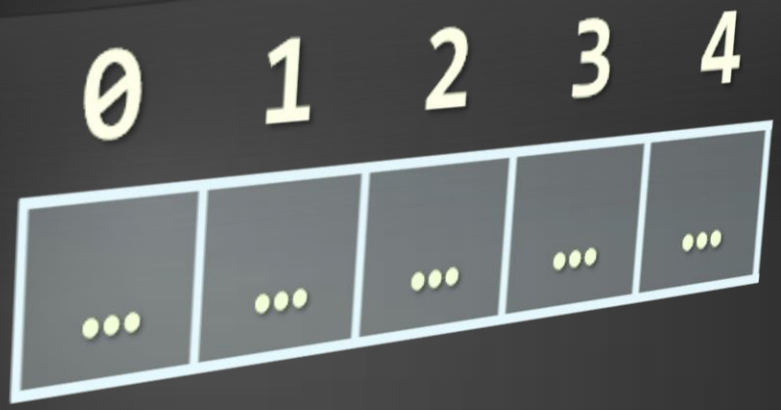
Telerik Academy



Arrays

Processing Sequences of Elements

JavaScript Fundamentals

Telerik Software Academy

http://academy.telerik.com

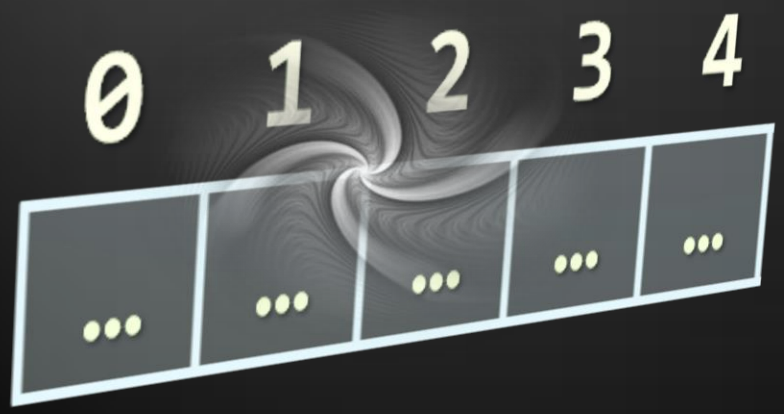
Telerik Academy

Table of Contents

1. Declaring and Creating Arrays
2. Accessing Array Elements
3. Processing Array Elements
4. Dynamic Arrays
5. Operations with arrays:
   - Concatenation
   - Slicing
   - Manipulation

Telerik Academy

Declaring and Creating Arrays



What are Arrays?

- An array is a sequence of elements
  - The order of the elements is fixed
  - Does not have fixed size
    - Can get the current length (`Array.length`)

Declaring Arrays

- Declaring an array in JavaScript (JS is typeless)

```
// Array holding integers
var numbers = [1, 2, 3, 4, 5];

// Array holding strings
var weekDays = ['Monday', 'Tuesday', 'Wednesday',
  'Thursday', 'Friday', 'Saturday', 'Sunday']

// Array of different types
var mixedArr = [1, new Date(), 'hello'];

// Array of arrays (matrix)
var matrix = [
  ['0,0', '0,1', '0,2'],
  ['1,0', '1,1', '1,2'],
  ['2,0', '2,1', '2,2']];
```

Telerik Academy

Declare and Initialize Arrays

- Initializing an array in JavaScript can be done in three ways:
  - Using `new Array(elements)`:

```
var arr = new Array(1, 2, 3, 4, 5);
```

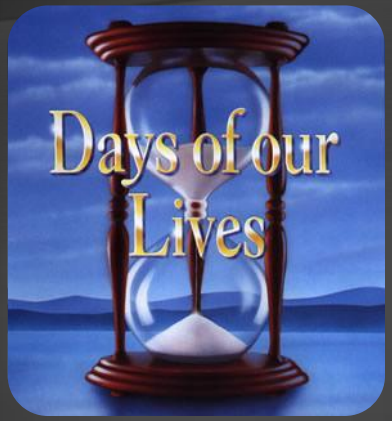
  - Using `new Array(initialLength)`:

```
var arr = new Array(10);
```

  - Using array literal (recommended):

```
var arr = [1, 2, 3, 4, 5];
```

Telerik Academy



Creating Arrays

Live Demo

Telerik Academy

Accessing Array Elements

Read and Modify Elements by Index

Telerik Academy

How to Access Array Element?

