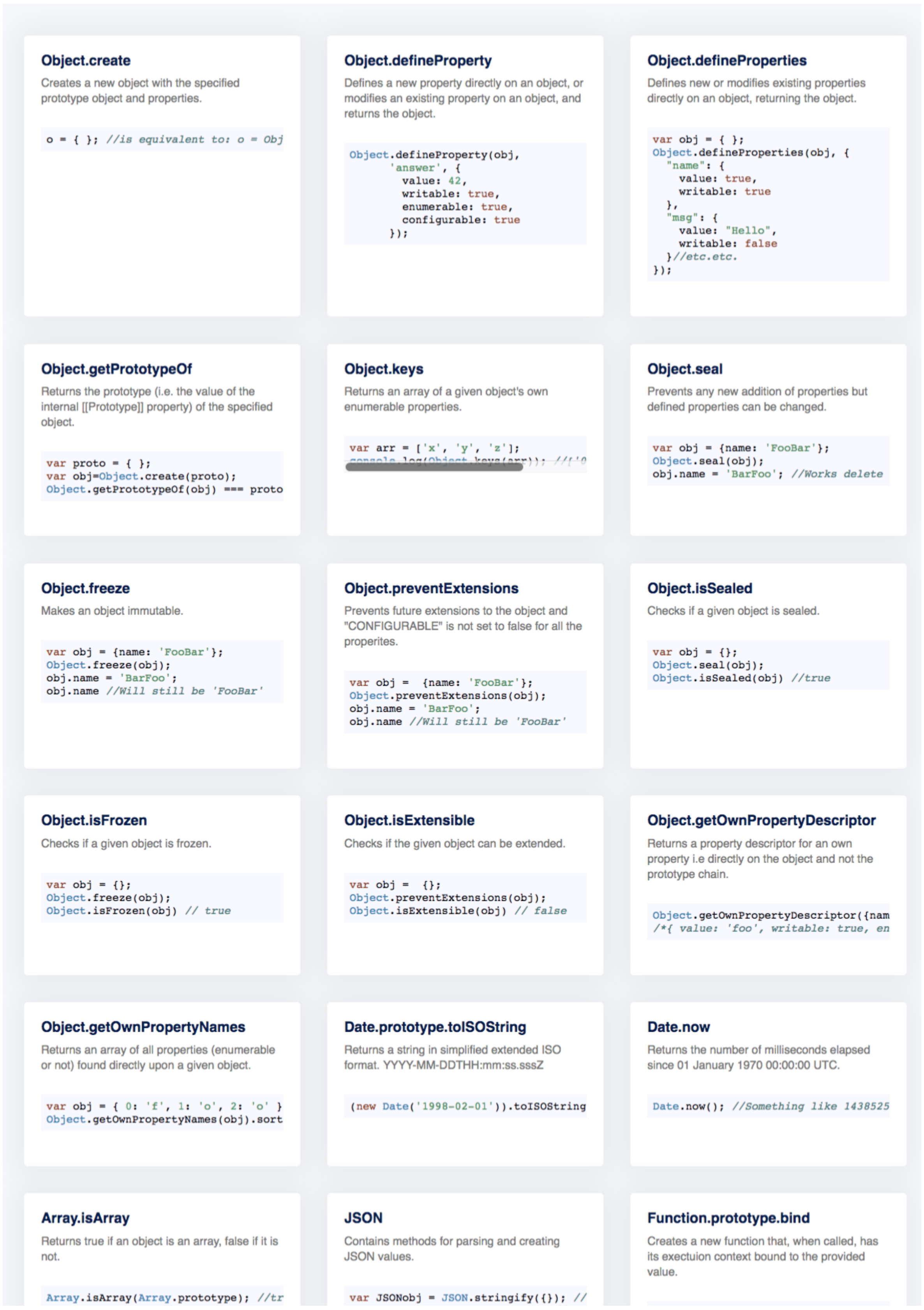
Ecma 5 Cheat Sheet



Object.create

Creates a new object with the specified prototype object and properties.

```
o = { }; //is equivalent to: o = Obj
```

Object.defineProperty

Defines a new property directly on an object, or modifies an existing property on an object, and returns the object.

```
Object.defineProperty(obj,
      'answer', {
        value: 42,
        writable: true,
        enumerable: true,
        configurable: true
      });
```

Object.defineProperties

Defines new or modifies existing properties directly on an object, returning the object.

```
var obj = { };
Object.defineProperties(obj, {
  "name": {
    value: true,
    writable: true
  },
  "msg": {
    value: "Hello",
    writable: false
  }//etc.etc.
});
```

Object.getPrototypeOf

Returns the prototype (i.e. the value of the internal [[Prototype]] property) of the specified object.

```
var proto = { };
var obj=Object.create(proto);
Object.getPrototypeOf(obj) === proto
```

Object.keys

Returns an array of a given object's own enumerable properties.

```
var arr = ['x', 'y', 'z'];
console.log(Object.keys(arr)); //['0
```

Object.seal

Prevents any new addition of properties but defined properties can be changed.

```
var obj = {name: 'FooBar'};
Object.seal(obj);
obj.name = 'BarFoo'; //Works delete
```

