

# Object.preventExtensions()

The **Object.preventExtensions()** method prevents new properties from ever being added to an object (i.e. prevents future extensions to the object).

```
JavaScript Demo: Object.preventExtensions()
1 const object1 = {};
 2
3 Object.preventExtensions(object1);
4
5 try {
     Object.defineProperty(object1, 'property1', {
7
       value: 42
     });
9 } catch (e) {
   console.log(e);
10
    // expected output: TypeError: Cannot define property
11
12 }
13
  Run>
  Reset
```

### Syntax

Object.preventExtensions(obj)

#### Parameters

#### obj

The object which should be made non-extensible.

### Return value

The object being made non-extensible.

## Description

An object is extensible if new properties can be added to it.

Object.preventExtensions() marks an object as no longer extensible, so that it will never have properties beyond the ones it had at the time it was marked as non-extensible. Note that the properties of a non-extensible object, in general, may still be *deleted*. Attempting to add new properties to a non-extensible object will fail, either silently or by throwing a <a href="TypeError">TypeError</a> (most commonly, but not exclusively, when in <a href="Strict mode">strict mode</a>).

Object.preventExtensions() only prevents addition of own properties. Properties can still be added to the object prototype.

This method makes the [[prototype]] of the target immutable; any [[prototype]] re-assignment will throw a TypeError. This behavior is specific to the internal [[prototype]] property, other properties of the target object will remain mutable.

There is no way to make an object extensible again once it has been made

non-extensible.

# **Examples**

# Using Object.preventExtensions // Object.preventExtensions returns the object // being made non-extensible. var obj = {}; var obj2 = Object.preventExtensions(obj); obj === obj2; // true // Objects are extensible by default. var empty = {}; Object.isExtensible(empty); // === true // ...but that can be changed. Object.preventExtensions(empty); Object.isExtensible(empty); // === false // Object.defineProperty throws when adding // a new property to a non-extensible object. var nonExtensible = { removable: true }; Object.preventExtensions(nonExtensible); Object.defineProperty(nonExtensible, 'new', { value: 8675309 }); // throws a TypeError // In strict mode, attempting to add new properties // to a non-extensible object throws a TypeError. function fail() { 'use strict'; // throws a TypeError nonExtensible.newProperty = 'FAIL'; fail();

A non-extensible object's prototype is immutable:

```
var fixed = Object.preventExtensions({});
// throws a 'TypeError'.
fixed.__proto__ = { oh: 'hai' };
```

### Non-object coercion

In ES5, if the argument to this method is not an object (a primitive), then it will cause a <a href="TypeError">TypeError</a>. In ES2015, a non-object argument will be treated as if it was a non-extensible ordinary object, return it.

```
Object.preventExtensions(1);
// TypeError: 1 is not an object (ES5 code)
Object.preventExtensions(1);
// 1 (ES2015 code)
```

### **Specifications**

Specification

ECMAScript (ECMA-262)

The definition of 'Object.preventExtensions' in that specification. ☐

# Browser compatibility

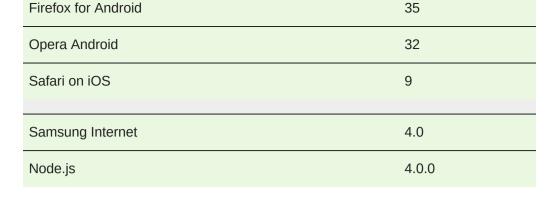
Report problems with this compatibility data on GitHub

preventExtensions

Chrome 6

. .

Eage	12
Firefox	4
Internet Explorer	9
Opera	12
Safari	5.1
WebView Android	1
Chrome Android	18
Firefox for Android	4
Opera Android	12
Safari on iOS	6
Samsung Internet	1.0
Node.js	0.10.0
ES2015 behavior for non-object argument	
Chrome	44
Edge	12
Firefox	35
Internet Explorer	11
Opera	31
Safari	9
WebView Android	44
Webview Android	





### See also

- Object.isExtensible()
- Object.seal()
- Object.isSealed()
- Object.freeze()
- Object.isFrozen()
- Reflect.preventExtensions()

Last modified: Apr 6, 2021, by MDN contributors

## Change your language

English (US) Change language