JavaScript + Angular

Part 3

Modules, ES6

Author: Andrey Kucherenko

Modules

```
var names = ["Sunday", "Monday", "Tuesday", "Wednesday"
             "Thursday", "Friday", "Saturday"];
function dayName(number) {
  return names[number];
}

console.log(dayName(1));
```

```
var epam = epam || {};

epam.project = epam.project || {};

epam.project.names = ["Sunday", "Monday", "Tuesday",
    "Wednesday", "Thursday", "Friday", "Saturday"];

epam.project.dayName = function dayName(number) {
  return epam.project.names[number];
}

console.log(epam.project.dayName(1));
```

```
var module = (function () {

    var names = ["Sunday", "Monday", "Tuesday",
        "Wednesday", "Thursday", "Friday", "Saturday"];

    function dayName(number) {
      return names[number];
    }

    return dayName;

})();
```

```
var myModule = (function (otherModule) {

    var names = ["Sunday", "Monday", "Tuesday",
        "Wednesday", "Thursday", "Friday", "Saturday"];

    function dayName(number) {
      return names[number];
    }
    console.log(otherModule.method());

    return dayName;

})(otherModule);

var myModule = (function (otherModule) {
 //...
})(otherModule || {});
```

```
define('myModule',
    ['foo', 'bar'],
    function ( foo, bar ) {
        var privateVAr = 'hello';
        function private() {

        }
        var myModule = {
            methos: function(){
                console.log('Hello, JS');
            }
        }

        return myModule;
});

require(['foo', 'bar'], function ( foo, bar ) {
        // rest of your code here
        foo.doSomething();
});
```

```
var lib = require('package/lib');
var myModule = require('./src/myModule');


function foo(){
    lib.log('hello js!');
}

exports.foo = foo;

module.exports = {
    //.. methods for export
};
```

```
//  lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

//  someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))

//  otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))


//  lib/mathplusplus.js
export * from "lib/math"
export var e = 2.71828182846
export default (x) => Math.exp(x)

//  someApp.js
import exp, { pi, e } from "lib/mathplusplus"
console.log("e^{π} = " + exp(pi))
```

```
import { load } from 'store/customer';
import when from 'when';

export default function (id) {
    return when(id).then(load);
};
```

https://www.npmjs.com/

```
npm install lodash --save // --save-dev

npm run build // run script from scripts section

npm init

npm untinstall lodash --save
```

package.json

Tasks

- git clone https://github.com/kucherenko/js-classes.git
- cd js-classes/lesson-3
- npm install
- Create new module and export function hello()
- Import created module in other file
- Install & import module from npm
- Init you own package with npm init and install jquery and angular as dependencies
- Add script for convert ES6 to JS
- Add script for run http server
- Add convert from CommonJS to browser JS

ES6/2015

```
const Z = 5;
let z, x, a = 5;

if (a) {
    let b = 5;
}

[z, x] = ['z', 'x']

let [, , js] = "hello my js".split(' ');
let [h, ...all] = "hello my js".split(' ');

let [a=5] = [];

let options = {
  width: 300,
  height: 500,
  arr: [1, 2]
};
let {width, height} = options;
let {width: w, height: h, name="Andrey", arr: [a1, a2]} = options;
```

Task

- Create two vars
- Swap values of vars
- Create constant and try to redefine it
- Get fields from array as values

```
function a(p = 3) {} // a.name == "a"

function b(...all) {}

if (true) {
  z();
  function z() {
    alert("!!!");
  }
}
z();


let inc = x => x+1;

let sum = (a, b) => {
    return a + b; // no own this, no arguments
}
```

Tasks

- Create function, add arguments with default value
- Create arrow function
- Get all arguments to array
- Create function inside block and try call it outside

```
let str = `ololo`;

let str = `ololo

    ololo

    olol`;

let zz = `${str} hello`;
```

Tasks

- Create multi-line string
- Create template with calculation in string

```
class User {

  constructor(name) {
    console.log(super);
    this.name = name;
  }

  say() {
    alert(this.name);
  }

  static createGuest() {
    return new User();
  }

}

let User = class {
  say() { alert('!!!'); }
};

new User().say();

class A extends B {}
```

