**JAVASCRIPT – Objects**

1. **Object Properties**

Properties are the values associated with a JavaScript object.

A JavaScript object is a collection of unordered properties.

Properties can usually be changed, added, and deleted, but some are read only.

1. **Object Methods**

JavaScript methods are actions that can be performed on objects.

A JavaScript method is a property containing a function definition.

1. **User-Defined Objects**

User-defined objects: these are objects which you have created in your program or application.

/\* defining a new constructor function \*/

function Person(name, age, city) {

this.name = name;

this.age = age;

this.city = city;

}

1. **Defining Methods for an Object**

**const person = {**

**firstName: "John",**

**lastName: "Doe",**

**id: 5566,**

**fullName: function() {**

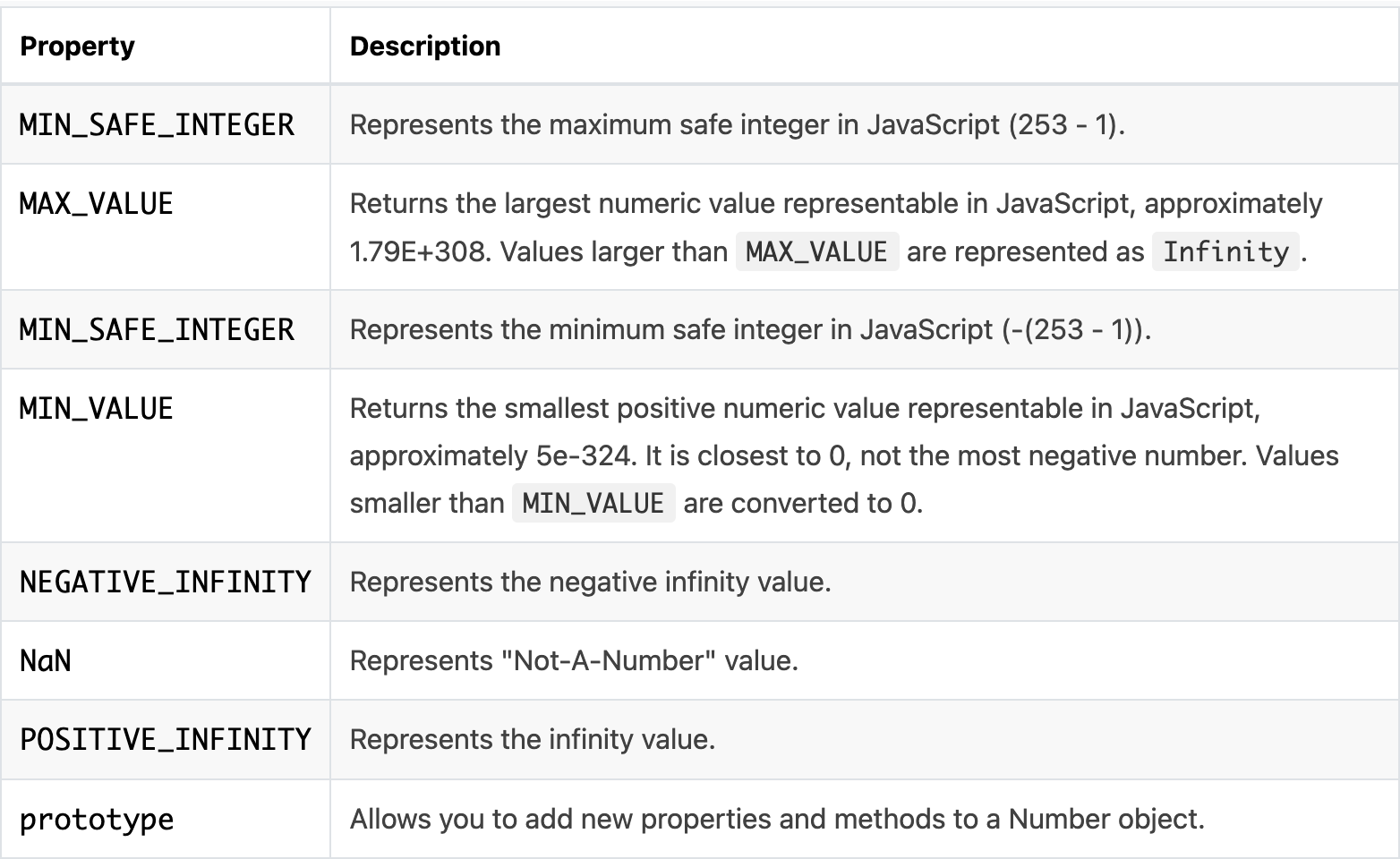
**return this.firstName + " " + this.lastName;**

**}**

**};**

**JAVASCRIPT – Number**

* **Number Properties**

****

* **MAX\_VALUE**

**Number.MAX\_VALUE** returns the largest number possible in JavaScript.

**Number.MAX\_VALUE** has the value of 1.7976931348623157e+308.

* **MIN\_VALUE**

**Number.MIN\_VALUE** returns the smallest number possible in JavaScript.

**Number.MIN\_VALUE** has a value of 5e-324.

* **NaN**

In JavaScript, NaN is short for "Not-a-Number".

In JavaScript, NaN is a number that is not a legal number.

The Global NaN property is the same as the Number.Nan property.

* **NEGATIVE\_INFINITY**

**Number.NEGATIVE\_INFINITY** returns negative infinity.

**Number.NEGATIVE\_INFINITY** is "a number lower than any other number".

* **POSITIVE\_INFINITY**

**POSITIVE\_INFINITY** returns positive infinity.

**POSITIVE\_INFINITY** is "something higher than any other number".

* **Prototype**

The JavaScript prototype property also allows you to add new methods to objects constructors:

**function Person(first, last, age, eyecolor) {**

**this.firstName = first;**

**this.lastName = last;**

**this.age = age;**

**this.eyeColor = eyecolor;**

**}**

**Person.prototype.name = function() {**

**return this.firstName + " " + this.lastName;**

**};**

* **Constructor**

In JavaScript, a constructor function is used to create objects. For example,

*// constructor function*

*function Person () {*

*this.name = 'John',*

*this.age = 23*

*}*

*// create an object*

*const person = new Person();*

* **Number Methods**

Primitive values (like 3.14 or 2014), cannot have properties and methods (because they are not objects).

But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

* **toExponential ()**

**toExponential()** returns a string, with a number rounded and written using exponential notation.

A parameter defines the number of characters behind the decimal point:

**let x = 9.656;**

**x.toExponential(2);**

**x.toExponential(4);**

**x.toExponential(6);**

* **toFixed ()**

**toFixed()** returns a string, with the number written with a specified number of decimals:

**let x = 9.656;**

**x.toFixed(0);**

**x.toFixed(2);**

**x.toFixed(4);**

**x.toFixed(6);**

* **toLocaleString ()**

The toLocaleString() returns a number as a string, using local language format.

The language format depends on the locale setup on your computer.

**let num = 1000000;**

**let text = num.toLocaleString();**

* **toPrecision ()**

The toPrecision() method formats a number to a specified length.

