НАСЛЕДОВАНИЕ ОСНОВАННОЕ НА ПРОТОТИПАХ

```
let someStudent = {
  name: "John",
  age: 20,
  courses: [/* ... */],
  passExam(course) {
    this.courses.find(el => el.name === course).passed = true;
    return true;
  }
};

let anotherStudent = {
  name: "Nick",
  age: 19,
  courses: [/* ... */],
  passExam(course) {/* ... */}
};
```

СВОЙСТВА PROTOTYPE И CONSTRUCTOR

```
function Student(name, age, courses = []) {
  this.name = name;
  this.age = age;
  this.courses = courses.slice();
}

Student.prototype.passExam = function(course) {
  this.courses.find(el => el.name === course).passed = true;
  return true;
}

let jsStudent = new Student("Bill", 20, [{name: "JS", passed: false}]
let phpStudent = new Student("Kent", 21, [{name: "PHP", passed: false

jsStudent.passExam("JS")         // true
```

```
function Person(
  name = "unknown",
  lastName = "unknown",
  age = 18) {
  this.name = name;
  this.lastName = lastName;
  this.age = age;
}

Person.prototype.introduce = function(greet = "Hello") {
  return `${greet}! My name is ${this.name} ${this.lastName}.\
I'm ${this.age} years old`;
};
```

```
function Student(
  name, lastName, age,
  courses = [],
  skills = []) {
  Person.apply(this, arguments);

  this.courses = courses.map(function(el) {
    return new Course(
      el.name, el.skills, el.hours, el.teacher);
  });
  this.skills = skills.slice();
}
```

```
Student.prototype = Object.create(Person.prototype);

Student.prototype.constructor = Student;

Student.prototype.addCourse = function(...courseData) {
  let course = new Course(...courseData);
  this.courses.push(course);
  return course;
}

Student.prototype.addMark = function(courseName, mark) {
  this.courses.find(el => el.name === courseName).addMark(mark);
}
```

```javascript
function Course(
courseName = "Unknown",
skills = [],
hours = 0,
teacher = "Unknown Unknown") {
  this.name = courseName;
  this.skills = skills;
  this.hours = hours;
  this.teacher = teacher;
  this.marks = [];
}

Course.prototype.addMark = function(mark) {
  this.marks.push(mark);
};
```

```
let jsCourse = ["JS", ["js", "web APIs"], 96, "Y.S."];
let phpCourse = ["PHP", ["php", "mySQL"], 720, "V.N"];

let studentData = [
  "Jack",
  "Sparrow",
  38,
  [],
  ["pirate", "diplomacy", "navigation"],
];
```

```
let jackSparrow = new Student(...studentData);

jackSparrow.introduce("Yohoho");

jackSparrow.addCourse(...jsCourse);
jackSparrow.addCourse(...phpCourse);

jackSparrow.addMark("JS", 5);
jackSparrow.addMark("JS", 4)
jackSparrow.addMark("PHP", 5);

console.log(jackSparrow.courses);
```

```
let Course = {
  constructor(courseName, skills, hours, teacher) {
    this.name = courseName;
    this.skills = skills;
    this.hours = hours;
    this.teacher = teacher;
    this.marks = [];

    return this;
  },
  addMark(mark) {
    let result = this.marks.push(mark);
    return result;
  }
};
```

```
let Person = {
  constructor(name, lastName, age) {
    this.name = name;
    this.lastName = lastName;
    this.age = age;

    return this;
  },
  introduce(greet) {
    return `${greet}! My name is ${this.name} ${this.lastName}.`;
  }
};
```

```javascript
let Student = Object.create(Person);

Student.constructor = function(name, lastName, age, courses, skills)

  Person.constructor.apply(this, arguments);

  this.courses = courses.map(function(el) {
    return Object
      .create(Course)
      .constructor(
        el.name, el.skills, el.hours, el.teacher);
  });
  this.skills = skills;

  return this;
};
```

```javascript
Student.addCourse = function(...courseData) {
   let course = Object.create(Course).constructor(...courseData);
   this.courses.push(course);
   return course;
};

Student.addMark = function(courseName, mark) {
  this.courses.find(el => el.name === courseName).addMark(mark);
}
```

```
let jackSparrow = Object.create(Student).constructor(...studentData);

jackSparrow.introduce("Yohoho");

jackSparrow.addCourse(...jsCourse);
jackSparrow.addCourse(...phpCourse);

jackSparrow.addMark("JS", 5);
jackSparrow.addMark("JS", 4)
jackSparrow.addMark("PHP", 5);

console.log(jackSparrow.courses);
```

