HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Section 7: Get Ready for the Future: ES6 / ES2015</title>

    <style>
      .box {
        width: 200px;
        padding: 25px 80px;
        text-align: center;
        font-size: 30px;
        margin-top: 30px;
      }

      .green { background-color: green; }
      .blue { background-color: dodgerblue; }
      .orange { background-color: orangered; }

    </style>

  </head>

  <body>
    <h1>Section 7: Get Ready for the Future: ES6 / ES2015</h1>

    <div class="box green">I'm green!</div>
    <div class="box blue">I'm blue!</div>
    <div class="box orange">I'm orange!</div>

    <script src="polyfill.min.js"></script>
    <script src="script-transpiled.js"></script>
  </body>
</html>
```

JS-script.js

```
// Lecture: let and const
```

```
/*
```

// ES5

```
var name5 = 'Jane Smith';
var age5 = 23;
name5 = 'Jane Miller';
console.log(name5);
```

// ES6

```
const name6 = 'Jane Smith';
let age6 = 23;
name6 = 'Jane Miller';
console.log(name6);
```

// ES5

```
function driversLicence5(passedTest) {

    if (passedTest) {
        console.log(firstName);
        var firstName = 'John';
        var yearOfBirth = 1990;
    }


    console.log(firstName + ', born in ' + yearOfBirth + ', is now officially allowed to drive a car.');
}

driversLicence5(true);
```

// ES6

```
function driversLicence6(passedTest) {

    //console.log(firstName);
    let firstName;
    const yearOfBirth = 1990;

    if (passedTest) {
        firstName = 'John';
    }

    console.log(firstName + ', born in ' + yearOfBirth + ', is now officially allowed to drive a car.');
```

```
}

driversLicence6(true);



var i = 23;

for (var i = 0; i < 5; i++) {
    console.log(i);
}

console.log(i);
*/
```

// Lecture: Blocks and IIFEs

```
/*
// ES6

{
    const a = 1;
    let b = 2;
    var c = 3;
}

//console.log(a + b);
console.log(c);


// ES5

(function() {
    var c = 3;
})();

//console.log(c);
*/


// Lecture: Strings
```

```
/*
let firstName = 'John';
let lastName = 'Smith';
const yearOfBirth = 1990;

function calcAge(year) {
    return 2016 - year;
}

// ES5
console.log('This is ' + firstName + ' ' + lastName + '. He was born in ' + yearOfBirth + '. Today, he is ' + calcAge(yearOfBirth) + ' years old.');

// ES6
console.log(`This is ${firstName} ${lastName}. He was born in ${yearOfBirth}. Today, he is ${calcAge(yearOfBirth)} years old.`);


const n = `${firstName} ${lastName}`;
console.log(n.startsWith('j'));
console.log(n.endsWith('Sm'));
console.log(n.includes('oh'));
console.log(`${firstName} `.repeat(5));
*/
```

// Lecture: Arrow functions

```
/*
const years = [1990, 1965, 1982, 1937];

// ES5
var ages5 = years.map(function(el) {
    return 2016 - el;
});
console.log(ages5);


// ES6
let ages6 = years.map(el => 2016 - el);
console.log(ages6);

ages6 = years.map((el, index) => `Age element ${index + 1}: ${2016 - el}.`);
console.log(ages6);
```

```
ages6 = years.map((el, index) => {
    const now = new Date().getFullYear();
    const age = now - el;
    return `Age element ${index + 1}: ${age}.`
});
console.log(ages6);
*/
```

// Lecture: Arrow functions 2

```
/*
// ES5

var box5 = {
    color: 'green',
    position: 1,
    clickMe: function() {

        var self = this; document.querySelector('.green').addEventListener('click', function() {
            var str = 'This is box number ' + self.position + ' and it is ' + self.color;
            alert(str);
        });
    }
}
//box5.clickMe();


// ES6

const box6 = {
    color: 'green',
    position: 1,
    clickMe: function() {
        document.querySelector('.green').addEventListener('click', () => {
            var str = 'This is box number ' + this.position + ' and it is ' + this.color;
            alert(str);
        });
    }
}
box6.clickMe();
```