A decimal point and nulls are added (if needed), to create the specified length.

**let num = 13.3714;**

**let n = num.toPrecision(2);**

* **toString ()**

The toString() returns a number as a string.

**let num = 15;**

**let text = num.toString();**

* **valueOf ()**

The valueOf() method returns the primitive value of a number.

**let num = 15;**

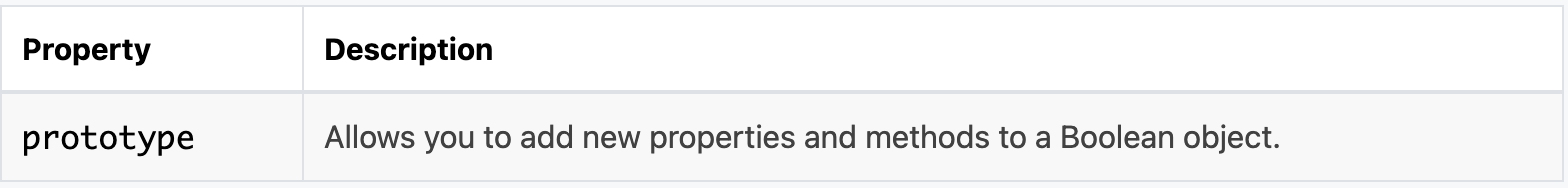
**let n = num.valueOf();**

**JAVASCRIPT – Boolean**

* **Boolean Properties**

A JavaScript Boolean represents one of two values: true or false.

The following table lists the standard properties of the Boolean object.



* **Constructor ()**

The constructor property returns the function that created the Boolean prototype.

For JavaScript booleans the constructor property returns:

function Boolean() { [native code] }

* **Prototype**

prototype allows you to add new properties and methods to booleans.

prototype is a property available with all JavaScript objects.

Make a new method for JavaScript booleans:

**Boolean.prototype.myColor = function() {**

**if (this.valueOf() == true) {**

**return "green";**

**} else {**

**return = "red";**

**}**

**};**

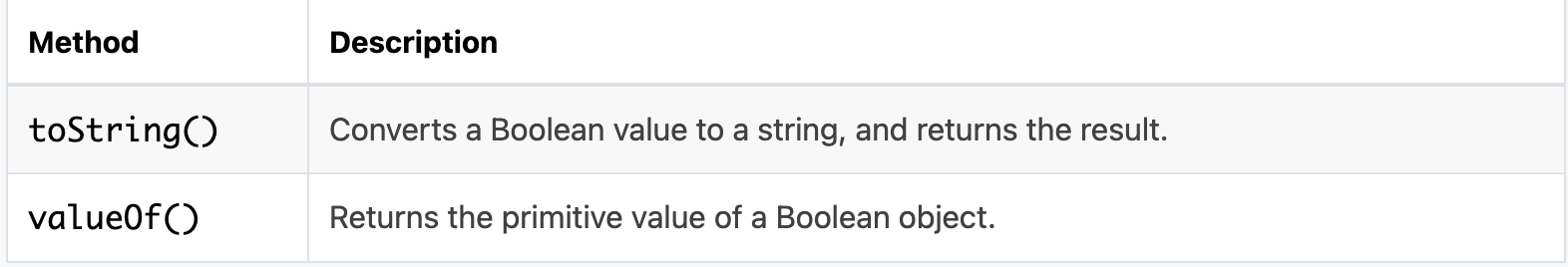
**Create a boolean, then call myColor():**

**let a = true;**

**let color = a.myColor();**

* **Boolean Methods**

The following table lists the standard methods of the Boolean object.

****

* **toSource ()**

JavaScript boolean toSource() method returns a string representing the source code of the object.

**<script type = "text/javascript">**

**function book(title, publisher, price) {**

**this.title = title;**

**this.publisher = publisher;**

**this.price = price;**

**}**

**var newBook = new book("Perl","Leo Inc",200);**

**document.write(newBook.toSource());**

**</script>**

* **toString ()**

The toString() method returns a boolean as a string.

**let bool = true;**

**let text = bool.toString();**

* **valueOf ()**

valueOf() returns the primitive value of a boolean.

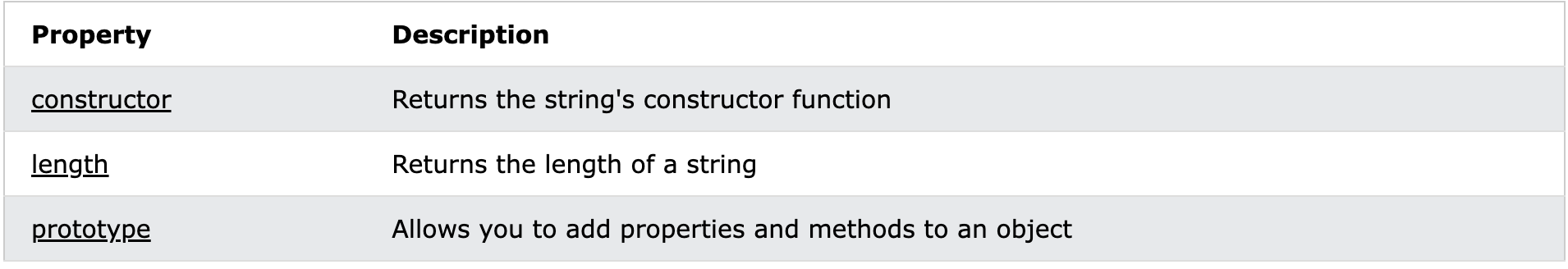
valueOf() is usually called by JavaScript behind the scenes, and not explicitly in code.

**let bool = false;**

**let value = bool.valueOf();**

**JAVASCRIPT – String**

* **String Properties**

****

* **Constructor**

The constructor property returns the function that created the String prototype.

For JavaScript strings the constructor returns:

function String() { [native code] }

* **Length**

The length property returns the length of a string.

The length property of an empty string is 0.

* **Prototype**

The prototype is a property available with all JavaScript objects.

The prototype property allows you to add new properties and methods to strings.

