# Method definitions

Starting with ECMAScript 2015, a shorter syntax for method definitions on objects initializers is introduced. It is a shorthand for a function assigned to the method's name.

## Syntax

```
var obj = {
  property( parameters… ) {},
  *generator( parameters… ) {},
  async property( parameters… ) {},
  async* generator( parameters… ) {},

  // with computed keys:
  [property]( parameters… ) {},
  *[generator]( parameters… ) {},
  async [property]( parameters… ) {},

  // compare getter/setter syntax:
  get property() {},
  set property(value) {}
};
```

## Description

The shorthand syntax is similar to the getter and setter syntax introduced in ECMAScript 2015.

Given the following code:

```
var obj = {
  foo: function() {

  },
  bar: function() {
```

```
  }
};
```

You are now able to shorten this to:

```
var obj = {
  foo() {

  },
  bar() {

  }
};
```

## Generator methods

Generator methods can be defined using the shorthand syntax as well. When using them,

- the asterisk (*) in the shorthand syntax must be before the generator property name. That is, `* g(){}` will work but `g *(){}` will not;
- non-generator method definitions may not contain the `yield` keyword. This means that legacy generator functions won't work either and will throw a `SyntaxError`. Always use `yield` in conjunction with the asterisk (*).

```
var obj2 = {
  g: function* () {
    var index = 0;
    while (true)
      yield index++;
  }
};
```

```
var obj2 = {
  * g() {
    var index = 0;
    while (true)
      yield index++;
  }
};

var it = obj2.g();
console.log(it.next().value);
console.log(it.next().value);
```

## Async methods

Async methods can also be defined using the shorthand syntax.

```
var obj3 = {
  f: async function () {
    await some_promise;
  }
};
```

```
var obj3 = {
  async f() {
    await some_promise;
  }
};
```

## Async generator methods

Generator methods can also be async.

```
var obj4 = {
  f: async function* () {
```

```
    yield 1;
    yield 2;
    yield 3;
  }
};


var obj4 = {
  async* f() {
   yield 1;
   yield 2;
   yield 3;
  }
};
```

## Method definitions are not constructable

All method definitions are not constructors and will throw a `TypeError` if you try to instantiate them.

```
var obj = {
  method() {}
};
new obj.method;

var obj = {
  * g() {}
};
new obj.g;
```

# Examples

## Simple test case

```
var obj = {
  a: 'foo',
  b() { return this.a; }
};
```

```
console.log(obj.b());
```

### Computed property names

The shorthand syntax also supports computed property names.

```
var bar = {
  foo0: function() { return 0; },
  foo1() { return 1; },
  ['foo' + 2]() { return 2; }
};

console.log(bar.foo0());
console.log(bar.foo1());
console.log(bar.foo2());
```

## Specifications

| Specification | Status | Comment |
| --- | --- | --- |
| ECMAScript 2015 (6th Edition, ECMA-262)<br>The definition of 'Method definitions' in that specification. | Standard | Initial definition. |
| ECMAScript 2016 (ECMA-262)<br>The definition of 'Method definitions' in that specification. | Standard | Changed that generator methods should also not have a [[Construct]] trap and will throw when used with `new`. |
| ECMAScript Latest Draft (ECMA-262)<br>The definition of 'Method definitions' in that specification. | Living Standard | |

## Browser compatibility

## See also

- `get`
- `set`
- Lexical grammar

*Was this article helpful?*

*Thank you!*