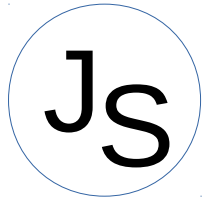
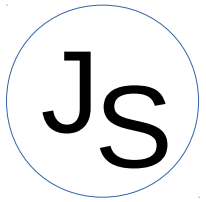
Классы и прототипы



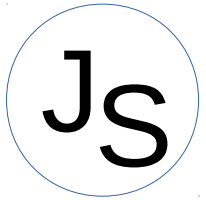
- Конструкторы
- Прототип и наследование свойств
- Эмуляция классов JavaScript
- Общие методы
- Наследование классов

Конструктор




```
function Rectangle(w, h){
   this.width = w;
   this.height = h;
}

var rect1 = new Rectangle(2, 4);
var rect2 = new Rectangle(8.5, 11);
```

Прототип



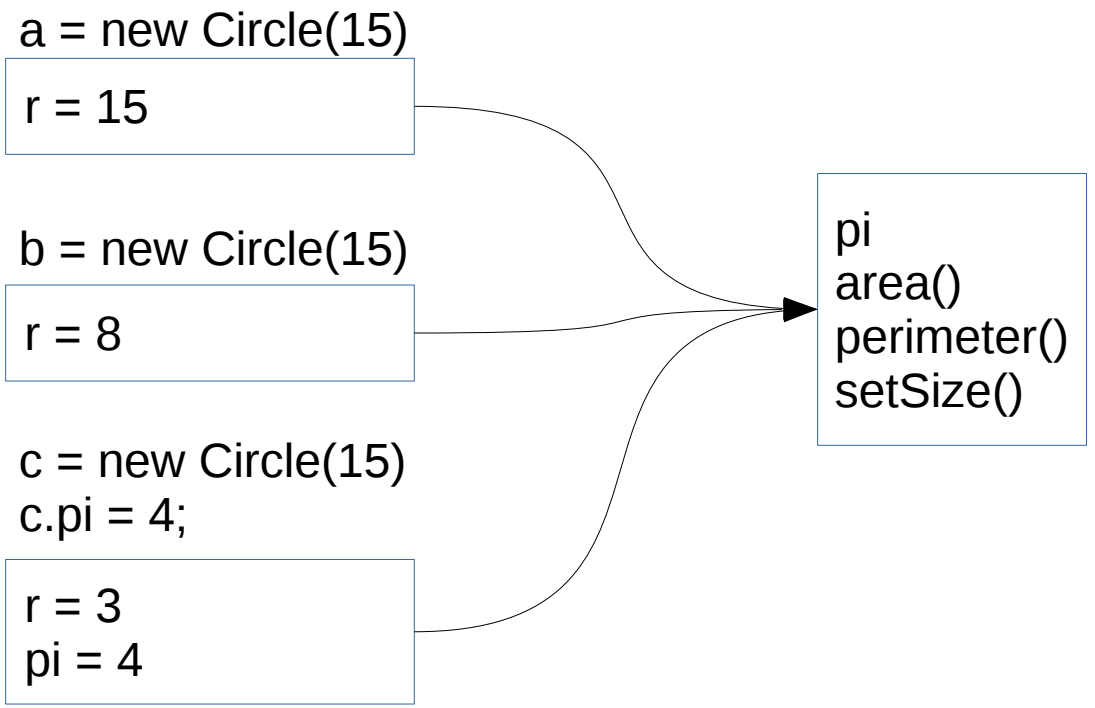

```
var pi = 3.1415926;
function c_perimeter(){return 2*pi*this.radius;}
function c_setRadius(r){this.radius=r;}

function Circle(r){

   this.radius = r;

   this.perimeter = c_perimeter;
   this.setRadius = c_setRadius;
}

var r = new Circle(3);
r.setRadius(10);
var p = r.perimeter();
```