```
const box66 = {
    color: 'green',
    position: 1,
    clickMe: () => {
        document.querySelector('.green').addEventListener('click', () => {
            var str = 'This is box number ' + this.position + ' and it is ' + this.color;
            alert(str);
        });
    }
}
box66.clickMe();


function Person(name) {
    this.name = name;
}
```

// ES5

```
Person.prototype.myFriends5 = function(friends) {

    var arr = friends.map(function(el) {
       return this.name + ' is friends with ' + el;
    }.bind(this));

    console.log(arr);
}

var friends = ['Bob', 'Jane', 'Mark'];
new Person('John').myFriends5(friends);
```

// ES6

```
Person.prototype.myFriends6 = function(friends) {

    var arr = friends.map(el => `${this.name} is friends with ${el}`);

    console.log(arr);
}

new Person('Mike').myFriends6(friends);
```

*/

// Lecture: Destructuring

```
/*
// ES5

var john = ['John', 26];
//var name = john[0];
//var age = john[1];


// ES6

const [name, age] = ['John', 26];
console.log(name);
console.log(age);

const obj = {
    firstName: 'John',
    lastName: 'Smith'
};

const {firstName, lastName} = obj;
console.log(firstName);
console.log(lastName);

const {firstName: a, lastName: b} = obj;
console.log(a);
console.log(b);



function calcAgeRetirement(year) {
    const age = new Date().getFullYear() - year;
    return [age, 65 - age];
}


const [age2, retirement] = calcAgeRetirement(1990);
console.log(age2);
console.log(retirement);
*/
```

// Lecture: Arrays

```
/*
const boxes = document.querySelectorAll('.box');

//ES5

var boxesArr5 = Array.prototype.slice.call(boxes);
boxesArr5.forEach(function(cur) {
    cur.style.backgroundColor = 'dodgerblue';
});

//ES6

const boxesArr6 = Array.from(boxes);
Array.from(boxes).forEach(cur => cur.style.backgroundColor = 'dodgerblue');


//ES5

for(var i = 0; i < boxesArr5.length; i++) {

    if(boxesArr5[i].className === 'box blue') {
        continue;
    }

    boxesArr5[i].textContent = 'I changed to blue!';

}


//ES6

for (const cur of boxesArr6) {
    if (cur.className.includes('blue')) {
        continue;
    }
    cur.textContent = 'I changed to blue!';
}
```

```
//ES5

var ages = [12, 17, 8, 21, 14, 11];

var full = ages.map(function(cur) {
   return cur >= 18;
});
console.log(full);

console.log(full.indexOf(true));
console.log(ages[full.indexOf(true)]);


//ES6

console.log(ages.findIndex(cur => cur >= 18));
console.log(ages.find(cur => cur >= 18));
*/
```

// Lecture: Spread operator

```
/*
function addFourAges (a, b, c, d) {
   return a + b + c + d;
}

var sum1 = addFourAges(18, 30, 12, 21);
console.log(sum1);

//ES5

var ages = [18, 30, 12, 21];
var sum2 = addFourAges.apply(null, ages);
console.log(sum2);

//ES6

const sum3 = addFourAges(...ages);
console.log(sum3);
```

```
const familySmith = ['John', 'Jane', 'Mark'];
const familyMiller = ['Mary', 'Bob', 'Ann'];
const bigFamily = [...familySmith, 'Lily', ...familyMiller];
console.log(bigFamily);


const h = document.querySelector('h1');
const boxes = document.querySelectorAll('.box');
const all = [h, ...boxes];

Array.from(all).forEach(cur => cur.style.color = 'purple');
*/
```

// Lecture: Rest parameters

```
/*
```

//ES5

```
function isFullAge5() {
    //console.log(arguments);
    var argsArr = Array.prototype.slice.call(arguments);

    argsArr.forEach(function(cur) {
        console.log((2016 - cur) >= 18);
    })
}


//isFullAge5(1990, 1999, 1965);
//isFullAge5(1990, 1999, 1965, 2016, 1987);
```

//ES6