**function employee(name, jobtitle, born) {**

**this.name = name;**

**this.jobtitle = jobtitle;**

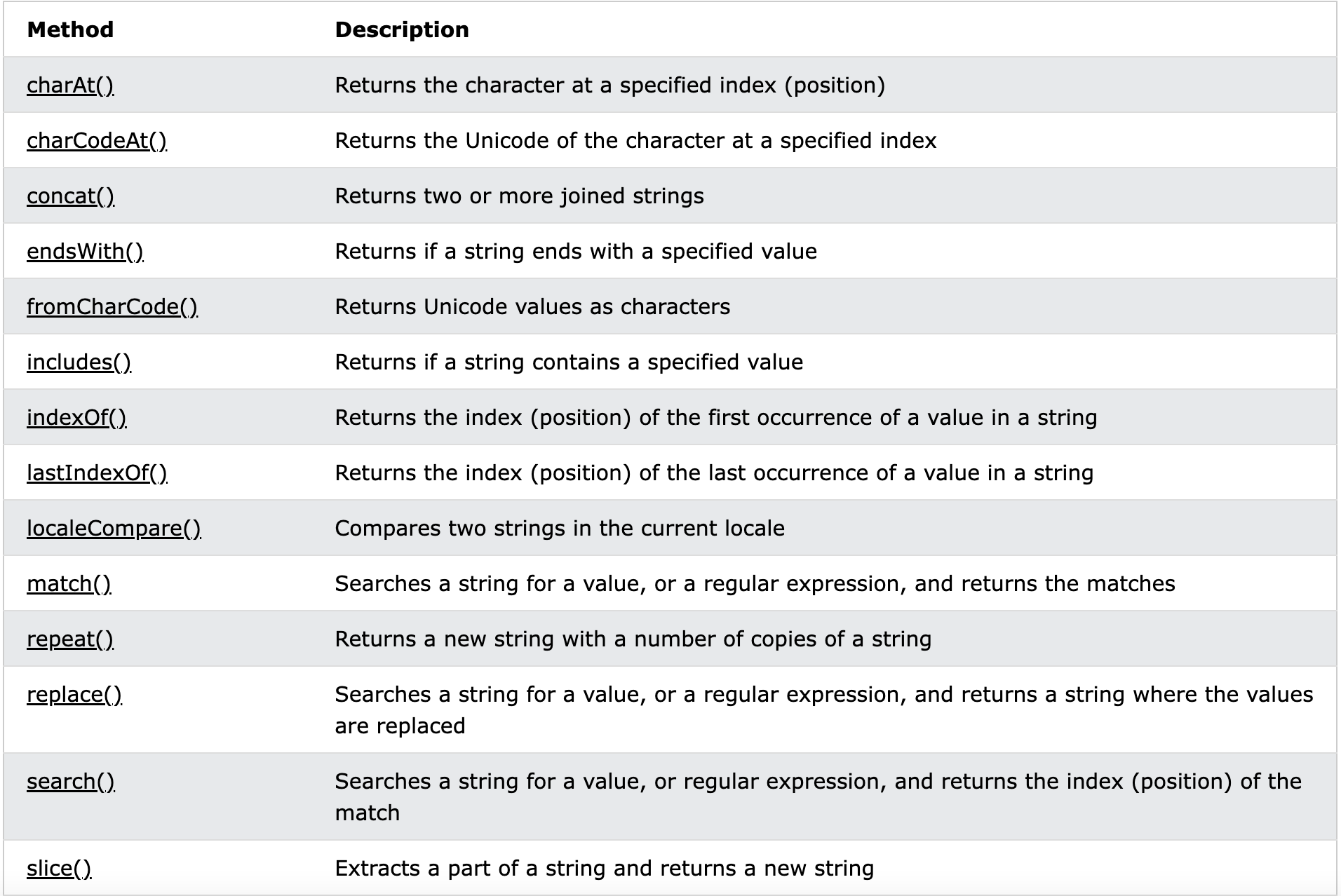
**this.born = born;**

**}**

**employee.prototype.salary = 2000;**

**const fred = new employee("Fred Flintstone", "Caveman", 1970);**

* **String Methods**



* **charAt**

The charAt() method returns the character at a specified index (position) in a string.

The index of the first character is 0, the second 1, ...

The index of the last character is string length - 1

**let text = "HELLO WORLD";**

**let letter = text.charAt(0);**

* **charCodeAt ()**

The charCodeAt() method returns the Unicode of the character at a specified index (position) in a string.

The index of the first character is 0, the second is 1, ....

The index of the last character is string length - 1

* **contact ()**

The concat() method joins two or more strings.

The concat() method does not change the existing strings.

The concat() method returns a new string.

**let text1 = "sea";**

**let text2 = "food";**

**let result = text1.concat(text2);**

* **indexOf ()**

The indexOf() method returns the position of the first occurrence of a value in a string.

The indexOf() method returns -1 if the value is not found.

The indexOf() method is case sensitive.

**let text = "Hello world, welcome to the universe.";**

**text.indexOf("e");**

* **lastIndexOf ()**

The lastIndexOf() method returns the index (position) of the last occurrence of a specified value in a string.

The lastIndexOf() method searches the string from the end to the beginning.

The lastIndexOf() method returns the index from the beginning (position 0).

The lastIndexOf() method returns -1 if the value is not found.

The lastIndexOf() method is case sensitive

**let text = "Hello planet earth, you are a great planet.";**

**let result = text.lastIndexOf("planet");**

* **localeCompare ()**

The localeCompare() method compares two strings in the current locale.

The localeCompare() method returns sort order -1, 1, or 0 (for before, after, or equal).

The current locale is based on the language settings of the browser.

**let text1 = "ab";**

**let text2 = "ab";**

**let result = text1.localeCompare(text2);**

* **match ()**

The match() method matches a string against a regular expression \*\*

The match() method returns an array with the matches.

The match() method returns *null* if no match is found.

**let text = "The rain in SPAIN stays mainly in the plain";**

**text.match("ain");**

* **replace ()**

The replace() method searches a string for a value or a regular expression.

The replace() method returns a new string with the value(s) replaced.

The replace() method does not change the original string.

**let text = "Visit Microsoft!";**

**let result = text.replace("Microsoft", "W3Schools");**

* **search ()**

The search() method matches a string against a regular expression \*\*

The search() method returns the index (position) of the first match.

The search() method returns -1 if no match is found.

The search() method is case sensitive.

**let text = "Mr. Blue has a blue house";**

**let position = text.search("Blue");**

* **slice()**

The slice() method extracts a part of a string.

The slice() method returns the extracted part in a new string.

The slice() method does not change the original string.

The start and end parameters specify the part of the string to extract.

The first position is 0, the second is 1, ...

A negative number is selected from the end of the string.

From position 3 to 8:

**let result = text.slice(3, 8);**

* **split()**

The split() method splits a string into an array of substrings.

The split() method returns the new array.

The split() method does not change the original string.

If (" ") is used as a separator, the string is split between words.

**let text = "How are you doing today?";**

**const myArray = text.split(" ");**

* **substr()**

The substr() method extracts a part of a string.

The substr() method begins at a specified position, and returns a specified number of characters.

The substr() method does not change the original string.

To extract characters from the end of the string, use a negative start position.

**let text = "Hello world!";**

**let result = text.substr(1, 4);"**

* **substring ()**

The substring() method extracts characters, between two indices (positions), from a string, and returns the substring.

The substring() method extracts characters from start to end (exclusive).

The substring() method does not change the original string.

If start is greater than end, arguments are swapped: (4, 1) = (1, 4).

Start or end values less than 0, are treated as 0.

**let text = "Hello world!";**

**let result = text.substring(1, 4);**

* **toLocaleLowerCase()**

The toLocaleLowerCase() method converts a string to lowercase letters, using the current locale.

The locale is based on the language settings of the browser.

The toLocaleLowerCase() method does not change the original string.

The toLocaleLowerCase() returns the same result as toLowerCase(), except for locales that conflict with the regular Unicode case mappings (such as Turkish).