Прототип

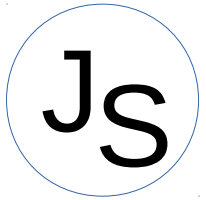
```
function Circle(r){
   this.radius = r;
}

new Circle(0);                          //создать прототип
Circle.prototype.pi = 3.1415926;
Circle.prototype.perimeter = () => 2*pi*this.radius;
Circle.prototype.setRadius = (r) => {this.radius=r;}

var r = new Circle(3);
r.setRadius(10);
var p = r.perimeter();
```

```
String.prototype.endsWith = function(c) {
    return (c == this.charAt(this.length-1))
}
'Привет, мир'.endsWith('р');          //true
```

Наследование свойств



Эмуляция классов

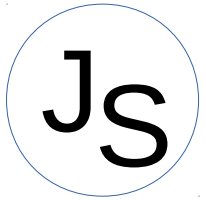


Классов нет, эмулируются конструкторами и прототипами.
Класс и экземпляр (instance).

c = new Circle();

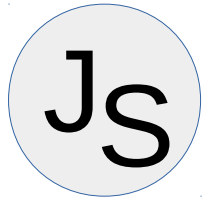
свойство экземпляра	c.r	прототип
метод экземпляра	c.area()	прототип
свойство класса	Circle.PI	класс
метод класса	Circle.max (c1, c2)	класс

Наследование классов



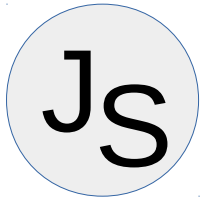
```
function Disk(r){
   this.radius = r;
}

Disk.prototype = new Circle(0);

Disk.prototype.area = function() {
   return Circle.PI * this.r * this.r;
}

Disk.prototype.constructor = Disk;
                                  // восстановили
```

Классы (ES6) - продолжение




```
var aggregation = (baseClass, ...mixins) => {
    let base = class _Combined extends baseClass {
        ...
    }
    ...
    return base
}

class Colored {...}
class ZCoord {...}
class Shape {...}

class Rectangle extends
   aggregation(Shape, Colored, ZCoord) {}

var rect = new Rectangle(7, 42)
rect.z     = 1000
rect.color = "red"
console.log(rect.x, rect.y, rect.z, rect.color)
```

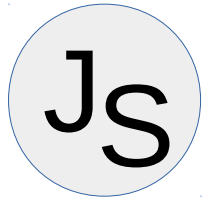
Классы (ES6) - продолжение



Доступ к базовому классу

```
class Shape {
    ...
    toString () {return `Shape(${this.id})`}
}
class Rectangle extends Shape {
    constructor (id, x, y, width, height) {
        super(id, x, y)
        ...
    }
    toString () {return "Rectangle > " + super.toString()}
}
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        …
    }
    toString () {return "Circle > " + super.toString()}
}
```

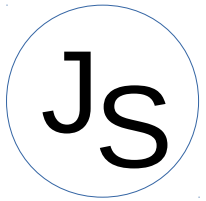
Классы (ES6) - продолжение



Статические члены класса

```
class Rectangle extends Shape {
    …
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
class Circle extends Shape {
    …
    static defaultCircle () {
        return new Circle("default", 0, 0, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
var defCircle    = Circle.defaultCircle()
```

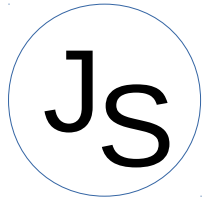
Классы (ES6) - продолжение



Геттеры и сеттеры

```
class Rectangle {
  constructor (width, height) {
    this._width  = width
    this._height = height
  }
  set width  (width)  { this._width = width }
  get width  ()       { return this._width  }
  set height (height) { this._height = height }
  get height ()       { return this._height }
  get area   ()       { return this._width*this._height }
}

var r = new Rectangle(50, 20)
r.area === 1000
```

Литералы объектов (ES6)




```
obj = { x, y }     // { x: x, y: y }
```

```
let obj = {
    foo: "bar",
    [ "baz" + quux() ]: 42
}
```

```
obj = {
    foo (a, b) {...},
    bar (x, y) {...},
    *quux (x, y) {...}
}
```

Классы и прототипы

- Конструкторы
- Прототип и наследование свойств
- Эмуляция классов JavaScript
- Общие методы
- Наследование классов

Конструктор

```
function Rectangle(w, h){
   this.width = w;
   this.height = h;
}

var rect1 = new Rectangle(2, 4);
var rect2 = new Rectangle(8.5, 11);
```

