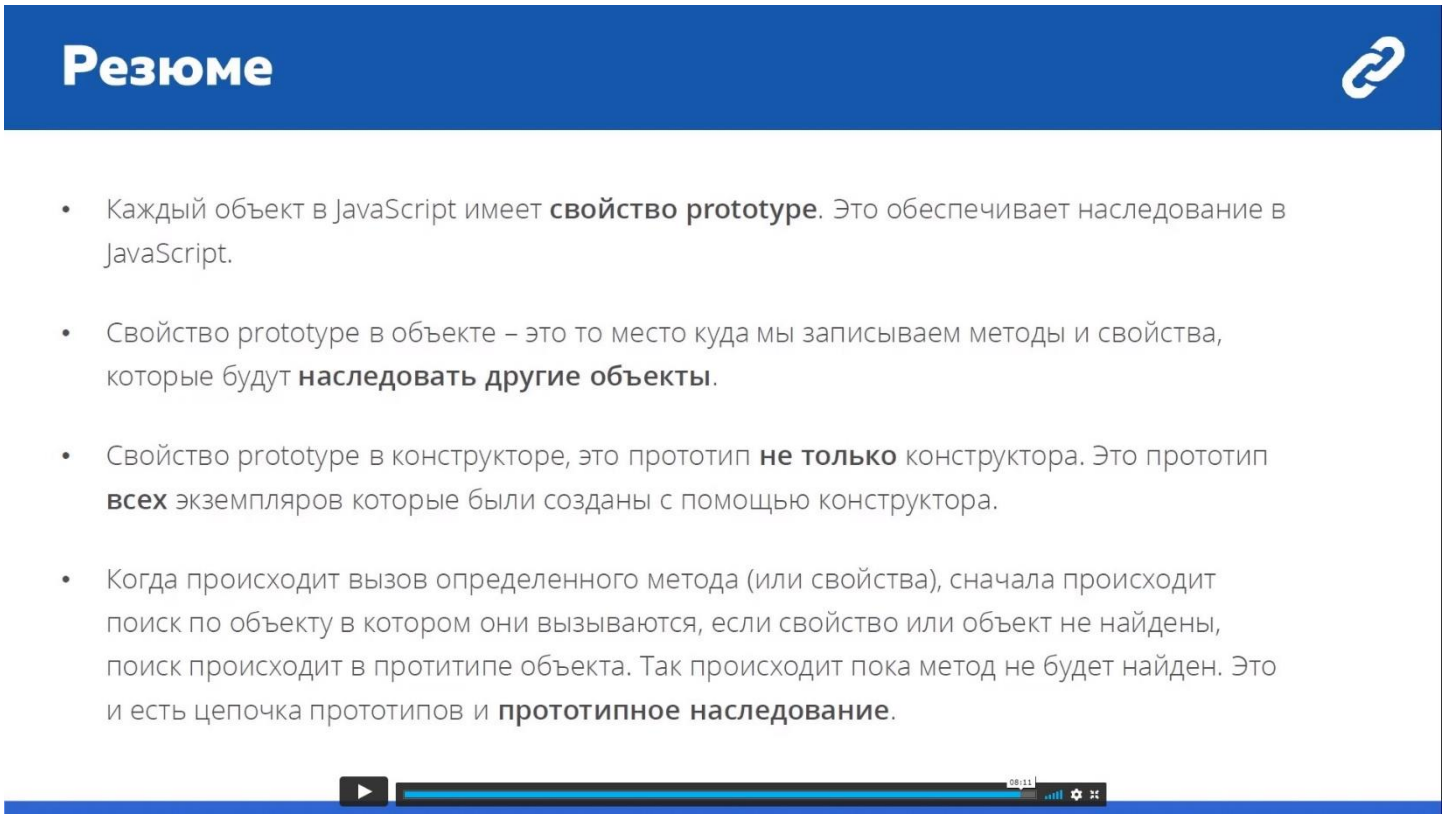
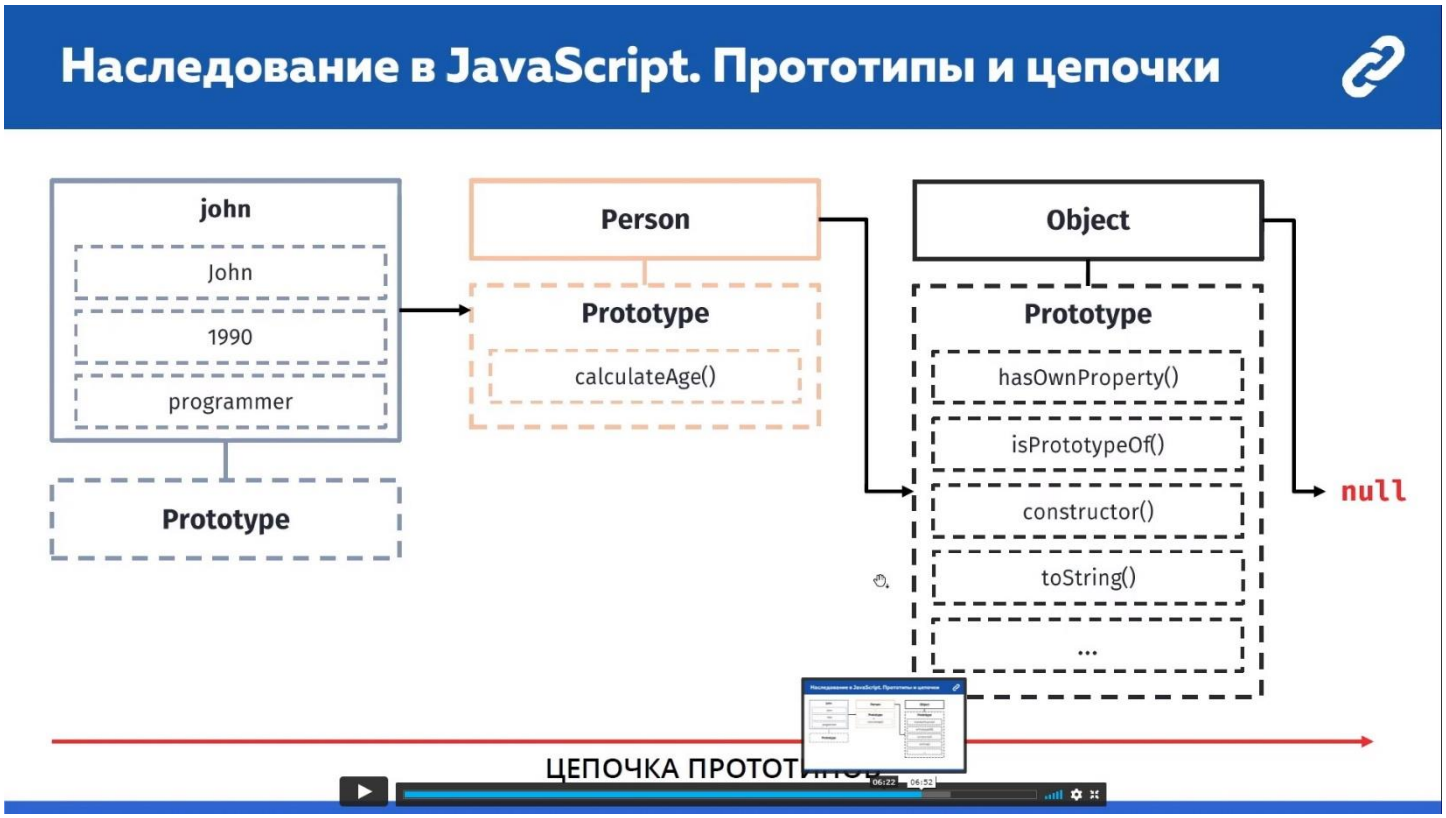
Прототипы. Объекты и примитивы





