

JS

Objects and OOP techniques

Объекты


- объект – это коллекция именованных значений (свойств)
- св-ва объекта могут содержать любой тип данных, включая массивы, функции и другие объекты
- объекты не статичны, новые св-ва и методы могут быть добавлены в любой момент
- доступ к свойствам объекта из его же методов осуществляется с помощью ключевого слова `this`
- передаются в функции по ссылке
- все объекты наследуются от базового – `Object`
- объекты бывают 3х категорий:
 - native objects
 - host objects
 - user-defined

Создание объектов

```
var order = {  
  price : 3.99,  
  user  : null,  
  goods : [],  
  add    : function(val){ // метод  
    this.price = this.price + val;  
  }  
};
```

Создание объектов

```
var order = {  
  price : 3.99,  
  user   : null,  
  goods  : [],  
  add    : function(val){ // метод  
    this.price = this.price + val;  
  }  
};
```



если тут окажется ,
то все будет хорошо везде,
кроме IE

Создание объектов

```
var order = new Object(),  
    date  = Date(),  
    re    = new RegExp("js", "ig"),  
    arr   = new Array(5);
```

Создание объектов

```
var order = Object.create(null),  
    obj1  = Object.create({protoVal : 123}),  
    obj2  = Object.create(  
        {protoVal      : 123},  
        {instanceVal  : 456}  
    );
```

Доступ к св-вам объектов, изменение

```
var order = {  
    price : 3.99,  
    user  : null,  
    goods : [],  
    add   : function(val){ // метод  
        this.price = this.price + val;  
    }  
};
```

```
order.price;  
order.add(28);
```

```
order["price"];  
order["add"](12);
```

```
order["some new property"] = "property with a strange key";
```

Доступ к св-вам объектов, изменение

```
var order = {  
    price : 3.99,  
    user  : null,  
    goods : [],  
    add   : function(val){ // метод  
        this.price = this.price + val;  
    }  
};
```

```
order.goods = null;  
order.goods.length; // Error
```

```
if(order && order.goods && order.goods.length)  
    console.log(order.goods.length); // NO Error
```


Удаление св-в объектов

```
var order = {  
  price : 3.99,  
  user  : null,  
  goods : [],  
  add    : function(val){ // метод  
    this.price = this.price + val;  
  }  
};
```

```
delete order.goods;  
delete order["super property"];
```

```
console.log(order.goods); // undefined
```

- удаляет не значение, а само свойство
- удаляет только собственные свойства
- возвращает булево значение

Общие св-ва объектов

```
var order = {  
    price : 3.99,  
    user  : null,  
    goods : [],  
    add    : function(val){ // метод  
        this.price = this.price + val;  
    }  
};
```

```
order.toString();           // => ?  
order.toLocaleString();    // => ?  
order.valueOf();           // => ?  
order.hasOwnProperty();    // => ?  
order.toJSON();            // => ?
```

Функции-конструкторы

- конструкторы – это функции, которые создают объекты
- статические свойства или методы добавляются к конструктору
- все свойства и методы объекта - public
- получить ссылку на функцию-конструктор объекта можно с помощью свойства `constructor`
- определить принадлежность объекта к конструктору можно с помощью `instanceof`
- чтобы создать новый экземпляр вызывайте конструктор с оператором `new`

Функции-конструкторы

```
function Car (brand, VIN) {  
  
    this.brand    = brand;  
    this.VIN      = VIN;  
    this.getBrand = function () {  
        return this.brand;  
    }  
}
```

```
var lada = new Car("Lada", "A120P593K87TZC"),  
    merc = new Car("Mercedes", "M1450P59K8PRHC");
```

```
var bmw = new Object();
```

```
Car.call(bmw, "BMW", "P27RYP59PRNT8Y");
```

Функции-конструкторы | избыточность

```
function Car (brand, VIN) {  
  
    this.brand    = brand;  
    this.VIN      = VIN;  
    this.getBrand = getBrand;  
}
```

```
function getBrand() {  
    return this.brand;  
}
```

```
var lada = new Car("Lada", "A120P593K87TZC");
```

```
lada.brand;           // => Lada  
lada.getBrand();      // => Lada
```

Функции-конструкторы | приватные свойства

```
function Car (brand, VIN) {  
  
    var brand = brand,  
        VIN    = VIN;  
  
    return {  
        getBrand : function () {  
            return brand;  
        }  
    }  
}
```

```
var lada = new Car("Lada", "A120P593K87TZC");
```

```
lada.brand;          // => undefined  
lada.getBrand();     // => Lada
```