```
function isFullAge6(...years) {
    years.forEach(cur => console.log( (2016 - cur) >= 18));
}

isFullAge6(1990, 1999, 1965, 2016, 1987);
```

//ES5

```
function isFullAge5(limit) {
    var argsArr = Array.prototype.slice.call(arguments, 1);

    argsArr.forEach(function(cur) {
        console.log((2016 - cur) >= limit);
    })
}


//isFullAge5(16, 1990, 1999, 1965);
isFullAge5(1990, 1999, 1965, 2016, 1987);
```

//ES6

```
function isFullAge6(limit, ...years) {
    years.forEach(cur => console.log( (2016 - cur) >= limit));
}

isFullAge6(16, 1990, 1999, 1965, 2016, 1987);
*/
```

// Lecture: Default parameters

```
/*
```

// ES5

```
function SmithPerson(firstName, yearOfBirth, lastName, nationality) {

    lastName === undefined ? lastName = 'Smith' : lastName = lastName;
    nationality === undefined ? nationality = 'american' : nationality = nationality;

    this.firstName = firstName;
    this.lastName = lastName;
    this.yearOfBirth = yearOfBirth;
    this.nationality = nationality;
}
```

//ES6

```
function SmithPerson(firstName, yearOfBirth, lastName = 'Smith', nationality = 'american') {
    this.firstName = firstName;
    this.lastName = lastName;
    this.yearOfBirth = yearOfBirth;
    this.nationality = nationality;
}


var john = new SmithPerson('John', 1990);
var emily = new SmithPerson('Emily', 1983, 'Diaz', 'spanish');
*/
```

// Lecture: Maps

```
/*
const question = new Map();
question.set('question', 'What is the official name of the latest major JavaScript version?');
question.set(1, 'ES5');
question.set(2, 'ES6');
question.set(3, 'ES2015');
question.set(4, 'ES7');
question.set('correct', 3);
question.set(true, 'Correct answer :D');
question.set(false, 'Wrong, please try again!');

console.log(question.get('question'));
//console.log(question.size);


if(question.has(4)) {
    //question.delete(4);
    //console.log('Answer 4 is here')
}

//question.clear();


//question.forEach((value, key) => console.log(`This is ${key}, and it's set to ${value}`));


for (let [key, value] of question.entries()) {
    if (typeof(key) === 'number') {
```

```
        console.log(`Answer ${key}: ${value}`);
    }
}

const ans = parseInt(prompt('Write the correct answer'));
console.log(question.get(ans === question.get('correct')));
*/
```

// Lecture: Classes

```
/*
//ES5

var Person5 = function(name, yearOfBirth, job) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;
    this.job = job;
}

Person5.prototype.calculateAge = function() {
    var age = new Date().getFullYear - this.yearOfBirth;
    console.log(age);
}

var john5 = new Person5('John', 1990, 'teacher');

//ES6

class Person6 {
    constructor (name, yearOfBirth, job) {
        this.name = name;
        this.yearOfBirth = yearOfBirth;
        this.job = job;
    }

    calculateAge() {
        var age = new Date().getFullYear - this.yearOfBirth;
        console.log(age);
    }

    static greeting() {
        console.log('Hey there!');
```

```
    }
}

const john6 = new Person6('John', 1990, 'teacher');

Person6.greeting();
*/
```

// Lecture: Classes and subclasses

```
/*
//ES5

var Person5 = function(name, yearOfBirth, job) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;
    this.job = job;
}

Person5.prototype.calculateAge = function() {
    var age = new Date().getFullYear() - this.yearOfBirth;
    console.log(age);
}

var Athlete5 = function(name, yearOfBirth, job, olymicGames, medals) {
    Person5.call(this, name, yearOfBirth, job);
    this.olymicGames = olymicGames;
    this.medals = medals;
}

Athlete5.prototype = Object.create(Person5.prototype);