**let text = "Hello World!";**

**let result = text.toLocaleLowerCase();**

# **toLocaleUpperCase()**

The toLocaleUpperCase() method converts a string to uppercase letters, using the current locale.

The locale is based on the language settings of the browser.

The toLocaleUpperCase() method does not change the original string.

The toLocaleUpperCase() returns the same result as toUpperCase(), except for locales that conflict with the regular Unicode case mappings (such as Turkish).

**let text = "Hello World!";**

**let result = text.toLocaleUpperCase();**

# **toLowerCase()**

The toLowerCase() method converts a string to lowercase letters.

The toLowerCase() method does not change the original string.

**let text = "Hello World!";**

**let result = text.toLowerCase();**

# **toString()**

The toString() method returns a string as a string.

The toString() method does not change the original string.

The toString() method can be used to convert a string object into a string.

**let text = "Hello World!";**

**let result = text.toString();**

# **toUpperCase()**

The toUpperCase() method converts a string to uppercase letters.

The toUpperCase() method does not change the original string.

**let text = "Hello World!";**

**let result = text.toUpperCase();**

# **valueOf()**

The valueOf() method returns the primitive value of a string.

The valueOf() method does not change the original string.

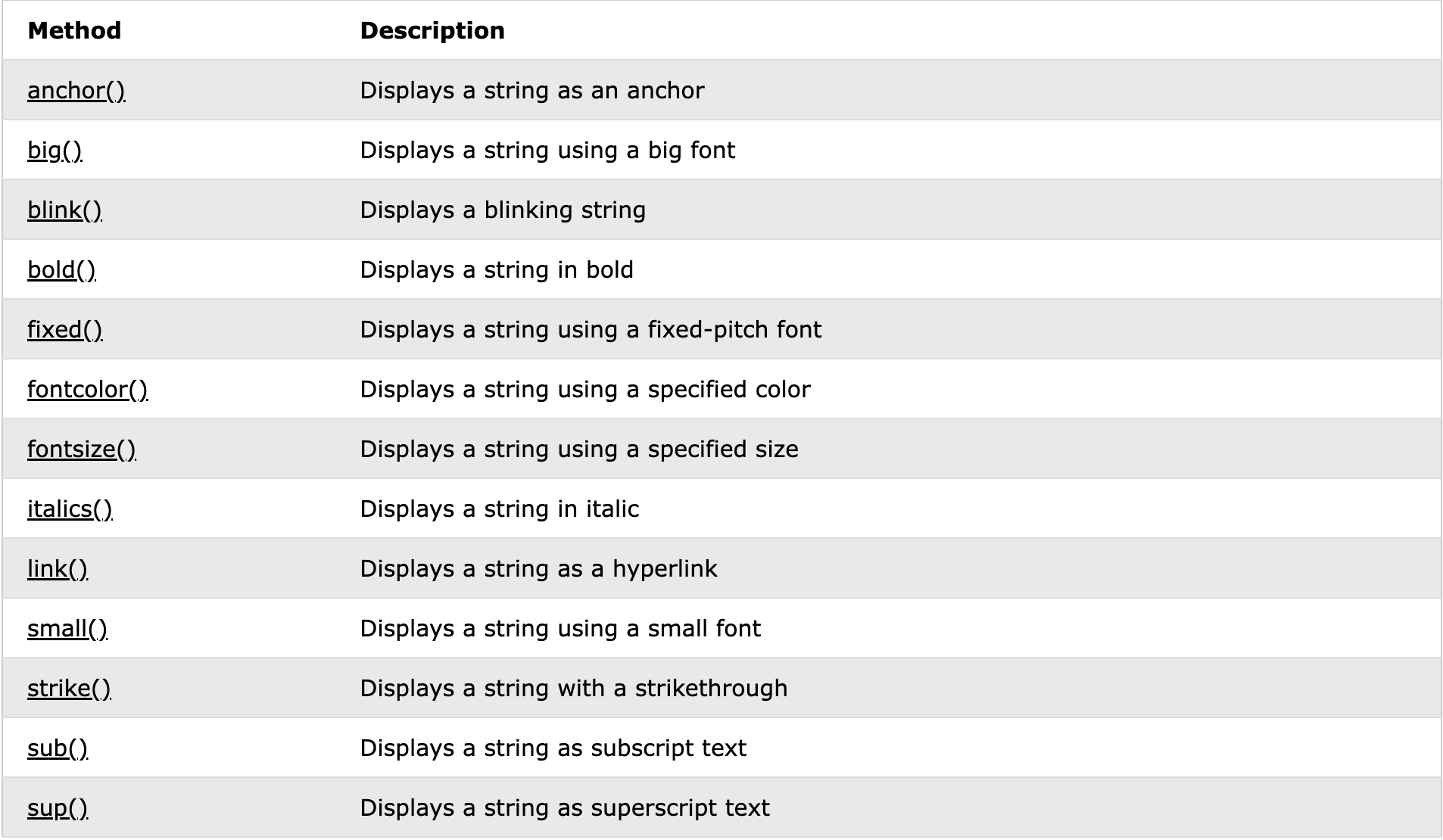
The valueOf() method can be used to convert a string object into a string.

**let text = "Hello World!";**

**let result = text.valueOf();**

* **String HTML Wrappers**

JS HTML Wrapper functions can wrapper up the string parameters for HTML context. For example, sup("test") will return "<sup>test</sup>".

****

* **anchor()**

The anchor method returns a string embedded in an <a> tag:

<a name="anchorname">string</a>

**let text = "Hello World!";**

**let result = text.anchor("Chapter 10");**

# **big()**

The big() method returns a string embedded in a <big> tag:

<big>string</big>

The <big> tag is not supported in HTML5.

**let text = "Hello World!";**

**let result = text.big();**

# **blink()**

The blink() method a string embedded in a <blink> tag:

<blink>string</blink>

The <blink> tag is not supported in HTML 5.

**let text = "Hello World!";**

**let result = text.blink();**

# **bold()**

The bold() method returns a string embedded in a <b> tag:

<b>string</b>

**let text = "Hello World!";**

**let result = text.bold();**

# **fixed()**

The fixed() method returns a string embedded in a <tt> tag:

<tt>string</tt>

The <tt> tag is not supported in HTML5.

**let text = "Hello World!";**

**let result = text.fixed();**

# **fontcolor()**

The fontcolor() method returns a string embedded in a <font> tag:

<font color="*colorvalue*">string</font>

The <font> tag is not supported in HTML5.

**let text = "Hello World!";**

**let result = text.fontcolor("green");**

# **fontsize()**

The fontsize() method returns a string embedded in a <font> tag:

<font size="size">string</font>

The <font> tag is not supported in HTML5.

