Telerik Academy

ECMAScript 6

classes, generators, iterators, inheritance etc.

JavaScript OOP

Telerik Software Academy
http://academy.telerik.com

Telerik Academy

Table of Contents

- JavaScript History
  - The ECMAScript standard
- Using JavaScript.Next
  - Running JavaScript.Next in the browsers
    - Chrome Harmony, Firefox Nightly
    - Compiling to JS5 – Traceur, Babel
- ECMAScript 6 features
  - Variables: var, let, const
  - OOP: classes, inheritance, super, get/set
  - Functions: generators, iterators, arrow functions, comprehensions, for-of

Telerik Academy

Table of Contents (2)

- ECMAScript 6 features:
  - Data Structures: set/weakset, map/weakmap
  - Async operations: built-in promises
  - Modules: imports, exports, compitability
  - Objects: computed properties, shorthand properties, Object.is(), Object.assign(), proxies
  - Others: templates, Math and Number extensions

Telerik Academy

JavaScript History

JavaScript History

- JavaScript is a front-end scripting language developed by Netscape for dynamic content
  - Lightweight, but with limited capabilities
  - Can be used as object-oriented language
  - Embedded in your HTML page
  - Interpreted by the Web browser
- Client-side, mobile and desktop technology
- Simple and flexible
- Powerful to manipulate the DOM

Telerik Academy

Using JavaScript.next

Running JS next today

Telerik Academy

Using JavaScript.next

- There are a few ways to use JavaScript.next today
  - Enable tags in Chrome and Firefox
  - Compile to JavaScript 5 using Traceur or Babel
- A compatibility table for ES6 support can be found at https://kangax.github.io/compat-table/es6/

Telerik Academy

ES6 Variables

ES6 Variables

- ES6 introduces new ways to declare variables:
  - let – creates a scope variable
    - Accessible only in its scope

```
for(let number of [1, 2, 3, 4]){
  console.log(number);
}
//accessing number here throws exception
```

  - const – creates a constant variable
    - Its value is read-only and cannot be changed

```
const MAX_VALUE = 16;
MAX_VALUE = 15; // throws exception
```

Telerik Academy

ES6 Variables

Live Demo

Telerik Academy

for-of loop

For-of loop

- The for-of loop iterates over the values
  - Of an array

```
let sum = 0;
for(let number of [1, 2, 3])
  sum+= number;
```

  - Of An iteratable object

```
function* generator(maxValue){
  for(let i = 0; i < maxValue; i+=1){
    yield i;
  }
}
let iter = generator(10);
for(let val of iter()){
  console.log(val);
}
```

Telerik Academy

For-of loop

Live Demo

Telerik Academy

Templated String

Telerik Academy

Templated Strings in ES6

- ES6 supports templated strings
  - i.e. strings with placeholders:

```
let people = [new Person('Doncho', 'Minkov'), … ];
for(let person of people){
   log(`Fullname: ${person.fname} ${person.lname}`);
}
```

  - Templates escape the strings
    - They do not call eval

Telerik Academy

Classes and Inheritance

The way of OOP in ES6

Telerik Academy

Classes and Inheritance in ES6

- ES6 introduces classes and a way to create classical OOP

```
class Person extends Mammal{
  constructor(fname, lname, age){
    super(age);
    this._fname = fname;
    this._lname = lname;
  }
  get fullname() {
    //getter property of fullname
  }
  set fullname(newfullname) {
    //setter property of fullname
  }
  // more class members…
}
```

Telerik Academy

Classes and Inheritance in ES6

- ES6 introduces classes and a way to create classical OOP

```
class Person extends Mammal{
  constructor(fname, lname, age){
    super(age);
    this._fname = fname;
    this._lname = lname;
  }
  get fullname() {
    //getter property of fullname
  }
  set fullname(newfullname) {
    //setter property of fullname
  }
  // more class members…
}
```

