생성자

- new 키워드도 객체 생성

생성자

```
<script>
        function Student(){}

        var student = new Student();
</script>
```

프로토타입

- 객체 안에 함수가 선언되는 것이 비효율적이어서 이를 해결하기 위해 만듦
- 프도도타입은 생성자 함수도 생성된 객체가 공동으도 가지는 공간
- 메서드를 하나만 생성해도 모든 객체가 해당 메서드를 사용할 수 있음.
- 모든 함수는 prototype을 가짐.
- 다음과 같이 prototype함수 선언할 수 있음

Prototype

```
<script>

    function Student(name, korean, math, english, science){
        this. = name;
        this. = korean;
        this. = math;
        this. = english;
        this. = science;
    }

    Student.prototype.getSum = function(){};
    Student.prototype.getAverage = function(){};
    Student.prototype.toString = function(){};

</script>
```

상속

→ 클래스 형식으도 작성하면 extends 명뎡어도 상속 가능(그냥 자바처덤 사용가능)

상속

```
<script>
     //Rectangle (function)

    //base
    function Square(length){
        this.base = Rectangle;
        this.base(length, length);
    }

    //  square  rectangle
    Square.prototype = Rectangle.prototype;

    //      Square
    Square.prototype.constructor = Square;

</script>
```

상속 확인

```
<script>

   var square = new Square(5);
   //  true square rectangle
   alert(square instanceof Rectangle);

</script>
```

클래스 식으로 나타내기

클래스 기반의 객체지향 언어적 득성도 지원한다.

Class식 선언

```
<script>

    class Rectangle{
        constructor(width, height){
            this.width = width;
            this.height = height;
        }

        getArea(){
            return this.width * this.height;
        }
    }

    const rectangle = new Rectangle(100, 200);
    alert(rectangle.getArea());

</script>
```

함수식 선언

```
<script>

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

    Rectangle.prototype.getArea = function(){
        return this.width * this.height;
    }

    var rectangle = new Rectangle(100, 200);
    alert(rectangle.getArea());

</script>
```

==> 둘이 같은 코드