**let text = "Hello World!";**

**let result = text.fontsize(6);**

# **italics()**

The italics() method returns a string embedded in an <i> tag:

<i>string</i>

**let text = "Hello World!";**

**let result = text.italics();**

# **link()**

The link() method returns a string embedded in an <a> tag:

<a href="url">string</a>

**let text = "Free Web Building Tutorials!";**

**let result = text.link("https://www.w3schools.com");**

# **small()**

The small() method returns a string embedded in a <small> tag:

<small>string</small>

**let text = "Hello World!";**

**let result = text.small();**

# **strike()**

The strike() method returns a string embedded in a <strike> tag:

<strike>string</strike>

The <strike> tag is not supported in HTML5.

**let text = "Hello World!";**

**let result = text.strike();**

# **sub()**

The sub() method returns a string embedded in a <sub> tag:

<sub>string</sub>

**let text = "Hello World!";**

**let result = text.sub();**

# **sup()**

The sup() method returns a string embedded in a <sup> tag:

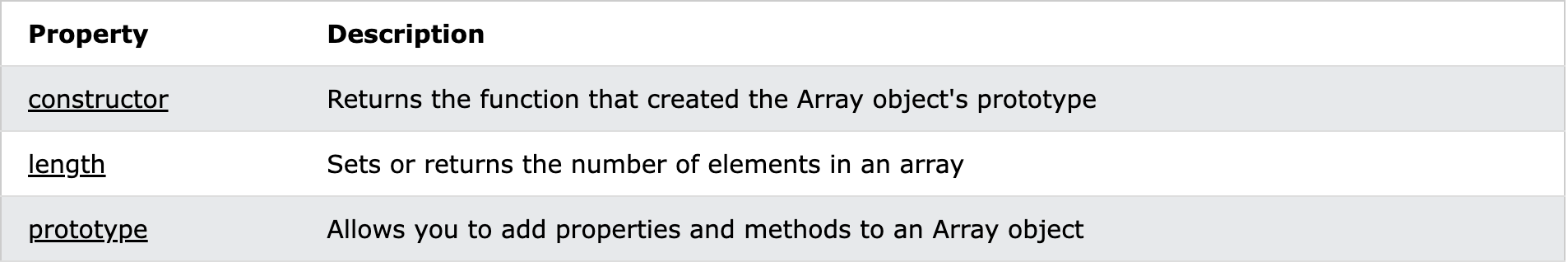
<sup>string</sup>

**let text = "Hello World!";**

**let result = text.sup();**

**JAVASCRIPT – Arrays**

## **Array Properties**

****

* **constructor**

The constructor property returns the function that created the Array prototype.

For JavaScript arrays the constructor property returns:

**function Array() { [native code] }**

* **length**

The length property sets or returns the number of elements in an array.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**let length = fruits.length;**

* **Prototype**

prototype allows you to add new properties and methods to arrays.

prototype is a property available with all JavaScript objects.

**Create a method that transforms array values into upper case:**

**Array.prototype.myUcase = function() {**

**for (let i = 0; i < this.length; i++) {**

**this[i] = this[i].toUpperCase();**

**}**

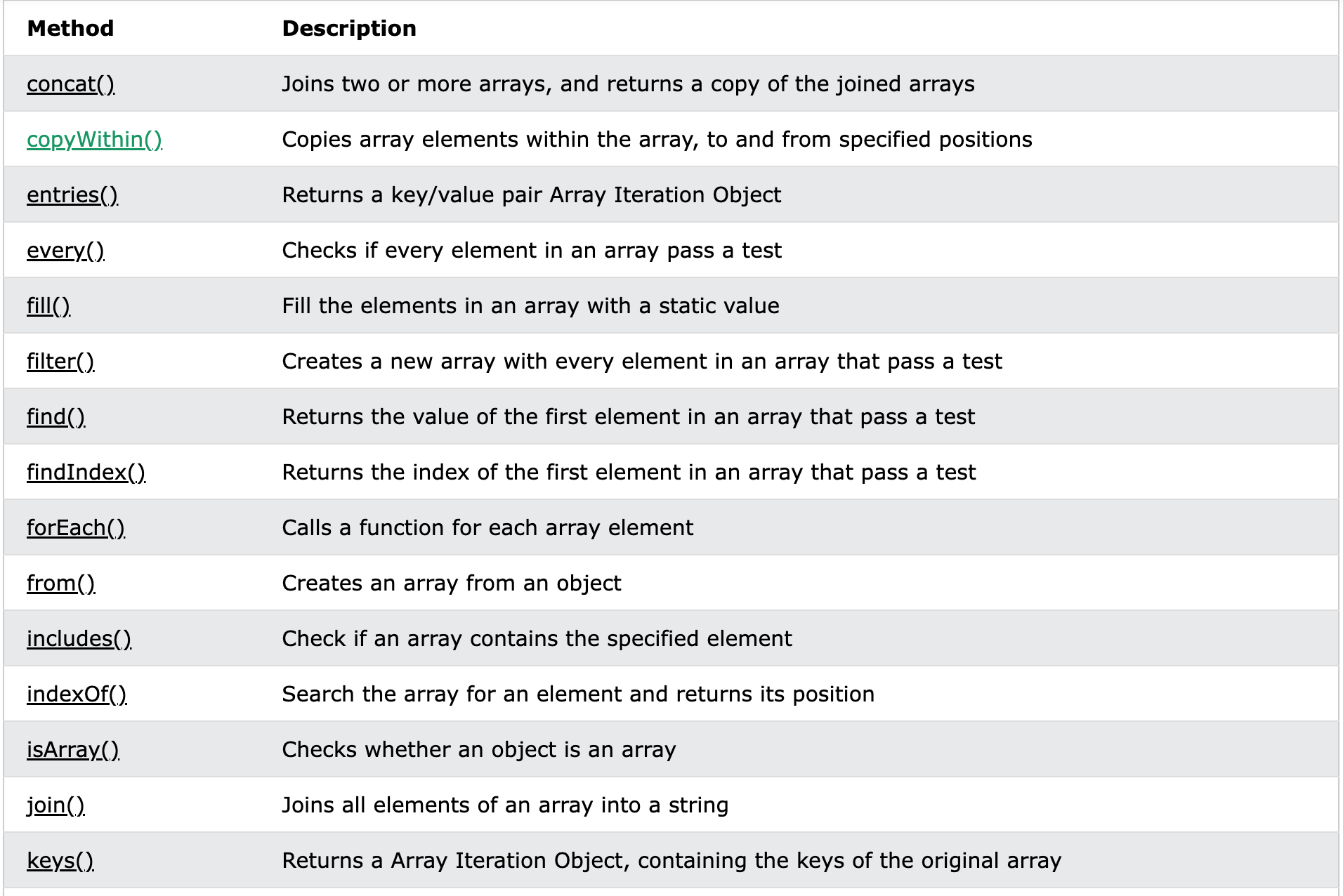
**};**

**Use the method on any array:**

**var fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.myUcase();**

* **Array Methods**

****

# **concat()**

The concat() method concatenates (joins) two or more arrays.

The concat() method returns a new array, containing the joined arrays.

The concat() method does not change the existing arrays.

**const arr1 = ["Cecilie", "Lone"];**

**const arr2 = ["Emil", "Tobias", "Linus"];**

**const arr3 = ["Robin"];**

**const children = arr1.concat(arr2,arr3);**

# **every()**

The every() method executes a function for each array element.