Constructor of the class

Telerik Academy

Classes and Inheritance in ES6

- ES6 introduces classes and a way to create classical OOP

```
class Person extends Mammal{
  constructor(fname, lname, age){
    super(age);
    this._fname = fname;
    this._lname = lname;
  }
  get fullname() {
    //getter property of fullname
  }
  set fullname(newfullname) {
    //setter property of fullname
  }
  // more class members…
}
```

Constructor of the class

Getters and setters

Telerik Academy

Classes and Inheritance in ES6

Live Demo

Telerik Academy

Arrow Functions

Also called LAMBDA expressions

Arrow Functions

- Arrow functions easify the creation of functions:

Arrow Functions

- Arrow functions easify the creation of functions:

```
numbers.sort(function(a, b){
  return b - a;
});
```

Arrow Functions

- Arrow functions easify the creation of functions:

```
numbers.sort(function(a, b){
  return b - a;
});
```

Becomes

```
numbers.sort((a, b) => b - a);
```

Telerik Academy

Arrow Functions

- Arrow functions easify the creation of functions:

```
numbers.sort(function(a, b){
  return b - a;
});
```

Becomes

```
numbers.sort((a, b) => b - a);
```

```
var fullnames =
    people.filter(function (person) {
      return person.age >= 18;
    }).map(function (person) {
      return person.fullname;
    });
```

Arrow Functions

- Arrow functions easify the creation of functions:

```
numbers.sort(function(a, b){
  return b - a;
});
```

Becomes

```
numbers.sort((a, b) => b - a);
```

```
var fullnames =
   people.filter(function (person) {
     return person.age >= 18;
   }).map(function (person) {
     return person.fullname;
   });
```

Becomes

```
var fullnames2 =
  people.filter(p => p.age >= 18)
    .map(p => p.fullname);
```

Telerik Academy

Arrow Functions

Live Demo

Telerik Academy

Object Literals

Object Literals

- ES6 adds a new feature (rule) to the way of defining properties:
  - Instead of

```
let name = 'Doncho Minkov',
    age = 25;
let person = {
  name: name,
  age: age
};
```

  - We can do just:

```
let name = 'Doncho Minkov';
let person = {
  name,
  age
};
```

Telerik Academy

Object Literals

Live Demo

Telerik Academy

Destructuring Assignments

Telerik Academy

Destructuring Assignments

- Destructuring assignments allow to set values to objects in an easier way:
  - Destructuring assignments with arrays:

```
var [a,b] = [1,2]; //a = 1, b = 2
var [x, , y] = [1, 2, 3] // x = 1, y = 3
var [first, second, ...rest] = people;
```

  - Swap values: `[x, y] = [y, x]`
  - Result of method:

```
function get(){ return [1, 2, 3]; }
var [x, y] = get();
```

Telerik Academy

Destructuring Assignments

- Destructuring assignments allow to set values to objects in an easier way:
  - Destructuring assignments with objects:

```
var person = {
  name: 'Doncho Minkov',
  address: {
    city: 'Sofia',
    street: 'Aleksander Malinov'
  }
};

var {name, address: {city}} = person;
```

Telerik Academy

Destructuring Assignments

Live Demo

Telerik Academy

Maps and Sets

Telerik Academy

Maps and Sets

- ES6 supports maps and sets natively
  - They do pretty much the same as associative arrays, but in cleaner way:

```
let names = new Set();
names.add('Doncho');
names.add('Nikolay');
names.add('Ivaylo');
names.add('Evlogi');
names.add('Doncho'); // won't be added
```

Telerik Academy

Maps and Sets

Live Demo

Telerik Academy

ES6 Modules

ES6 Modules

- ES6 supports modules
  - A way to write JavaScript in different files
    - Each file has its own scope (not the global)
    - Each file decides what to export from its module
  - Export the objects you want from a module:

persons.js

```
module.exports = {
    Person: Person,
    Mammal: Mammal
}
```

  - To use the module in another file:

```
import classes from './persons'
import {Mammal, Person} form '.persons'
```

Telerik Academy

ES6 Modules

Live Demo

Telerik Academy

JavaScript.next

http://academy.telerik.com