Object.freeze

Makes an object immutable.

```
var obj = {name: 'FooBar'};
Object.freeze(obj);
obj.name = 'BarFoo';
obj.name //Will still be 'FooBar'
```

Object.preventExtensions

Prevents future extensions to the object and "CONFIGURABLE" is not set to false for all the properites.

```
var obj =  {name: 'FooBar'};
Object.preventExtensions(obj);
obj.name = 'BarFoo';
obj.name //Will still be 'FooBar'
```

Object.isSealed

Checks if a given object is sealed.

```
var obj = {};
Object.seal(obj);
Object.isSealed(obj) //true
```

Object.isFrozen

Checks if a given object is frozen.

```
var obj = {};
Object.freeze(obj);
Object.isFrozen(obj) // true
```

Object.isExtensible

Checks if the given object can be extended.

```
var obj =  {};
Object.preventExtensions(obj);
Object.isExtensible(obj) // false
```

Object.getOwnPropertyDescriptor

Returns a property descriptor for an own property i.e directly on the object and not the prototype chain.

```
Object.getOwnPropertyDescriptor({nam
/*{ value: 'foo', writable: true, en
```

Object.getOwnPropertyNames

Returns an array of all properties (enumerable or not) found directly upon a given object.

```
var obj = { 0: 'f', 1: 'o', 2: 'o' }
Object.getOwnPropertyNames(obj).sort
```

Date.prototype.toISOString

Returns a string in simplified extended ISO format. YYYY-MM-DDTHH:mm:ss.sssZ

```
(new Date('1998-02-01')).toISOString
```

Date.now

Returns the number of milliseconds elapsed since 01 January 1970 00:00:00 UTC.

```
Date.now(); //Something like 1438525
```

Array.isArray

Returns true if an object is an array, false if it is not.

```
Array.isArray(Array.prototype); //tr
```

JSON

Contains methods for parsing and creating JSON values.

```
var JSONobj = JSON.stringify({}); //
```

Function.prototype.bind

Creates a new function that, when called, has its exectuion context bound to the provided value.

```
Array.isArray(); //true
Array.isArray({}); //false
Array.isArray(null); //false
```

```
var obj = JSON.parse(JSONobj); //{}
```

```
var log = console.log.bind(console);
log('meow') //meow
```

String.prototype.trim

Removes whitespace from both ends of a string.

```
var name = 'foo';
name.trim(); //'foo'
```

Array.prototype.indexOf

Returns the index if found, else returns -1, takes an optional starting index.

```
var name = 'Brendan Eich';
name.indexOf('Eich'); //8
name.indexOf('Brendan', 5); //-1
```

Array.prototype.lastIndexOf

Returns the last index if found or -1 for the specified value, also takes an optional starting index.

```
var city = 'mississippi';
city.lastIndexOf('i'); //10
```

Array.prototype.every

Checks if all the elements of an array passes the specified test.

```
[1, 2, 3].every(function(v, i, a) {
  return v > 3;
});  //false

[1, 2, 3].every(function(v, i, a) {
  return v > 0;
}); //true
```

Array.prototype.some

Checks if any of the elements of an array passes the specified test.

```
[1, 3, 5, 7, 6].some(function(v, i,
  return v%2 === 0;
}); //true
```

Array.prototype.forEach

For each element in an array performs the specified action.

```
['foo', 'bar', 'baz'].forEach(functi
  console.log(v, i);
}); /* foo 0 bar 1 baz 2 */
```

Array.prototype.map

Returns an array that contains the results of a invokation of the function passed to it.

```
[64, 49].map(Math.sqrt); //[8, 7]
```

Array.prototype.filter

Returns an array that meet the condition specified in a callback function.

```
[1, '', true, false].filter(Boolean)
//[1, true]
```

Array.prototype.reduce

Invokes the callback with an accumulator and each value of the array and reduce it to a single value.

```
[0, 1, 2, 3, 4].reduce(function(prev
  return previousValue+currentValue;
}, 10); //20
```

Array.prototype.reduceRight

Similar to reduce, but acts upon the array from right-to-left instead.

```
[0, -1, -2, 5, -6].reduceRight(funct
  return previousValue+currentValue;
}); //-4
```

Getter property in objects

Binds an object property to a function that will be invoked when that property is accessed.

```
foo= { getx() { return42 }};
foo.x; //42
```

Setter property in objects

Binds an object property to a function that will be invoked when that property is been set.

```
var val = 0;
var foo = { set x(v) {
  val = v
}};
foo.x = 42
val; //42
```

Property accessor of strings

Helps to access char in a string by index.

```
"foobar"[2]; //'o'
```

Reserved word property name

Using reserved as property names.

```
{ class: 42 }.class //42
```

ZWSP identifiers

zero-width space identifiers.

```
var _c = 42;
_c; //42
```

ignore leading zeros in parseInt()

Ingoring the leading zero.

```
parseInt('000420'); //420
```

Immutable undefined

Can not mutate undefined.

```
undefined = 42;
typeof undefined; //'undefined'
```