The every() method returns true if the function returns true for all elements.

The every() method returns false if the function returns false for one element.

The every() method does not execute the function for empty elements.

The every() method does not change the original array

**const ages = [32, 33, 16, 40];**

**ages.every(checkAge)**

**function checkAge(age) {**

**return age > 18;**

**}**

# **filter()**

The filter() method creates a new array filled with elements that pass a test provided by a function.

The filter() method does not execute the function for empty elements.

The filter() method does not change the original array.

**const ages = [32, 33, 16, 40];**

**const result = ages.filter(checkAdult);**

**function checkAdult(age) {**

**return age >= 18;**

**}**

# **forEach()**

The forEach() method calls a function for each element in an array.

The forEach() method is not executed for empty elements.

**const fruits = ["apple", "orange", "cherry"];**

**fruits.forEach(myFunction);**

# **indexOf()**

The indexOf() method returns the first index (position) of a specified value.

The indexOf() method returns -1 if the value is not found.

The indexOf() method starts at a specified index and searches from left to right.

By default the search starts at the first element and ends at the last.

Negative start values count from the last element (but still searches from left to right).

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**let index = fruits.indexOf("Apple");**

# **join()**

The join() method returns an array as a string.

The join() method does not change the original array.

Any separator can be specified. The default is comma (,)

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**let text = fruits.join();**

# **lastIndexOf()**

The lastIndexOf() method returns the last index (position) of a specified value.

The lastIndexOf() method returns -1 if the value is not found.

The lastIndexOf() starts at a specified index and searches from right to left.

By default the search starts at the last element and ends at the first.

Negative start values count from the last element (but still searches from right to left).

**const fruits = ["Apple", "Orange", "Apple", "Mango"];**

**let index = fruits.lastIndexOf("Apple");**

# **map()**

map() creates a new array from calling a function for every array element.

map() calls a function once for each element in an array.

map() does not execute the function for empty elements.

map() does not change the original array.

**const numbers = [65, 44, 12, 4];**

**const newArr = numbers.map(myFunction)**

**function myFunction(num) {**

**return num \* 10;**

**}**

# 

# **pop()**

The pop() method removes (pops) the last element of an array.

The pop() method changes the original array.

The pop() method returns the removed element.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.pop();**

# **push()**

The push() method adds new items to the end of an array.

The push() method changes the length of the array.

The push() method returns the new length.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.push("Kiwi", "Lemon");**

# **reduce()**

The reduce() method executes a reducer function for array elements.

The reduce() method returns a single value: the function's accumulated result.

The reduce() method does not execute the function for empty array elements.

The reduce() method does not change the original array.

**const numbers = [15.5, 2.3, 1.1, 4.7];**

**document.getElementById("demo").innerHTML = numbers.reduce(getSum, 0);**

**function getSum(total, num) {**

**return total + Math.round(num);**

**}**

# **reduceRight()**

The reduceRight() method executes a reducer function for each array element.

The reduceRight() method works from right to left.

The reduceRight() method returns a single value: the function's accumulated result.

The reduceRight() method does not execute the function for empty elements.

**const numbers = [175, 50, 25];**

**document.getElementById("demo").innerHTML = numbers.reduceRight(myFunc);**

**function myFunc(total, num) {**

**return total - num;**

**}**

# **reverse()**

The reverse() method reverses the order of the elements in an array.

The reverse() method overwrites the original array.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.reverse();**

# **shift()**

The shift() method removes the first item of an array.

The shift() method changes the original array.

The shift() method returns the shifted element.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.shift();**

# **slice()**

The slice() method returns selected elements in an array, as a new array.

The slice() method selects from a given *start*, up to a (not inclusive) given *end*.

The slice() method does not change the original array.

**const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];**

**const citrus = fruits.slice(1, 3);**

# **some()**

The some() method checks if any array elements pass a test (provided as a function).

The some() method executes the function once for each array element:

* If the function returns *true*, some() returns true and stops.
* If the function returns *false*, some() returns false and stops.

The some() method does not execute the function for empty array elements.

The some() method does not change the original array.

**const ages = [3, 10, 18, 20];**

**ages.some(checkAdult);**

**function checkAdult(age) {**

**return age > 18;**

**}**

# **sort()**

The sort() sorts the elements of an array.

The sort() overwrites the original array.

The sort() sorts the elements as strings in alphabetical and ascending order.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.sort();**

# **splice()**

The splice() method adds and/or removes array elements.

The splice() method overwrites the original array.

At position 2, add new items, and remove 1 item:

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.splice(2, 1, "Lemon", "Kiwi");**

# **toString()**

The toString() method returns a string with array values separated by commas.

The toString() method does not change the original array.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**let text = fruits.toString();**

# **unshift()**

The unshift() method adds new elements to the beginning of an array.

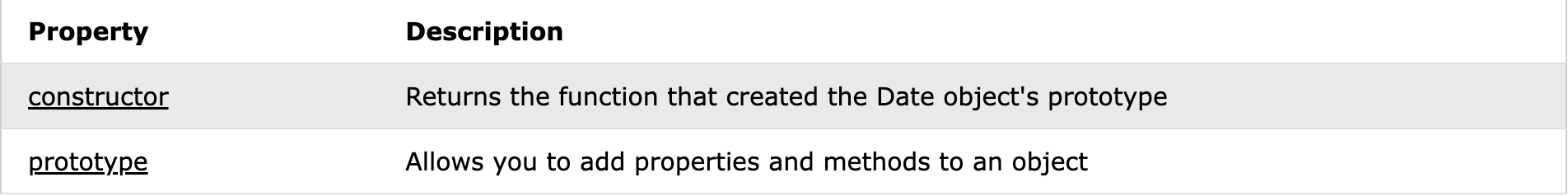
The unshift() method overwrites the original array.

**const fruits = ["Banana", "Orange", "Apple", "Mango"];**

**fruits.unshift("Lemon","Pineapple");**

**JAVASCRIPT – Date**

* **Date Properties**



* **constructor**

The constructor property returns the function that created the Date prototype.

