

# Section Resources and Code Examples

## JavaScript Methods

```
querySelector()  
addEventListener()  
forEach()  
classList.add()  
classList.remove()  
parseInt()  
Number()  
createElement()  
append()
```

## Element Style

```
el.style["transform"] = prop + "(" + val + ")";  
el.style.opacity = 1;  
el.style.fontSize = '1.4em';  
el.style.padding = '10px';  
el.style.backgroundColor = 'black';  
el.style.color = 'white';
```

## Array Methods

### push()

**Add into end of array.**

### sort()

The `sort()` method sorts the elements of an array in place and returns the sorted array. The default sort order is ascending, built upon converting the elements into strings, then comparing their sequences of UTF-16 code units values.

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/sort](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort)

If `compareFunction(a, b)` returns less than 0, sort a to an index lower than b (i.e. a comes first).

If `compareFunction(a, b)` returns 0, leave a and b unchanged with respect to each other, but sorted with respect to all different elements. Note: the ECMAScript standard does not guarantee this behavior, thus, not all browsers (e.g. Mozilla versions dating back to at least 2003) respect this.

If `compareFunction(a, b)` returns greater than 0, sort b to an index lower than a (i.e. b comes first).

`compareFunction(a, b)` must always return the same value when given a specific pair of elements a and b as its two arguments. If inconsistent results are returned, then the sort order is undefined.

So, the compare function has the following form:

```
function compare(a, b) {  
  if (a is less than b by some ordering criterion) {  
    return -1;  
  }  
  if (a is greater than b by the ordering criterion) {  
    return 1;  
  }  
  // a must be equal to b  
  return 0;  
}
```

## slice()

The `slice()` method returns a shallow copy of a portion of an array into a new array object selected from start to end (end not included) where start and end represent the index of items in that array. The original array will not be modified.

```
const animals = ['ant', 'bison', 'camel', 'duck', 'elephant'];  
  
console.log(animals.slice(2));  
// expected output: Array ["camel", "duck", "elephant"]  
  
console.log(animals.slice(2, 4));  
// expected output: Array ["camel", "duck"]
```

```
console.log(animals.slice(1, 5));  
// expected output: Array ["bison", "camel", "duck", "elephant"]
```

## join()

The `join()` method creates and returns a new string by concatenating all of the elements in an array (or an array-like object), separated by commas or a specified separator string. If the array has only one item, then that item will be returned without using the separator.

```
const elements = ['Fire', 'Air', 'Water'];  
  
console.log(elements.join());  
// expected output: "Fire,Air,Water"  
  
console.log(elements.join(''));  
// expected output: "FireAirWater"  
  
console.log(elements.join('-'));  
// expected output: "Fire-Air-Water"  
  
var a = ['Wind', 'Water', 'Fire'];  
a.join();           // 'Wind,Water,Fire'  
a.join(', ');       // 'Wind, Water, Fire'  
a.join(' + ');      // 'Wind + Water + Fire'  
a.join('');         // 'WindWaterFire'
```

## parseInt()

The `parseInt()` function parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems).

```
parseInt('08') // 0, because '8' is not an octal digit.
```

## Date()

JavaScript Date objects represent a single moment in time in a platform-independent format. Date objects contain a Number that represents milliseconds since 1 January 1970 UTC.

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Date](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date)

## **Date.prototype.getTime()**

Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC. (Negative values are returned for prior times.)

## **Date.prototype.toISOString()**

Converts a date to a string following the ISO 8601 Extended Format.

```
var d = new Date(1993, 5, 28, 14, 39, 7);

console.log(d.toString());      // logs Mon Jun 28 1993 14:39:07 GMT-0600 (PDT)
console.log(d.toDateString()); // logs Mon Jun 28 1993
```

## **Random()**

The **Math.random()** function returns a floating-point, pseudo-random number in the range 0 to less than 1 (inclusive of 0, but not 1) with approximately uniform distribution over that range — which you can then scale to your desired range. The implementation selects the initial seed to the random number generation algorithm; it cannot be chosen or reset by the user.