Athlete5.prototype.wonMedal = function() {
    this.medals++;
    console.log(this.medals);
}


var johnAthlete5 = new Athlete5('John', 1990, 'swimmer', 3, 10);

johnAthlete5.calculateAge();
johnAthlete5.wonMedal();
```

//ES6

```
class Person6 {
    constructor (name, yearOfBirth, job) {
        this.name = name;
        this.yearOfBirth = yearOfBirth;
        this.job = job;
    }

    calculateAge() {
        var age = new Date().getFullYear() - this.yearOfBirth;
        console.log(age);
    }
}

class Athlete6 extends Person6 {
    constructor(name, yearOfBirth, job, olympicGames, medals) {
        super(name, yearOfBirth, job);
        this.olympicGames = olympicGames;
        this.medals = medals;
    }

    wonMedal() {
        this.medals++;
        console.log(this.medals);
    }
}

const johnAthlete6 = new Athlete6('John', 1990, 'swimmer', 3, 10);

johnAthlete6.wonMedal();
johnAthlete6.calculateAge();
*/
```

// CODING CHALLENGE

Suppose that you're working in a small town administration, and you're in charge of two town elements:
1. Parks
2. Streets

It's a very small town, so right now there are only 3 parks and 4 streets. All parks and streets have a name and a build year.

At an end-of-year meeting, your boss wants a final report with the following:
1. Tree density of each park in the town (formula: number of trees/park area)
2. Average age of each town's park (formula: sum of all ages/number of parks)
3. The name of the park that has more than 1000 trees
4. Total and average length of the town's streets
5. Size classification of all streets: tiny/small/normal/big/huge. If the size is unknown, the default is normal

All the report data should be printed to the console.

HINT: Use some of the ES6 features: classes, subclasses, template strings, default parameters, maps, arrow functions, destructuring, etc.

*/

```
class Element {
    constructor(name, buildYear) {
        this.name = name;
        this.buildYear = buildYear;
    }
}


class Park extends Element {
    constructor(name, buildYear, area, numTrees) {
        super(name, buildYear);
        this.area = area; //km2
        this.numTrees = numTrees;
    }

    treeDensity() {
        const density = this.numTrees / this.area;
        console.log(`${this.name} has a tree density of ${density} trees per square km.`);
    }
}


class Street extends Element {
    constructor(name, buildYear, length, size = 3) {
```

```
        super(name, buildYear);
        this.length = length;
        this.size = size;
    }

    classifyStreet () {
        const classification = new Map();
        classification.set(1, 'tiny');
        classification.set(2, 'small');
        classification.set(3, 'normal');
        classification.set(4, 'big');
        classification.set(5, 'huge');
        console.log(`${this.name}, build in ${this.buildYear}, is a ${classification.get(this.size)}
street.`);
    }
}


const allParks = [new Park('Green Park', 1987, 0.2, 215),
                  new Park('National Park', 1894, 2.9, 3541),
                  new Park('Oak Park', 1953, 0.4, 949)];

const allStreets = [new Street('Ocean Avenue', 1999, 1.1, 4),
                    new Street('Evergreen Street', 2008, 2.7, 2),
                    new Street('4th Street', 2015, 0.8),
                    new Street('Sunset Boulevard', 1982, 2.5, 5)];


function calc(arr) {

    const sum = arr.reduce((prev, cur, index) => prev + cur, 0);

    return [sum, sum / arr.length];

}


function reportParks(p) {

    console.log('-----PARKS REPORT-----');

    // Density
    p.forEach(el => el.treeDensity());
```

```
    // Average age
    const ages = p.map(el => new Date().getFullYear() - el.buildYear);
    const [totalAge, avgAge] = calc(ages);
    console.log(`Our ${p.length} parks have an average of ${avgAge} years.`);

    // Which park has more than 1000 trees
    const i = p.map(el => el.numTrees).findIndex(el => el >= 1000);
    console.log(`${p[i].name} has more than 1000 trees.`);

}


function reportStreets(s) {

    console.log('-----STREETS REPORT-----');

    //Total and average length of the town's streets
    const [totalLength, avgLength] = calc(s.map(el => el.length));
    console.log(`Our ${s.length} streets have a total length of ${totalLength} km, with an average
of ${avgLength} km.`);

    // CLassify sizes
    s.forEach(el => el.classifyStreet());
}

reportParks(allParks);
reportStreets(allStreets);
```

// Lecture: let and const

```
/*
// ES5

var name5 = 'Jane Smith';
var age5 = 23;
name5 = 'Jane Miller';
console.log(name5);