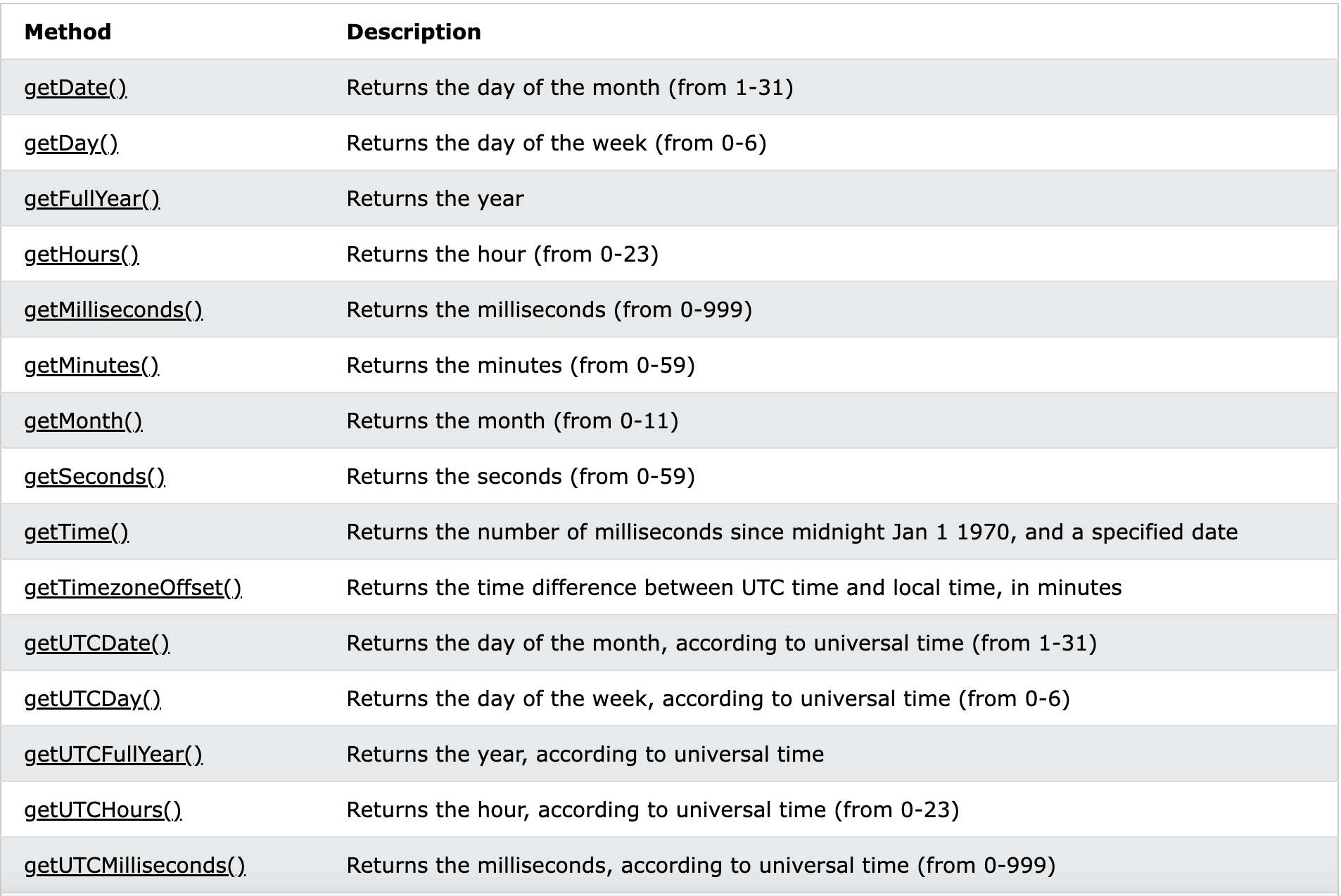
For JavaScript dates the constructor property returns:

function Date() { [native code] }

* **Prototype**

prototype allows you to add new properties and methods to dates.

prototype is a property available with all JavaScript objects.

* **Date Methods**
* **Date()**

When called as a function, it returns a string representation of the current date and time, exactly as new Date().toString() does.

const date1 = new Date('December 17, 1995 03:24:00');

* **getDate()**

The getDate() method returns the day of the month (1 to 31) of a date.

**const d = new Date();**

**let day = d.getDate();**

* **getDay()**

The getDay() method returns the day of the week (0 to 6) of a date.

Sunday = 0, Monday = 1, ...

* **getFullYear()**

getFullYear() returns the full year (4 digits) of a date.

**const d = new Date();**

**let year = d.getFullYear();**

* **getHours()**

getHours() returns the hour (0 to 23) of a date.

**const d = new Date();**

**let hour = d.getHours();**

* **getMilliseconds()**

getMilliseconds() returns the milliseconds (0 to 999) of a date.

**const d = new Date();**

**let ms = d.getMilliseconds();**

* **getMinutes ()**

getMinutes() returns the minutes (0 to 59) of a date.

**const d = new Date();**

**let minutes = d.getMinutes();**

* **getMonth ()**

getMonth() returns the month (0 to 11) of a date.

January =0, February = 1, ...

**const d = new Date();**

**let month = d.getMonth();**

* **getSeconds ()**

getSeconds() returns the seconds (0 to 59) of a date.

**const d = new Date();**

**let seconds = d.getSeconds();**

* **getTime ()**

getTime() returns the number of milliseconds since January 1, 1970 00:00:00.

**const d = new Date();**

**let time = d.getTime();**

* **getTimezoneOffset ()**

getTimezoneOffset() returns the difference between UTC time and local time.

getTimezoneOffset() returns the difference in minutes.

For example, if your time zone is GMT+2, -120 will be returned.

**const d = new Date();**

**let diff = d.getTimezoneOffset();**

* **getUTCDate ()**

getUTCDate() returns the day of the month (1 to 31) of a date object.

getUTCDate() returns the day according to UTC.

**const d = new Date();**

**let day = d.getUTCDate();**

* **getUTCDay ()**

getUTCDay() returns the day of the week (0 to 6) of a date.

getUTCDay() returns the day of the week according to universal time.

Sunday = 0, Monday = 1, …

**const d = new Date();**

**let day = d.getUTCDay();**

* **getUTCFullYear ()**

getUTCFullYear() returns the year of a date.

getUTCFullYear() returns the year according to UTC.

getUTCFullYear() returns four digits for years between 1000 and 9999.

**const d = new Date();**

**let year = d.getUTCFullYear();**

* **getUTCHours ()**

getUTCHours() returns the hour (0 to 23) of a date.

getUTCHours() returns the hour according to UTC.

**const d = new Date();**

**let hour = d.getUTCHours();**

* **getUTCMilliseconds ()**

getUTCMilliseconds() returns the milliseconds (0 to 999) of a date.

getUTCMilliseconds() returns the milliseconds according to UTC.

**const d = new Date();**

**let ms = d.getUTCMilliseconds();**

* **getUTCMinutes ()**

getUTCMinutes() returns the minutes (0 to 59) of a date.

getUTCMinutes() returns the minutes according to UTC.

**const d = new Date();**

**let minutes = d.getUTCMinutes();**

* **getUTCMonth ()**

getUTCMonth() returns the month (0 to 11) of a date.

getUTCMonth() returns the month according to UTC.

January = 0, February = 1, ...

**const d = new Date();**

**let month = d.getUTCMonth();**

* **getUTCSeconds ()**

getUTCSeconds() returns the seconds (0 to 59) of a date.

getUTCSeconds() returns the seconds according to UTC.

**const d = new Date();**

**let seconds = d.getUTCSeconds();**

* **getYear ()**

Javascript date getYear() method returns the year in the specified date according to universal time. The getYear is no longer used and has been replaced by the getYear method.

**Date.getYear()**

* **setDate ()**

setDate() sets the day of the month of a date.

**const d = new Date();**

**d.setDate(15);**

* **setFullYear ()**

setFullYear() sets the year of a date.

setFullYear() can also set month and day.