- Array elements are accessed using the square brackets operator [ ] (indexer)
  - Array indexer takes element's index as parameter in the range `0` … `length-1`
  - The first element has index `0`
  - The last element has index `length-1`
- Array elements can be retrieved and changed by the [ ] operator

Reversing an Array – Example

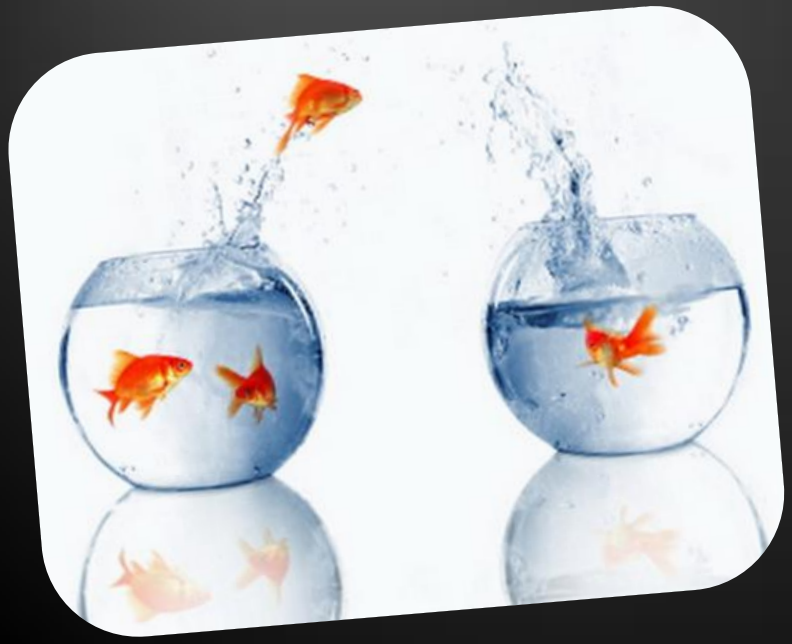
- Reversing the elements of an array

```
//always declare variables on the top of the scope!
var array, len, reversed, i;

array = [1, 2, 3, 4, 5];
reversed = [];

for (i = 0, len = array.length; i < len; i += 1) {
    reversed.push(array[length - i - 1]);
}
```

Telerik Academy

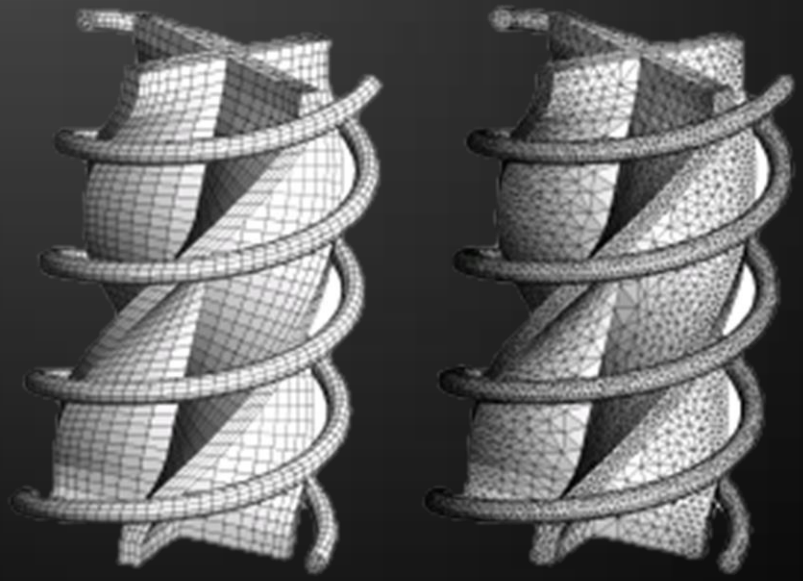
Reversing an Array

Live Demo



Telerik Academy

Processing Array Elements Using `for` and `for-in`



Processing Arrays: for Statement

- Use for loop to process an array when you need to keep track of the index
- In the loop body use the element at the loop index (array[index]):

```
var i, len;
for (i = 0, len = array.length; i < len; i += 1) {
   squares[i] = array[i] * array[i];
}
```

Processing Arrays Using for Loop – Examples

- Printing array of numbers in reversed order:

```
var arr, i, len;
arr = [1, 2, 3, 4, 5];
for (len = arr.length, i = len - 1; i >= 0; i -= 1) {
    console.log(arr[i]);
}
// Result: 5 4 3 2 1
```

- Initialize all array elements with their corresponding index number:

```
var i, len
for (i = 0, len = array.length; i < len; i += 1) {
    array[i] = i;
}
```