// ES6
const name6 = 'Jane Smith';
let age6 = 23;
name6 = 'Jane Miller';
```

```
console.log(name6);


// ES5

function driversLicence5(passedTest) {

    if (passedTest) {
        console.log(firstName);
        var firstName = 'John';
        var yearOfBirth = 1990;
    }


    console.log(firstName + ', born in ' + yearOfBirth + ', is now officially allowed to drive a car.');
}

driversLicence5(true);


// ES6

function driversLicence6(passedTest) {

    //console.log(firstName);
    let firstName;
    const yearOfBirth = 1990;

    if (passedTest) {
        firstName = 'John';
    }

    console.log(firstName + ', born in ' + yearOfBirth + ', is now officially allowed to drive a car.');
}

driversLicence6(true);



var i = 23;

for (var i = 0; i < 5; i++) {
    console.log(i);
```

```
}

console.log(i);
*/
```

// Lecture: Blocks and IIFEs

```
/*
// ES6

{
    const a = 1;
    let b = 2;
    var c = 3;
}

//console.log(a + b);
console.log(c);


// ES5
(function() {
    var c = 3;
})();

//console.log(c);
*/
```

// Lecture: Strings

```
/*
let firstName = 'John';
let lastName = 'Smith';
const yearOfBirth = 1990;

function calcAge(year) {
    return 2016 - year;
}
```

// ES5

```
console.log('This is ' + firstName + ' ' + lastName + '. He was born in ' + yearOfBirth + '. Today, he is ' + calcAge(yearOfBirth) + ' years old.');

// ES6
console.log(`This is ${firstName} ${lastName}. He was born in ${yearOfBirth}. Today, he is ${calcAge(yearOfBirth)} years old.`);




const n = `${firstName} ${lastName}`;
console.log(n.startsWith('j'));
console.log(n.endsWith('Sm'));
console.log(n.includes('oh'));
console.log(`${firstName} `.repeat(5));
*/
```

// Lecture: Arrow functions

```
/*
const years = [1990, 1965, 1982, 1937];
```

// ES5

```
var ages5 = years.map(function(el) {
   return 2016 - el;
});
console.log(ages5);


// ES6
let ages6 = years.map(el => 2016 - el);
console.log(ages6);

ages6 = years.map((el, index) => `Age element ${index + 1}: ${2016 - el}.`);
console.log(ages6);

ages6 = years.map((el, index) => {
   const now = new Date().getFullYear();
   const age = now - el;
   return `Age element ${index + 1}: ${age}.`
```

```
});
console.log(ages6);
*/
```

// Lecture: Arrow functions 2

```
/*
```

// ES5

```
var box5 = {
    color: 'green',
    position: 1,
    clickMe: function() {

        var self = this; document.querySelector('.green').addEventListener('click', function() {
            var str = 'This is box number ' + self.position + ' and it is ' + self.color;
            alert(str);
        });
    }
}
//box5.clickMe();
```

// ES6

```
const box6 = {
    color: 'green',
    position: 1,
    clickMe: function() {
        document.querySelector('.green').addEventListener('click', () => {
            var str = 'This is box number ' + this.position + ' and it is ' + this.color;
            alert(str);
        });
    }
}
box6.clickMe();


const box66 = {
    color: 'green',
    position: 1,
    clickMe: () => {
```

```
        document.querySelector('.green').addEventListener('click', () => {
            var str = 'This is box number ' + this.position + ' and it is ' + this.color;
            alert(str);
        });
    }
}
box66.clickMe();


function Person(name) {
    this.name = name;
}

// ES5

Person.prototype.myFriends5 = function(friends) {

    var arr = friends.map(function(el) {
      return this.name + ' is friends with ' + el;
    }.bind(this));

    console.log(arr);
}

var friends = ['Bob', 'Jane', 'Mark'];
new Person('John').myFriends5(friends);


// ES6

Person.prototype.myFriends6 = function(friends) {

    var arr = friends.map(el => `${this.name} is friends with ${el}`);

    console.log(arr);
}

new Person('Mike').myFriends6(friends);
*/
```

// Lecture: Destructuring

```
/*
// ES5

var john = ['John', 26];
//var name = john[0];
//var age = john[1];


