When your constructor has something like  this.member and you invoke the constructor without  new,  you’re  actually  creating  a  new  property  of  the  global  object  called member and accessible through window.member or simply member.

[复制代码](javascript:void(0);)

// constructor

function Waffle() {

this.tastes = "yummy";

}

// a new object

var good\_morning = new Waffle();

console.log(typeof good\_morning); // "object"

console.log(good\_morning.tastes); // "yummy"

// antipattern:

// forgotten `new`

var good\_morning = Waffle();

console.log(typeof good\_morning); // "undefined"

console.log(window.tastes); // "yummy"

[复制代码](javascript:void(0);)

This undesired behavior is addressed in ECMAScript 5, and in strict mode this  will no longer point to the global object. If ES5 is not available, there’s still something you can do to make sure that a constructor function always behaves like one even if called without new.

**Naming Convention**

uppercase the first letter in constructor names (MyConstructor) and lowercase it in “normal” functions and methods (myFunction).

**Using that**

[复制代码](javascript:void(0);)

function Waffle() {

var that = {};

that.tastes = "yummy";

return that;

}

function Waffle() {

return {

tastes: "yummy"

};

}

var first = new Waffle(),

second = Waffle();

console.log(first.tastes); // "yummy"

console.log(second.tastes); // "yummy"

[复制代码](javascript:void(0);)

**Disadvantage**

The link to the prototype is lost, so any members you add to the Waffle()prototype will not be available to the objects.

**Self-Invoking Constructor**

In the constructor you check whether this is an instance of your constructor, and if not, the constructor invokes itself again, this time properly with new:

[复制代码](javascript:void(0);)

function Waffle() {

if (!(this instanceof Waffle)) {

return new Waffle();

}

this.tastes = "yummy";

}

Waffle.prototype.wantAnother = true;

// testing invocations

var first = new Waffle(),

second = Waffle();

console.log(first.tastes); // "yummy"

console.log(second.tastes); // "yummy"

console.log(first.wantAnother); // true

console.log(second.wantAnother); // true

[复制代码](javascript:void(0);)