Tasks

- Create two classes
- Make inheritance

```
let name = "Adam";
let isAdmin = true;

let user = {
  name,
  isAdmin
};

let prop = 'role';
let user = {
  [prop]: "admin"
};


let user = {
  get role() {
    return `SuperUser`;
  }
};

alert( user.role );

let sym = Symbol();
alert( typeof sym ); // symbol
```

Tasks

- Make getter and setter
- Make object from variables
- Create new Symbol and output it

```
let arr = ["a", 'b'];
for (let value of arr) {
  alert(value);
}

let myIterator = {
  from: 1,
  to: 5
}

myIterator[Symbol.iterator] = function() {

  let current = this.from;
  let last = this.to;

  return {
    next() {
      if (current <= last) {
        return {
          done: false,
          value: current++
        };
      } else {
        return {
          done: true
        };
      }
    }
  }
};
alert(...myIterator)
for (let num of myIterator) {
  alert(num);
}
```

Tasks

- Create iterator for iterate string by two symbols

```
let map = new Map();

map.set('1', 'string');
map
    .set(1, 'number')
    .set(true, 'boolean');


alert( map.get(1)   );
alert( map.get('1') );

alert( map.size );

map.keys();
map.values();
map.entries();

let map = new Map([
  ['1',  'str1'],
  [1,    'num1'],
  [true, 'bool1']
]);

map.has('1');
map.delete('1');
map.clear();
```

Tasks

- Create Map with different keys
- Try to create Map with NaN as key
- Delete NaN key
- Clear Map

```
let set = new Set();

set.add(1);
set.add(2);
set.add(3);
set.add(3);
set.add(3);

alert( set.size ); // 3

set.forEach( item => alert(item) );

set.clear();
set.delete(1);
set.has(3);
```

Tasks

- Create Set
- Try to set two equals value
- Iterate Set

```
let arr = [
  {test: '1'},
  {test: '2'},
  {test: '3'}
];

let weakSet = new WeakSet();//WeakMap

weakSet.add(arr[0]);
weakSet.add(arr[1]);
weakSet.add(arr[2]);

delete arr[0];

alert(weakSet.size)
```

Tasks

- Create WeakMap and WeakSet with links to object
- Remove links
- Check WeakMap and WeakSet for removed keys

```
var promise =
    new Promise(function(resolve, reject) {

    makeAsyncAction(
        success => resolve({hello: 'js'})
        error => reject({error: "MyError"})
    );

});

promise.then(success, error);

promise.catch(errorHandler);

Promise.race(array);
Promise.all(array);
```

Tasks

- Create promise for setTimeout
- Try to reject and resolve promise one by one
- Throw new Error in callback

```
//Generator
function* gen() {
  yield 'a';
  yield 'b';
  return 'c';
}
let generator = gen();
let one = generator.next();
```

Tasks

- Make your own iterator based on generators

```
<!DOCTYPE html>
<html ng-app>
  <head>
    <meta charset="utf-8">
    <title>JS+Angular Classes: Lesson 3</title>
    <script src="lib/boundle.js" charset="utf-8"></script>
  </head>
  <body>
    <h1>JS+Angular Classes: Lesson 3</h1>
    <p>{{'Hello' + ' JS!'}}</p>
  </body>
</html>
```

```
import 'angular';
```

```
<!DOCTYPE html>
<html ng-app="app">
  <head>
    <meta charset="utf-8">
    <title>JS+Angular Classes: Lesson 3</title>
    <script src="lib/boundle.js" charset="utf-8"></script>
  </head>
  <body ng-controller="MyConroller">
    <h1>JS+Angular Classes: Lesson 3</h1>
    <p>{{'Hello ' + framework + 'JS!'}}</p>
  </body>
</html>
```

```
import * as angular from 'angular';


angular.module('app', [
    //.. list of dependencies
]).controller('MyConroller', ($scope) => {
  $scope.framework = 'Angular';
  console.log($scope);
});
```

Tasks

- Create angular.js application
- Run html file with angular app
- Create different angular modules
- Create dependencies between modules
- Use different html parts for different apps

Pet Project