// ES6

const [name, age] = ['John', 26];
console.log(name);
console.log(age);

const obj = {
    firstName: 'John',
    lastName: 'Smith'
};

const {firstName, lastName} = obj;
console.log(firstName);
console.log(lastName);

const {firstName: a, lastName: b} = obj;
console.log(a);
console.log(b);




function calcAgeRetirement(year) {
    const age = new Date().getFullYear() - year;
    return [age, 65 - age];
}


const [age2, retirement] = calcAgeRetirement(1990);
console.log(age2);
console.log(retirement);
*/
```

// Lecture: Arrays

```
/*
const boxes = document.querySelectorAll('.box');

//ES5

var boxesArr5 = Array.prototype.slice.call(boxes);
boxesArr5.forEach(function(cur) {
    cur.style.backgroundColor = 'dodgerblue';
});

//ES6

const boxesArr6 = Array.from(boxes);
Array.from(boxes).forEach(cur => cur.style.backgroundColor = 'dodgerblue');


//ES5

for(var i = 0; i < boxesArr5.length; i++) {

    if(boxesArr5[i].className === 'box blue') {
        continue;
    }

    boxesArr5[i].textContent = 'I changed to blue!';

}


//ES6

for (const cur of boxesArr6) {
    if (cur.className.includes('blue')) {
        continue;
    }
    cur.textContent = 'I changed to blue!';
}
```

```
//ES5

var ages = [12, 17, 8, 21, 14, 11];

var full = ages.map(function(cur) {
   return cur >= 18;
});
console.log(full);

console.log(full.indexOf(true));
console.log(ages[full.indexOf(true)]);


//ES6

console.log(ages.findIndex(cur => cur >= 18));
console.log(ages.find(cur => cur >= 18));
*/
```

// Lecture: Spread operator

```
/*
function addFourAges (a, b, c, d) {
   return a + b + c + d;
}

var sum1 = addFourAges(18, 30, 12, 21);
console.log(sum1);

//ES5

var ages = [18, 30, 12, 21];
var sum2 = addFourAges.apply(null, ages);
console.log(sum2);

//ES6

const sum3 = addFourAges(...ages);
console.log(sum3);
```

```
const familySmith = ['John', 'Jane', 'Mark'];
const familyMiller = ['Mary', 'Bob', 'Ann'];
const bigFamily = [...familySmith, 'Lily', ...familyMiller];
console.log(bigFamily);


const h = document.querySelector('h1');
const boxes = document.querySelectorAll('.box');
const all = [h, ...boxes];

Array.from(all).forEach(cur => cur.style.color = 'purple');
*/
```

// Lecture: Rest parameters

```
/*
//ES5

function isFullAge5() {
    //console.log(arguments);
    var argsArr = Array.prototype.slice.call(arguments);

    argsArr.forEach(function(cur) {
        console.log((2016 - cur) >= 18);
    })
}


//isFullAge5(1990, 1999, 1965);
//isFullAge5(1990, 1999, 1965, 2016, 1987);


//ES6

function isFullAge6(...years) {
    years.forEach(cur => console.log( (2016 - cur) >= 18));
}

isFullAge6(1990, 1999, 1965, 2016, 1987);
```

//ES5

```
function isFullAge5(limit) {
    var argsArr = Array.prototype.slice.call(arguments, 1);

    argsArr.forEach(function(cur) {
        console.log((2016 - cur) >= limit);
    })
}


//isFullAge5(16, 1990, 1999, 1965);
isFullAge5(1990, 1999, 1965, 2016, 1987);
```

//ES6

```
function isFullAge6(limit, ...years) {
    years.forEach(cur => console.log( (2016 - cur) >= limit));
}

isFullAge6(16, 1990, 1999, 1965, 2016, 1987);

*/
```

// Lecture: Default parameters

```
/*
```

// ES5

```
function SmithPerson(firstName, yearOfBirth, lastName, nationality) {

    lastName === undefined ? lastName = 'Smith' : lastName = lastName;
    nationality === undefined ? nationality = 'american' : nationality = nationality;

    this.firstName = firstName;
    this.lastName = lastName;
    this.yearOfBirth = yearOfBirth;
    this.nationality = nationality;
}
```

//ES6