Функции-конструкторы | статические свойства

```
function Car (brand, VIN) {  
  
    var brand = brand,  
        VIN    = VIN;  
  
    return {  
        getBrand : function () {  
            return brand;  
        }  
    }  
}  
  
Car.beep = function () { alert("Beep!"); }  
  
Car.beep(); // => Beep!
```

Абстрактные конструкторы

```
function Car () {  
    throw new Error("Car is an abstract constructor");  
}
```

```
Car.beep = function (){ alert("Beep!"); }
```

```
Car.beep(); // => Beep!
```


Прототип

- каждая функция обладает свойством prototype
- все свойства и методы, определенные в prototype, будут общими для всех экземпляров объекта, которому принадлежит прототип
- prototype — механизм, через который реализуется наследование

ПРОТОТИП

```
var brand = "Lada",  
    VIN    = "A120P593K87TZC";
```

```
function Car () { }
```

```
Car.prototype.brand    = brand;  
Car.prototype.VIN      = VIN;  
Car.prototype.getBrand = function() { return this.brand; }
```

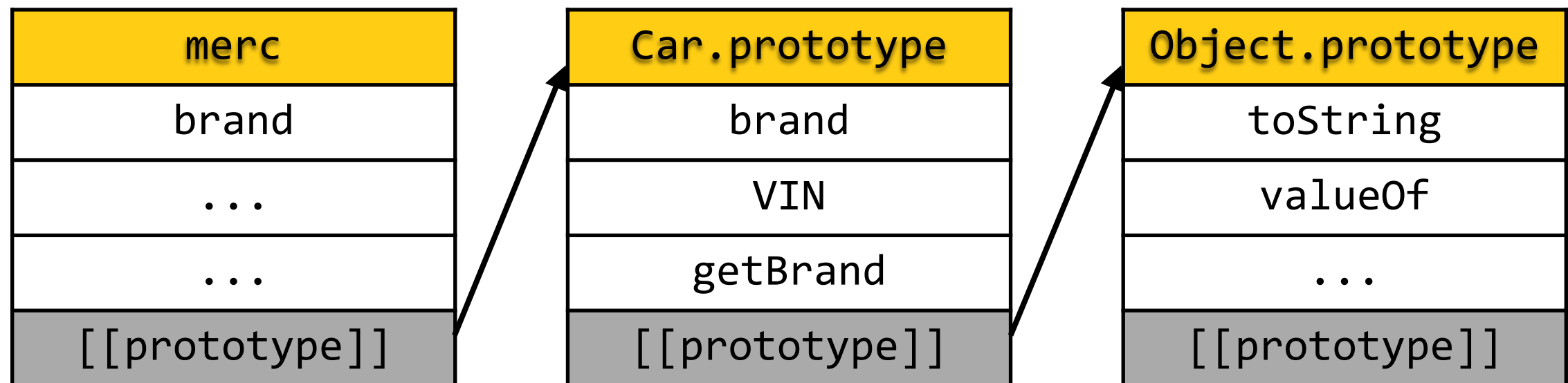
```
var lada = new Car();
```

```
lada.brand;           // => Lada  
lada.getBrand();      // => Lada
```

```
var merc = new Car();
```

```
merc.getBrand();       // => Lada  
merc.brand = "Mercedes";  
merc.getBrand();       // => Mercedes
```

Цепочка прототипов



Функции-конструкторы + прототипы

```
function Car (brand, VIN) {  
    this.brand    = brand;  
    this.VIN      = VIN;  
}
```

```
Car.prototype.getBrand = function() {  
    return this.brand;  
}
```

```
var lada = new Car("Lada", "A120P593K87TZC");
```

```
lada.brand;           // => Lada  
lada.getBrand();      // => Lada
```

Наследование

```
function Car (brand, wheels) {  
    this.brand    = brand;  
    this.wheels   = wheels || 4;  
}
```

```
Car.prototype.getBrand = function() {  
    return this.brand;  
}
```

```
function Truck (brand, wheels, trailerAttached) {  
    this.trailerAttached = trailerAttached;  
}
```

```
Truck.prototype.attachTrailer = function() {  
    return this.trailerAttached = true;  
}
```

Наследование

```
function Truck (brand, wheels, trailerAttached) {  
    Car.call(this, brand, wheels || 8);  
    this.trailerAttached = trailerAttached;  
}  
  
Truck.prototype = new Car();  
Truck.prototype.constructor = Truck;  
  
Truck.prototype.attachTrailer = function() {  
    return this.trailerAttached = true;  
}
```

Наследование

```
function extend (Parent, Child) {  
    var temp = new Function();  
  
    temp.prototype = Parent.prototype;  
  
    Child.prototype          = new temp();  
    Child.prototype.constructor = Child;  
    Child.super              = Parent.prototype;  
}
```