

super

Jump to: [Syntax](#) [Description](#) [Example](#) [Specifications](#) [Browser compatibility](#) [See also](#)

The **super** keyword is used to access and call functions on an object's parent.

The `super.prop` and `super[expr]` expressions are valid in any method definition in both classes and object literals.

Syntax

```
super([arguments]); // calls the parent constructor.
super.functionOnParent([arguments]);
```

Description

When used in a constructor, the `super` keyword appears alone and must be used before the `this` keyword is used. The `super` keyword can also be used to call functions on a parent object.

Example

Using `super` in classes

This code snippet is taken from the [ES6 classes sample](#) ([live demo](#)). Here `super()` is called to avoid duplicating the constructor parts that are common between `Rectangle` and `Square`.

```
1  class Rectangle {
2    constructor(height, width) {
3      this.name = 'Rectangle';
4      this.height = height;
5      this.width = width;
6    }
7    sayName() {
8      console.log('Hi, I am a ', this.name + '.');
9    }
10   get area() {
11     return this.height * this.width;
12   }
13   set area(value) {
14     this.height = this.width = Math.sqrt(value);
15   }
16 }
17
18 class Square extends Rectangle {
19   constructor(length) {
20     this.height; // ReferenceError, super needs to be called first!
21
22     // Here, it calls the parent class' constructor with lengths
```

```

23     // provided for the Rectangle's width and height
24     super(length, length);
25
26     // Note: In derived classes, super() must be called before you
27     // can use 'this'. Leaving this out will cause a reference error.
28     this.name = 'Square';
29 }
30 }

```

Super-calling static methods

You are also able to call super on static methods.

```

1  class Rectangle {
2      constructor() {}
3      static logNbSides() {
4          return 'I have 4 sides';
5      }
6  }
7
8  class Square extends Rectangle {
9      constructor() {}
10     static logDescription() {
11         return super.logNbSides() + ' which are all equal';
12     }
13 }
14 Square.logDescription(); // 'I have 4 sides which are all equal'

```

Deleting super properties will throw an error

You cannot use the `delete` operator and `super.prop` or `super[expr]` to delete a parent class' property, it will throw a `ReferenceError`.

```

1  class Base {
2      constructor() {}
3      foo() {}
4  }
5  class Derived extends Base {
6      constructor() {}
7      delete() {
8          delete super.foo; // this is bad
9      }
10 }
11
12 new Derived().delete(); // ReferenceError: invalid delete involving 'super'.

```

super.prop cannot overwrite non-writable properties

When defining non-writable properties with e.g. `Object.defineProperty`, `super` cannot overwrite the value of the property.

```

1  class X {
2      constructor() {
3          Object.defineProperty(this, 'prop', {
4              configurable: true,
5              writable: false,
6              value: 1
7          });
8      }
9  }
10
11 class Y extends X {
12     constructor() {
13         super();
14     }
15     foo() {
16         super.prop = 2; // Cannot overwrite the value.
17     }
18 }
19
20 var y = new Y();
21 y.foo(); // TypeError: "prop" is read-only
22 console.log(y.prop); // 1

```

Using `super.prop` in object literals

Super can also be used in the object initializer / literal notation. In this example, two objects define a method. In the second object, `super` calls the first object's method. This works with the help of `Object.setPrototypeOf()` with which we are able to set the prototype of `obj2` to `obj1`, so that `super` is able to find `method1` on `obj1`.

```
1  var obj1 = {
2    method1() {
3      console.log('method 1');
4    }
5  }
6
7  var obj2 = {
8    method2() {
9      super.method1();
10   }
11 }
12
13 Object.setPrototypeOf(obj2, obj1);
14 obj2.method2(); // logs "method 1"
```

Specifications

Specification	Status	Comment
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'super' in that specification.	<div><div>ST</div>Standard</div>	Initial definition.
ECMAScript Latest Draft (ECMA-262) The definition of 'super' in that specification.	<div><div>D</div>Draft</div>	

Browser compatibility

New compatibility tables are in beta ▾

Basic support															
42	Yes	45	No	Yes	Yes	42	42	Yes	45	Yes	Yes	Yes	4.0	Yes	

..

Full support

..

No support

See also

- Classes