

Function

Property Summary

Function [#constructor](#)
Specifies the function that creates an object's prototype.

Number [#length](#)
Specifies the number of arguments expected by the function.

Method Summary

void [#apply\(\)](#)
Applies the method of another object in the context of a different object (the calling object); arguments can be passed as an Array object.

void [#bind\(\)](#)
Creates a new function which, when called, itself calls this function in the context of the provided value, with a given sequence of arguments preceding any provided when the new function was called.

void [#call\(\)](#)
Calls (executes) a method of another object in the context of a different object (the calling object); arguments can be passed as they are.

String [#toString\(\)](#)
Returns a string representing the source code of the function.

Property Details

constructor
Specifies the function that creates an object's prototype.

Returns

[Function](#)

Sample

```
function Tree(name) {
    this.name = name;
}
theTree = new Tree("Redwood");
application.output("theTree.constructor is " + theTree.constructor);
```

length
Specifies the number of arguments expected by the function.

Returns

[Number](#)

Sample

```
function addNumbers(x, y){
    if (addNumbers.length == 2) {
        return (x + y);
    } else
        return 0;
}
```

Method Details

apply
void **apply()**
Applies the method of another object in the context of a different object (the calling object); arguments can be passed as an Array object.

Returns

void

Sample

```
function book(name, author) {
    this.name = name;
    this.author = author;
}

function book_with_topic(name, author, topic) {
    this.topic = topic;
    book.apply(this, arguments);
}
book_with_topic.prototype = new book();

var aBook = new book_with_topic("name", "author", "topic");
```

[bind](#)

void **bind()**

Creates a new function which, when called, itself calls this function in the context of the provided value, with a given sequence of arguments preceding any provided when the new function was called.

Returns

void

Sample

```
var x = 9,
    module = {
        getX: function() {
            return this.x;
        },
        x: 81
    };
// "module.getX()" called, "module" is "this", "module.x" is returned
module.getX(); // > 81
// "getX()" called, "this" is global, "x" is returned
getX(); // > 9
// store a reference with "module" bound as "this"
var boundGetX = getX.bind(module);
// "boundGetX()" called, "module" is "this" again, "module.x" is returned
boundGetX(); // > 81
```

[call](#)

void **call()**

Calls (executes) a method of another object in the context of a different object (the calling object); arguments can be passed as they are.

Returns

void

Sample

```
function book(name) {
    this.name = name;
}

function book_with_author(name, author) {
    this.author = author;
    book.call(this, name);
}
book_with_author.prototype = new book();

var aBook = new book_with_author("name", "author");
```

[toString](#)

[String](#) **toString()**

Returns a string representing the source code of the function. Overrides the Object.toString method.

Returns

[String](#)

Sample

```
function printHello() {  
    return "Hello";  
}  
application.output(printHello.toString());
```