Прототип

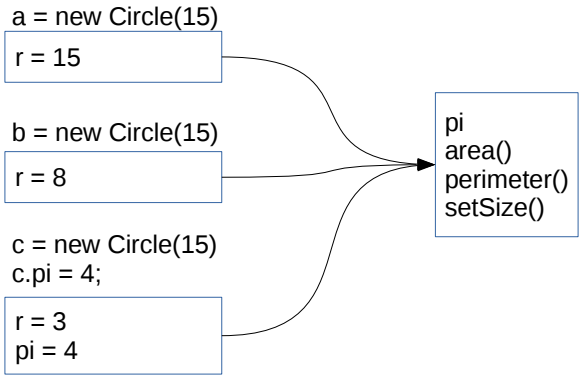
```
var pi = 3.1415926;
function c_perimeter(){return 2*pi*this.radius;}
function c_setRadius(r){this.radius=r;}

function Circle(r){

   this.radius = r;

   this.perimeter = c_perimeter;
   this.setRadius = c_setRadius;
}

var r = new Circle(3);
r.setRadius(10);
var p = r.perimeter();
```

Прототип

```
function Circle(r){
   this.radius = r;
}

new Circle(0);                        //создать прототип
Circle.prototype.pi = 3.1415926;
Circle.prototype.perimeter = () => 2*pi*this.radius;
Circle.prototype.setRadius = (r) => {this.radius=r;}

var r = new Circle(3);
r.setRadius(10);
var p = r.perimeter();
```

```
String.prototype.endsWith = function(c) {
    return (c == this.charAt(this.length-1))
}
'Привет, мир'.endsWith('p');          //true
```

Наследование свойств



Эмуляция классов

Классов нет, эмулируются конструкторами и прототипами.
Класс и экземпляр (instance).

c = new Circle();

свойство экземпляра	c.r	прототип
метод экземпляра	c.area()	прототип
свойство класса	Circle.PI	класс
метод класса	Circle.max (c1, c2)	класс

Наследование классов

```
function Disk(r){
   this.radius = r;
}

Disk.prototype = new Circle(0);

Disk.prototype.area = function() {
   return Circle.PI * this.r * this.r;
}

Disk.prototype.constructor = Disk;
                                  // восстановили
```

Классы (ES6) - продолжение

```
var aggregation = (baseClass, ...mixins) => {
    let base = class _Combined extends baseClass {
        ...
    }
    ...
    return base
}

class Colored {...}
class ZCoord {...}
class Shape {...}

class Rectangle extends
   aggregation(Shape, Colored, ZCoord) {}

var rect = new Rectangle(7, 42)
rect.z     = 1000
rect.color = "red"
console.log(rect.x, rect.y, rect.z, rect.color)
```

Классы (ES6) - продолжение

Доступ к базовому классу

```
class Shape {
    ...
    toString () {return `Shape(${this.id})`}
}
class Rectangle extends Shape {
    constructor (id, x, y, width, height) {
        super(id, x, y)
        ...
    }
    toString () {return "Rectangle > " + super.toString()}
}
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        …
    }
    toString () {return "Circle > " + super.toString()}
}
```

Классы (ES6) - продолжение

Статические члены класса

```
class Rectangle extends Shape {
    …
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
class Circle extends Shape {
    …
    static defaultCircle () {
        return new Circle("default", 0, 0, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
var defCircle    = Circle.defaultCircle()
```

Классы (ES6) - продолжение

Геттеры и сеттеры

```
class Rectangle {
  constructor (width, height) {
    this._width  = width
    this._height = height
  }
  set width  (width)  { this._width = width }
  get width  ()       { return this._width  }
  set height (height) { this._height = height }
  get height ()       { return this._height }
  get area   ()       { return this._width*this._height }
}

var r = new Rectangle(50, 20)
r.area === 1000
```

Литералы объектов (ES6)

```
obj = { x, y }     // { x: x, y: y }
```

```
let obj = {
    foo: "bar",
    [ "baz" + quux() ]: 42
}
```

```
obj = {
    foo (a, b) {...},
    bar (x, y) {...},
    *quux (x, y) {...}
}
```