```
function SmithPerson(firstName, yearOfBirth, lastName = 'Smith', nationality = 'american') {
    this.firstName = firstName;
    this.lastName = lastName;
    this.yearOfBirth = yearOfBirth;
    this.nationality = nationality;
}


var john = new SmithPerson('John', 1990);
var emily = new SmithPerson('Emily', 1983, 'Diaz', 'spanish');
*/
```

// Lecture: Maps

```
/*
const question = new Map();
question.set('question', 'What is the official name of the latest major JavaScript version?');
question.set(1, 'ES5');
question.set(2, 'ES6');
question.set(3, 'ES2015');
question.set(4, 'ES7');
question.set('correct', 3);
question.set(true, 'Correct answer :D');
question.set(false, 'Wrong, please try again!');

console.log(question.get('question'));
//console.log(question.size);


if(question.has(4)) {
    //question.delete(4);
    //console.log('Answer 4 is here')
}

//question.clear();


//question.forEach((value, key) => console.log(`This is ${key}, and it's set to ${value}`));
```

```
for (let [key, value] of question.entries()) {
    if (typeof(key) === 'number') {
        console.log(`Answer ${key}: ${value}`);
    }
}

const ans = parseInt(prompt('Write the correct answer'));
console.log(question.get(ans === question.get('correct')));
*/
```

// Lecture: Classes

```
/*
```

//ES5

```
var Person5 = function(name, yearOfBirth, job) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;
    this.job = job;
}

Person5.prototype.calculateAge = function() {
    var age = new Date().getFullYear - this.yearOfBirth;
    console.log(age);
}

var john5 = new Person5('John', 1990, 'teacher');
```

//ES6

```
class Person6 {
    constructor (name, yearOfBirth, job) {
        this.name = name;
        this.yearOfBirth = yearOfBirth;
        this.job = job;
    }

    calculateAge() {
        var age = new Date().getFullYear - this.yearOfBirth;
        console.log(age);
    }

    static greeting() {
```

```
        console.log('Hey there!');
    }
}

const john6 = new Person6('John', 1990, 'teacher');

Person6.greeting();
*/
```

// Lecture: Classes and subclasses

```
/*
//ES5

var Person5 = function(name, yearOfBirth, job) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;
    this.job = job;
}

Person5.prototype.calculateAge = function() {
    var age = new Date().getFullYear() - this.yearOfBirth;
    console.log(age);
}

var Athlete5 = function(name, yearOfBirth, job, olymicGames, medals) {
    Person5.call(this, name, yearOfBirth, job);
    this.olymicGames = olymicGames;
    this.medals = medals;
}

Athlete5.prototype = Object.create(Person5.prototype);


Athlete5.prototype.wonMedal = function() {
    this.medals++;
    console.log(this.medals);
}


var johnAthlete5 = new Athlete5('John', 1990, 'swimmer', 3, 10);

johnAthlete5.calculateAge();
```

```
johnAthlete5.wonMedal();
```

//ES6

```
class Person6 {
    constructor (name, yearOfBirth, job) {
        this.name = name;
        this.yearOfBirth = yearOfBirth;
        this.job = job;
    }

    calculateAge() {
        var age = new Date().getFullYear() - this.yearOfBirth;
        console.log(age);
    }
}

class Athlete6 extends Person6 {
    constructor(name, yearOfBirth, job, olympicGames, medals) {
        super(name, yearOfBirth, job);
        this.olympicGames = olympicGames;
        this.medals = medals;
    }

    wonMedal() {
        this.medals++;
        console.log(this.medals);
    }
}

const johnAthlete6 = new Athlete6('John', 1990, 'swimmer', 3, 10);

johnAthlete6.wonMedal();
johnAthlete6.calculateAge();
*/
```

// CODING CHALLENGE