Processing Arrays: for-in

- How for-in loop works?

```
var index;
for (index in array)
```

  - index iterates through the indexes of the array
- Used when the indexes are unknown
  - All elements are accessed one by one
  - Order is not guaranteed

Example: Processing Arrays Using for-in Loop

- Print all elements of an array of strings:

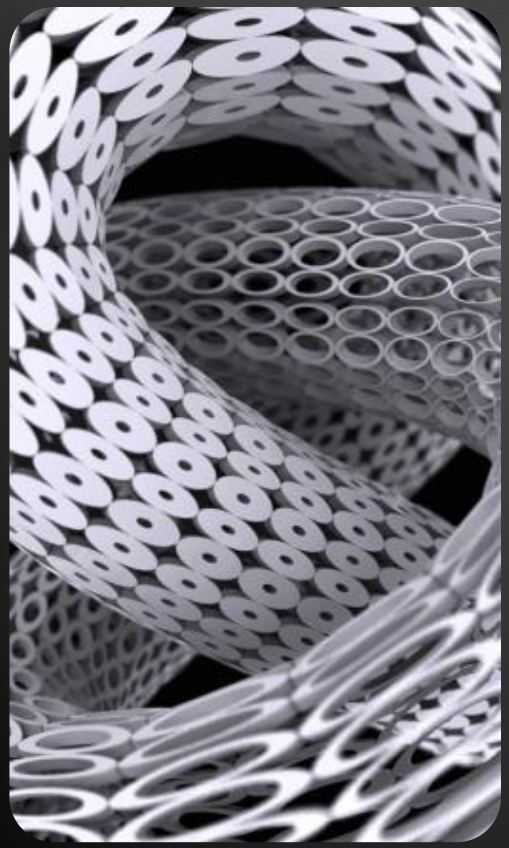
```
var capitals, i;
capitals = [
    'Sofia',
    'Washington',
    'London',
    'Paris'
];

for (i in capitals) {
    console.log(capitals[i]);
}
```

Telerik Academy

Processing Arrays

Live Demo

Telerik Academy



Dynamic Arrays

Dynamic Arrays

- All arrays in JavaScript are dynamic
  - Their size can be changed at runtime
  - New elements can be inserted to the array
  - Elements can be removed from the array
- Methods for array manipulation:
  - `[].push(element)`
    - Inserts a new element at the tail of the array
  - `[].pop()`
    - Removes the element at the tail
    - Returns the removed element

Dynamic Arrays (2)

- Methods for array manipulation (cont.)
  - `[].unshift(element)`
    - Inserts a new element at the head of the array
  - `[].shift()`
    - Removes and returns the element at the head

```
var numbers, tail, head;
numbers = [1, 2, 3, 4, 5];
console.log(numbers.join(', ')); // result: 1, 2, 3, 4, 5

tail = number.pop();          // tail = 5;
console.log(numbers.join(', ')); // result: 1, 2, 3, 4

number.unshift(0);
console.log(numbers.join(', ')); // result: 0, 1, 2, 3, 4

head = number.shift();        // head = 0;
console.log(numbers.join(', ')); // result: 1|2|3|4
```