Функция-конструктор

Функции-конструкторы являются обычными функциями. Но есть два соглашения:

- Имя функции-конструктора должно начинаться с большой буквы.
- Функция-конструктор должна вызываться при помощи оператора "new"

Например:

```
1. function User(name) {
  this.name = name;
  this.isAdmin = false;
}
```

```
let user = new User("Вася");
```

```
2. var Person = function (name, year, job) {
  this.name = name;
  this.year = year;
  this.job = job;
  this.calculateAge = function () {
    var age = 2020 - this.year;
    console.log(age);
  };
};
var john = new Person("John", 1990, "teacher");
```

Метод в прототипе

Можем описать **метод** calculateAge в **прототипе**:

```
Person.prototype.calculateAge = function () {
  var age = 2020 - this.year;
  console.log(age);
}
jane.calculateAge();  //30
```

Свойство в прототипе

Можем аналогично описать **свойство** в **прототипе**:

```
Person.prototype.city = "Moscow";
console.log(john.city);// Moscow
```

obj.hasOwnProperty(key

obj.hasOwnProperty(key): возвращает true, если у obj есть **собственное**, не унаследованное, свойство с именем key.

```
console.log(john.hasOwnProperty("year")); //true
console.log(john.hasOwnProperty("city")); //false
```

.__proto__

```
console.log(john.__proto__ === Person.prototype);
```

instanceof

Можно проверить - является ли объект сущностью класса или конструктора:

```
console.log(john instanceof Person);
```

Object.create()

Другой способ построения объекта на основе прототипа (1-ый способ - с помощью функции-конструктора).

Object.create(proto, [descriptors]) – создаёт пустой объект со свойством **[[Prototype]],** указанным как **proto**, и необязательными дескрипторами свойств **descriptors** (в том числе **значениями** свойств).

```
var personProto = {
  calculateAge: function () {
    var age = 2020 - this.year;
    console.log(age);
  },
};

var john = Object.create(personProto);

var mark = Object.create(personProto, {
  name: { value: "Mark" },
  year: { value: 1988 },
  job: { value: "programmer" },
});
console.log(mark);
// Вывод:
```

{name: 'Mark', year: 1988, job: 'programmer'}
job: "programmer"
name: "Mark"
year: 1988
[[Prototype]]: Object
calculateAge: *ƒ ()*
[[Prototype]]: Object

Примитивы и объекты

Примитивы - содержат в себе значения
Объекты - не содержат в себе значения, они содержат в себе **ссылку** на место в памяти, где расположен данный объект.

```
// Пример с примитивами
var a = 20;
var b = a; // 20
a = 50; // 50

console.log(b); //20

// Пример с объектами
var object1 = {
  name: "John",
  age: 25,
};

var object2 = object1;
object1.age = 30;
console.log(object2.age); //30

object2.name = "Mark";
console.log(object1.name); //mark

// Пример с функциями
var number = 30;
var object = {
  name: "John",
  city: "Moscow",
};

function change(a, b) {
  a = 50;
  b.city = "Kiev";
}

change(number, object);
console.log(number); //30 !!!
console.log(object.city); //Kiev

// Пример с массивами
var arr1 = [1, 2, 3];

var arr2 = arr1;
arr2[2] = 10;

console.log(arr1); //[1,2,10]
```