + ' is logged out');
    }
};

var user_01 = Object.create(User).constructor('Lisa', 'test@test.by', 'female')

console.log(user_01);
```

OBJECT

```
var User = {
    constructor: function (name, email, gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
        return this;
    },
    logIn: function () {
        console.log(this.name + ' is logged in');
    },
    logOut: function () {
        console.log(this.name + ' is logged out');
    }
};
```

```
var AdminUser = Object.create(User);
AdminUser.constructor = function (name, email, gender, department) {
    User.constructor.apply(this, arguments);
    this.isAdmin = true;
    this.department = department;
    return this;
};
```

```
var admin = Object.create(AdminUser).constructor('Alex', 'admin@test.by', 'male', 'sales');
admin.logOut();
```

OBJECT

Конструкторы

```
function User(name, email, gender) {
    this.name = name;
    this.email = email;
    this.gender = gender;
    this.logIn = function () {
        console.log(this.name + ' is logged in');
    };
}

User.prototype.logOut = function () {
    console.log(this.name + ' is logged out');
};


var firstUser = new User('Lisa', 'test@test.by', 'female');
firstUser.logOut();

firstUser instanceof User
// true
```

OBJECT

Конструкторы

```
function AdminUser (name, email, gender, department) {
    User.apply(this, arguments);
    this.isAdmin = true;
    this.department = department;
    return this;
};

AdminUser.prototype = Object.create(User.prototype);
AdminUser.prototype.constructor = AdminUser;

var admin = new AdminUser('Alex', 'admin@test.by', 'male', 'sales');

admin instanceof User // true
admin instanceof AdminUser  // true
```