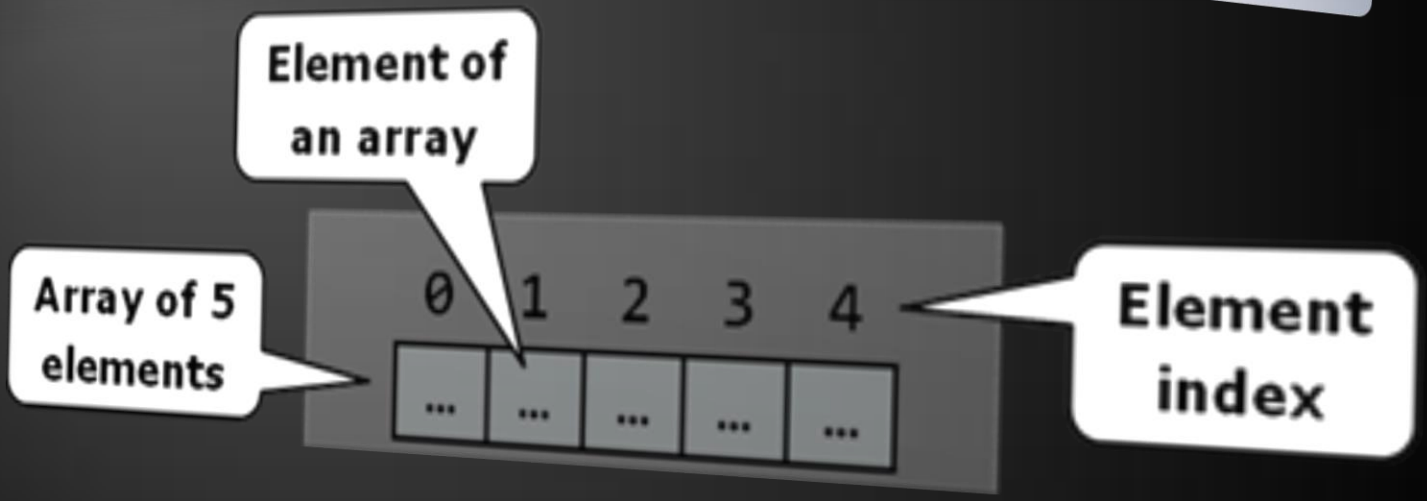
Telerik Academy

Dynamic Arrays

Live Demo

Telerik Academy

$$f(x, y) = x + y$$



Array Methods

Telerik Academy

[].reverse() and [].join()

- `[].reverse()` method
  - Returns a new arrays with elements in reversed order
- `[].join(separator)` method
  - Concatenates the elements of the array with a provided separator (by default none)
  - Returns a string

```
var numbers, reversed, result;
numbers = ['One', 'Two', 'Three'];
reversed = numbers.reverse();
result = reversed.join(', ');
console.log(result);                //prints 'Three, Two, One'
result = reversed.join(' then ');
console.log(result);     //prints 'Three then Two then One'
```

[].reverse() and [].join()

Live Demo

Telerik Academy

Concatenating Arrays

- `arr1.concat(arr2)`
  - Inserts the elements of `arr2` at the end of `arr1` and returns a new array
    - `arr1` and `arr2` remain unchanged!

```
var arr1, arr2, concatenated;

arr1 = [1, 2, 3];
arr2 = ['One', 'Two', 'Three'];
concatenated = arr1.concat(arr2);

console.log(arr1);
console.log(arr2);
console.log(concatenated);
```

Result:

```
[1, 2, 3]
['One', 'Two', 'Three']
[1, 2, 3, 'One', 'Two', 'Three']
```

Concatenating Arrays

Live Demo

Slicing Arrays

- `[].slice(from [, to])`
  - Returns a new array that is a shallow copy of a portion of the array
  - The new array contains the elements from indices `from` to `to` (excluding `to`)
  - Can be used to clone an array

```
var arr, portion, copy;
arr = [1, 2, 3, 4, 5];
portion = arr.slice(3, 6);
copy = arr.slice(0) //arr.slice()

console.log(arr);
console.log(portion);
console.log(copy);
```

Result:

```
[ 1, 2, 3, 4, 5 ]
[ 4, 5 ]
[ 1, 2, 3, 4, 5 ]
```

Slicing Arrays

Live Demo

Telerik Academy

Manipulating Arrays: [].splice()

- `[].splice(index, count, elements)`
  - Adds and removes elements from an array at the given position

```
var arr = [1, 2, 3, 4, 5, 6], p = 3, c = 2;
//remove c elements at position p
arr.splice(p, c);
console.log(arr);
//prints [ 1, 2, 3, 6 ]
//insert 'four', 'five', 'extra' at position p
arr.splice(p, 0, 'four', 'five', 'extra');
console.log(arr);
//prints [ 1, 2, 3, 'four', 'five', 'extra', 6 ]
//remove c elements at position p
//and insert 'x', 'y' and 'z' at the same position
arr.splice(p, c, 'x', 'y', 'z');
console.log(arr);
//prints [ 1, 2, 3, 'x', 'y', 'z', 'extra', 6 ]
```

Telerik Academy

Manipulating Arrays: [].splice()

Live Demo

Telerik Academy

Other Array Functions (2)

- `[].indexOf(element [, rightOf])`
  - Returns the index of the first match in the array
  - Returns `-1` is the element is not found
- `[].lastIndexOf(element, [leftOf])`
  - Returns the index of the first match in the array
  - Returns `-1` is the element is not found
- `indexOf()` and `lastIndexOf()` do not work in all browsers
  - Need to be shimmed

Telerik Academy

[].indexOf() and [].lastIndexOf()

Live Demo

Other Arrays Functions

- Arrays official documentation:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- Checking for array
  - `typeof([1, 2, 3])` → `object`
    - Not working
  - `Array.isArray([1, 2, 3])` → `true`
  - Supported on all modern browsers

Questions?

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com