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Math/random](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random)

```
function getRandomInt(max) {
  return Math.floor(Math.random() * Math.floor(max));
}

console.log(getRandomInt(3));
// expected output: 0, 1 or 2

console.log(getRandomInt(1));
// expected output: 0

console.log(Math.random());
// expected output: a number from 0 to <1
```

```
function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min) + min); //The maximum is exclusive and
the minimum is inclusive
}
```

## String Methods

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/String](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String)

### charAt()

The String object's charAt() method returns a new string consisting of the single UTF-16 code unit located at the specified offset into the string. let character = str.charAt(index)

```
const sentence = 'The quick brown fox jumps over the lazy dog.';

const index = 4;

console.log(`The character at index ${index} is ${sentence.charAt(index)}`);
// expected output: "The character at index 4 is q"

var anyString = 'Brave new world';
console.log("The character at index 0 is '" + anyString.charAt() + "'");
// No index was provided, used 0 as default

console.log("The character at index 0 is '" + anyString.charAt(0) + "'");
console.log("The character at index 1 is '" + anyString.charAt(1) + "'");
console.log("The character at index 2 is '" + anyString.charAt(2) + "'");
console.log("The character at index 3 is '" + anyString.charAt(3) + "'");
console.log("The character at index 4 is '" + anyString.charAt(4) + "'");
console.log("The character at index 999 is '" + anyString.charAt(999) + "'");

/*
The character at index 0 is 'B'
The character at index 0 is 'B'
The character at index 1 is 'r'
The character at index 2 is 'a'
The character at index 3 is 'v'
The character at index 4 is 'e'
The character at index 999 is ''
*/
```

### substr()

The `substr()` method returns a portion of the string, starting at the specified index and extending for a given number of characters afterwards.

```
str.substr(start[, length])
```

**start**

The index of the first character to include in the returned substring.

**length**

Optional. The number of characters to extract.

```
var aString = 'Mozilla';

console.log(aString.substr(0, 1)); // 'M'
console.log(aString.substr(1, 0)); // ''
console.log(aString.substr(-1, 1)); // 'a'
console.log(aString.substr(1, -1)); // ''
console.log(aString.substr(-3)); // 'lla'
console.log(aString.substr(1)); // 'ozilla'
console.log(aString.substr(-20, 2)); // 'Mo'
console.log(aString.substr(20, 2)); // ''

const str = 'Mozilla';

console.log(str.substr(1, 2));
// expected output: "oz"

console.log(str.substr(2));
// expected output: "zilla"
```

**toUpperCase()**

The `toUpperCase()` method returns the calling string value converted to uppercase (the value will be converted to a string if it isn't one).

```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toUpperCase());
// expected output: "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG."

console.log('alphabet'.toUpperCase()); // 'ALPHABET'
```

**trim()**

The `trim()` method removes whitespace from both ends of a string. Whitespace in this context is all the whitespace characters (space, tab, no-break space, etc.) and all the line terminator characters (LF, CR, etc.).

```
const greeting = '  Hello world!  ';

console.log(greeting);
// expected output: "  Hello world!  ";

console.log(greeting.trim());
// expected output: "Hello world!";

var orig = '  foo  ';
console.log(orig.trim()); // 'foo'

// Another example of .trim() removing whitespace from just one side.

var orig = 'foo  ';
console.log(orig.trim()); // 'foo'
```

## split()

The `split()` method divides a String into an ordered list of substrings, puts these substrings into an array, and returns the array. The division is done by searching for a pattern; where the pattern is provided as the first parameter in the method's call.

```
const str = 'The quick brown fox jumps over the lazy dog.';

const words = str.split(' ');
console.log(words[3]);
// expected output: "fox"

const chars = str.split('');
console.log(chars[8]);
// expected output: "k"

const strCopy = str.split();
console.log(strCopy);
// expected output: Array ["The quick brown fox jumps over the lazy dog."]
```

## getBoundingClientRect()

The `Element.getBoundingClientRect()` method returns a `DOMRect` object providing information about the size of an element and its position relative to the viewport.

The returned value is a DOMRect object which is the smallest rectangle which contains the entire element, including its padding and border-width. The left, top, right, bottom, x, y, width, and height properties describe the position and size of the overall rectangle in pixels. Properties other than width and height are relative to the top-left of the viewport.

```
domRect = element.getBoundingClientRect();
```

## CSS

```
translateX
transition: all ease-in-out .5s;

.timeInd li:before {
  content: "";
  position: absolute;
  width: 300px;
  height: 10px;
  top: 25px;
  left: -300px;
  z-index: -999;
  background-color: #ddd;
  transition: all ease-in-out .5s;
}
```