```
/*
```

Suppose that you're working in a small town administration, and you're in charge of two town elements:
1. Parks
2. Streets

It's a very small town, so right now there are only 3 parks and 4 streets. All parks and streets have a name and a build year.

At an end-of-year meeting, your boss wants a final report with the following:
1. Tree density of each park in the town (formula: number of trees/park area)
2. Average age of each town's park (formula: sum of all ages/number of parks)
3. The name of the park that has more than 1000 trees
4. Total and average length of the town's streets
5. Size classification of all streets: tiny/small/normal/big/huge. If the size is unknown, the default is normal

All the report data should be printed to the console.

HINT: Use some of the ES6 features: classes, subclasses, template strings, default parameters, maps, arrow functions, destructuring, etc.

*/

```
var Element = function Element(name, buildYear) {
  _classCallCheck(this, Element);

  this.name = name;
  this.buildYear = buildYear;
};

var Park = function (_Element) {
  _inherits(Park, _Element);

  function Park(name, buildYear, area, numTrees) {
    _classCallCheck(this, Park);

    var _this = _possibleConstructorReturn(this, (Park.__proto__ ||
Object.getPrototypeOf(Park)).call(this, name, buildYear));

    _this.area = area; //km2
    _this.numTrees = numTrees;
    return _this;
```

```
    }

    _createClass(Park, [{
        key: 'treeDensity',
        value: function treeDensity() {
            var density = this.numTrees / this.area;
            console.log(this.name + ' has a tree density of ' + density + ' trees per square km.');
        }
    }]);

    return Park;
}(Element);

var Street = function (_Element2) {
    _inherits(Street, _Element2);

    function Street(name, buildYear, length) {
        var size = arguments.length > 3 && arguments[3] !== undefined ? arguments[3] : 3;

        _classCallCheck(this, Street);

        var _this2 = _possibleConstructorReturn(this, (Street.__proto__ ||
Object.getPrototypeOf(Street)).call(this, name, buildYear));

        _this2.length = length;
        _this2.size = size;
        return _this2;
    }

    _createClass(Street, [{
        key: 'classifyStreet',
        value: function classifyStreet() {
            var classification = new Map();
            classification.set(1, 'tiny');
            classification.set(2, 'small');
            classification.set(3, 'normal');
            classification.set(4, 'big');
            classification.set(5, 'huge');
            console.log(this.name + ', build in ' + this.buildYear + ', is a ' + classification.get(this.size)
+ ' street.');
        }
    }]);

    return Street;
```

```
}(Element);

var allParks = [new Park('Green Park', 1987, 0.2, 215), new Park('National Park', 1894, 2.9,
3541), new Park('Oak Park', 1953, 0.4, 949)];

var allStreets = [new Street('Ocean Avenue', 1999, 1.1, 4), new Street('Evergreen Street', 2008,
2.7, 2), new Street('4th Street', 2015, 0.8), new Street('Sunset Boulevard', 1982, 2.5, 5)];

function calc(arr) {

    var sum = arr.reduce(function (prev, cur, index) {
        return prev + cur;
    }, 0);

    return [sum, sum / arr.length];
}

function reportParks(p) {

    console.log('-----PARKS REPORT-----');

    // Density
    p.forEach(function (el) {
        return el.treeDensity();
    });

    // Average age
    var ages = p.map(function (el) {
        return new Date().getFullYear() - el.buildYear;
    });

    var _calc = calc(ages),
        _calc2 = _slicedToArray(_calc, 2),
        totalAge = _calc2[0],
        avgAge = _calc2[1];

    console.log('Our ' + p.length + ' parks have an average of ' + avgAge + ' years.');

    // Which park has more than 1000 trees
    var i = p.map(function (el) {
        return el.numTrees;
    }).findIndex(function (el) {
        return el >= 1000;
    });
```

```
    console.log(p[i].name + ' has more than 1000 trees.');
}

function reportStreets(s) {

    console.log('-----STREETS REPORT-----');

    //Total and average length of the town's streets

    var _calc3 = calc(s.map(function (el) {
        return el.length;
    })),
        _calc4 = _slicedToArray(_calc3, 2),
        totalLength = _calc4[0],
        avgLength = _calc4[1];

    console.log('Our ' + s.length + ' streets have a total length of ' + totalLength + ' km, with an
average of ' + avgLength + ' km.');

    // CLassify sizes
    s.forEach(function (el) {
        return el.classifyStreet();
    });
}

reportParks(allParks);
reportStreets(allStreets);
```