**const d = new Date();**

**d.setFullYear(2020);**

* **setHours ()**

setHours() sets the hour of a date.

setHours() can also set minutes, seconds and milliseconds.

**const d = new Date();**

**d.setHours(15);**

* **setMilliseconds ()**

setMilliseconds() sets the milliseconds of a date.

**const d = new Date();**

**d.setMilliseconds(192);**

# **setMinutes()**

setMinutes() sets the minutes of a date.

This method can also be used to set the seconds and milliseconds.

**const d = new Date();**

**d.setMinutes(17);**

* **setMonth ()**

The setMonth() method sets the month of a date object.

Note: January is 0, February is 1, and so on.

This method can also be used to set the day of the month.

**const d = new Date();**

**d.setMonth(4);**

* **setSeconds ()**

The setSeconds() method sets the seconds of a date object.

This method can also be used to set the milliseconds.

**const d = new Date();**

**d.setSeconds(35);**

* **setTime ()**

The setTime() method sets a date and time by adding or subtracting a specified

number of milliseconds to/from midnight January 1, 1970.

**const d = new Date();**

**d.setTime(1332403882588);**

* **setUTCDate ()**

The setUTCDate() method sets the day of a Date object, according to

UTC.

**const d = new Date();**

**d.setUTCDate(15);**

* **setUTCFullYear ()**

The setUTCFullYear() method sets the year of a date object, according to UTC.

**const d = new Date();**

**d.setUTCFullYear(2015);**

# **setUTCHours()**

The setUTCHours() method sets the hour of a date object, according to UTC.

It can also be used to set the minutes, seconds and milliseconds.

**const d = new Date();**

**d.setUTCHours(15);**

* **setUTCMilliseconds ()**

The setUTCMilliseconds() method sets the milliseconds (from 0 to 999) of a date object, according to UTC.

**const d = new Date();**

**d.setUTCMilliseconds(192);**

* **setUTCMinutes ()**

The setUTCMinutes() method sets the minutes of a date object, according to UTC.

**const d = new Date();**

**d.setUTCMinutes(17);**

# **setUTCMonth()**

The setUTCMonth() method sets the month (from 0 to 11) of a date object, according to UTC.

Note: January is 0, February is 1, and so on.

**const d = new Date();**

**d.setUTCMonth(4);**

# **setUTCSeconds()**

The setUTCSeconds() method sets the seconds of a date object, according to UTC.

It can also be used to set the milliseconds.

**const d = new Date();**

**d.setUTCSeconds(35);**

* **setYear ()**

The legacy setYear() method sets the year for a specified date according to local time.

**setYear(yearValue)**

# **toDateString()**

The toDateString() method returns the date (not the time) of a date object as a

string.

**const d = new Date();**

**let text = d.toDateString();**

* **toGMTString ()**

Javascript date toGMTString() method converts a date to a string, using Internet GMT conventions.

This method is no longer used and has been replaced by the toUTCString method.

**Date.toGMTString()**

* **toLocaleDateString ()**

The toLocaleDateString() method returns the date (not the time) of a date object as a string, using locale conventions.

**const d = new Date();**

**let text = d.toLocaleDateString();**

# **toLocaleTimeString()**

The toLocaleTimeString() method returns the time portion of a date object as a string, using locale conventions.

**const d = new Date();**

**let text = d.toLocaleTimeString();**

* **toLocaleFormat ()**

Javascript date toLocaleFormat() method converts a date to a string using the specified formatting.

Note − This method may not be compatible with all the browsers.

**Date.toLocaleFormat()**

* **toLocaleString ()**

The toLocaleString() method returns a Date object as a string, using locale settings.

The default language depends on the locale setup on your computer.

**const d = new Date();**

**let text = d.toLocaleString();**

* **toSource ()**

The toSource() method returns a string representing the source code of the object.

**Date.toSource()**

# **toString()**

The toString() method returns a date object as a string.

**const d = new Date();**

**let text = d.toString();**

# **toTimeString()**

The toTimeString() method returns the time portion of a date object as a string.

**const d = new Date();**

**let text = d.toTimeString();**

* **toUTCString ()**

The toUTCString() method returns a date object as a string, according to UTC.

Tip: The Universal Coordinated Time (UTC) is the time set by the World Time Standard.

Note: UTC time is the same as GMT time.

**const d = new Date();**

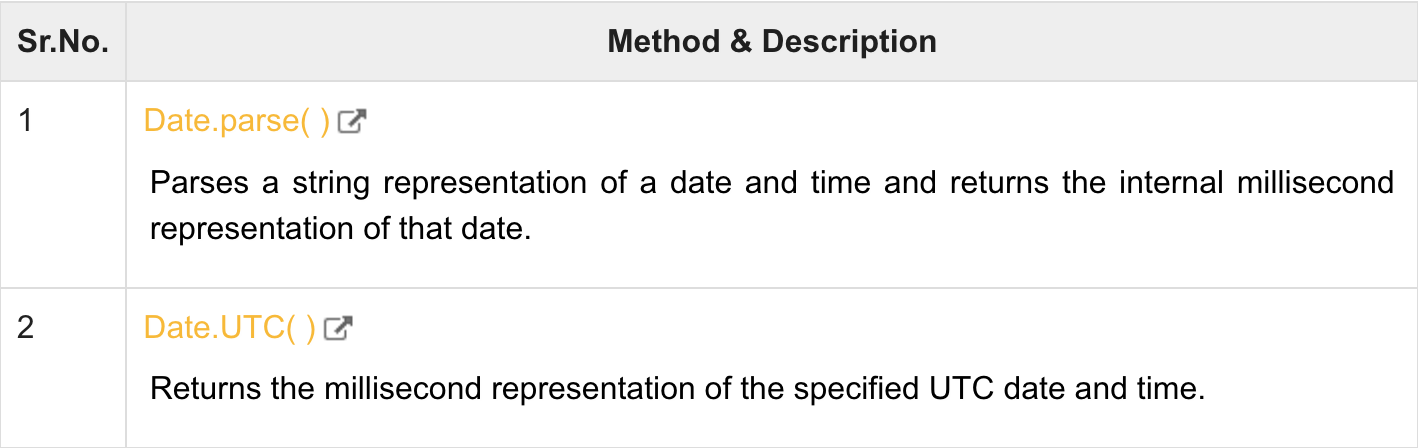
**let text = d.toUTCString();**

* **valeOf ()**

The valueOf() method returns the primitive value of a date object.

Note: The primitive value is returned as the number of millisecond since midnight January 1, 1970 UTC.

* **Date Static Methods**

In addition to the many instance methods listed previously, the Date object also defines two static methods. These methods are invoked through the Date() constructor itself.

* **Date.parse ( )**

Javascript date parse() method takes a date string and returns the number of milliseconds since midnight of January 1, 1970.

Date.parse(datestring)

* **Date.UTC ( )**

The Date.UTC() method returns the number of milliseconds between a specified date and midnight of January 1, 1970, according to UTC.

**let ms = Date.UTC(2020, 02, 30);**