КЛЮЧЕВОЕ СЛОВО

CLASS

```
class Course {
  constructor(courseName, skills, hours, teacher) {
    this.name = courseName;
    this.skills = skills;
    this.hours = hours;
    this.teacher = teacher;
    this.marks = [];
  }

  addMark(mark) {
    let result = this.marks.push(mark);
    return result;
  }
};
```

```
class Person {
  constructor(name, lastName, age) {
    this.name = name;
    this.lastName = lastName;
    this.age = age;
  }

  introduce(greeting) {
    return `${greeting}! My name is ${this.name} ${this.lastName}.`;
  }
};
```

```
class Student extends Person {
  constructor(name, lastName, age, courses, skills) {
    super(name, lastName, age);

    this.courses = courses.map(el => new Course(
      el.name, el.skills, el.hours, el.teacher));
    this.skills = skills;
  }

  addCourse(...courseData) {
     this.courses.push(new Course(...courseData));
  }
  addMark(courseName, mark) {
    this.courses.find(el => el.name === courseName).addMark(mark);
  }
};
```

```
let jackSparrow = new Student(...studentData);

jackSparrow.introduce("Yohoho");

jackSparrow.addCourse(...jsCourse);
jackSparrow.addCourse(...phpCourse);

jackSparrow.addMark("JS", 5);
jackSparrow.addMark("JS", 4)
jackSparrow.addMark("PHP", 5);

console.log(jackSparrow.courses);
```

```
{}.toString();                                          // ["object Object"]
Object.prototype.toString.call([]);
Object.prototype.toString.call(1);
Object.prototype.toString.call(function() {});
Object.prototype.toString.call("");
Object.prototype.toString.call(null);
Object.prototype.toString.call(undefined);

let getType = val => Object.prototype.toString.call(val).slice(8, -1)
```

```
// не повтояйте это дома

Array.prototype.map = function() {
  return ".map() was changed";
}

let arr = [1, 2, 3];
let doubleValues = arr.map((el) => el * 2);

console.log(doubleValues);
```

ОБЪЕКТ DATE

```
// синтаксис
new Date();
new Date(value);
new Date(dateString);
new Date(year, month[, day[, hour[, minute[, second[, millisecond]]]]

new Date(1463295600000);
new Date('May 15, 2016 10:00:00');
new Date(2016, 4, 15, 10);
Date()   // "Sun May 15 2016 10:00:00 GMT+0300 (Финляндия (лето))"
```

```
Date.now()
Date.parse(new Date('May 15, 2016 10:00:00'));     // 1463295600000
Date.UTC(2016, 4, 15, 10)                          // 1463306400000
```

```
let dateNow = new Date();

// Date.prototype.getDate();
dateNow.getDate();            // 15

// Date.prototype.getDay();   // zero-based
dateNow.getDay();             // 0

Date.prototype.getFullYear();
Date.prototype.getHours();
Date.prototype.getMilliseconds();
Date.prototype.getMinutes();
Date.prototype.getMonth();              // zero-based
Date.prototype.getSeconds();
Date.prototype.getTime();               // UTC
Date.prototype.getUTCDate();
Date.prototype.getUTCDay();
Date.prototype.getUTCFullYear();
Date.prototype.getUTCHours();
Date.prototype.getUTCMilliseconds();
Date.prototype.getUTCMinutes();
Date.prototype.getUTCMonth();
Date.prototype.getUTCSeconds();
Date.prototype.setDate();
Date.prototype.setDay();
Date.prototype.setFullYear()

/* ... */

Date.prototype.setUTCSeconds();
Date.prototype.toDateString();
Date.prototype.toISOString();
Date.prototype.toJSON();
Date.prototype.toLocaleString();
Date.prototype.toLocaleDateString();
Date.prototype.toLocaleTimeString();
Date.prototype.toUTCString();
```