S1 Z5

Classes and Modules

01
Basic

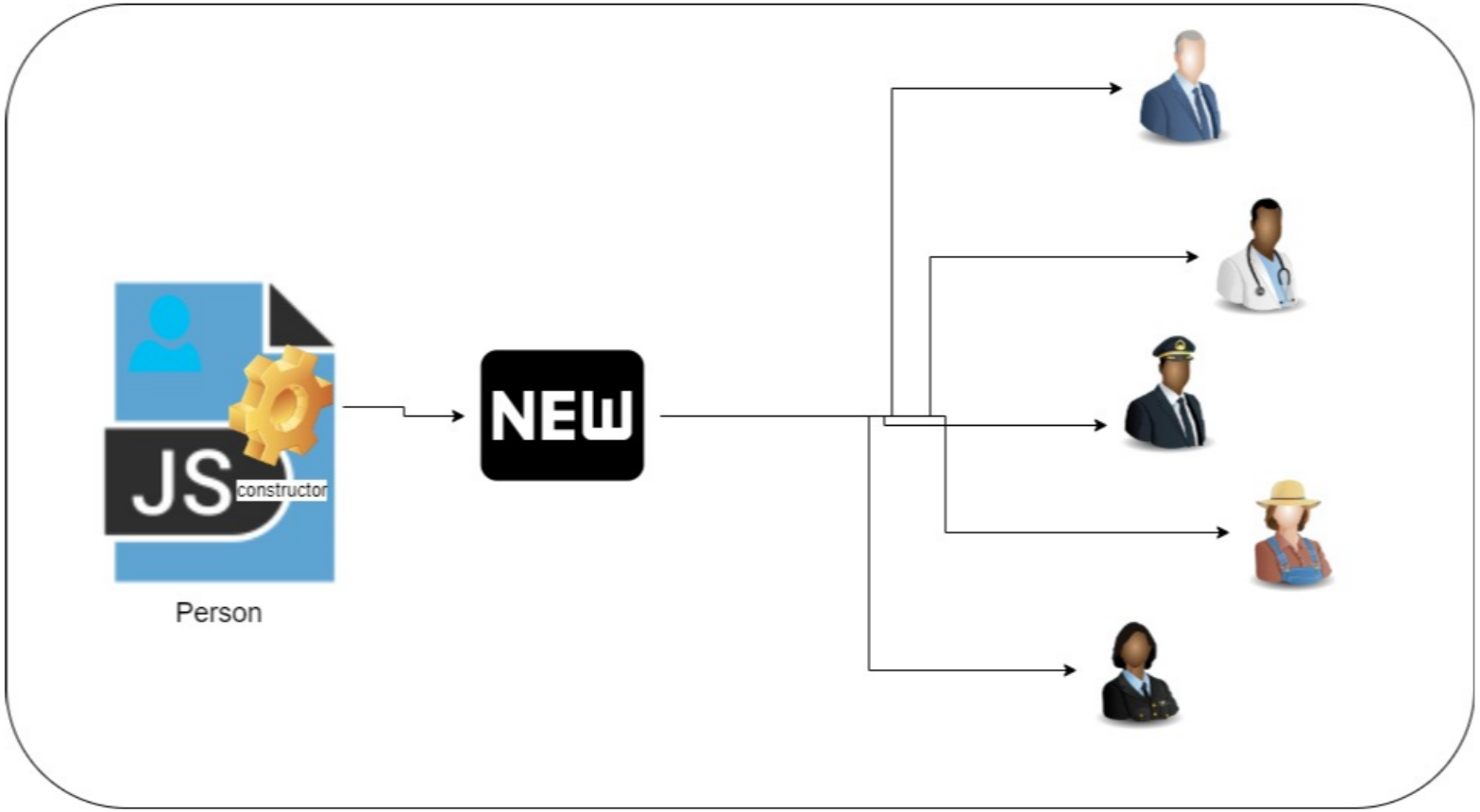
02
Constructor

03
Methods

04
Inheritance

05
Modules

Classes and Modules


JS
constructor
Person
NEW

Basic

```
class Person {

}

let me = new Person();
console.log(me);
```

Constructor

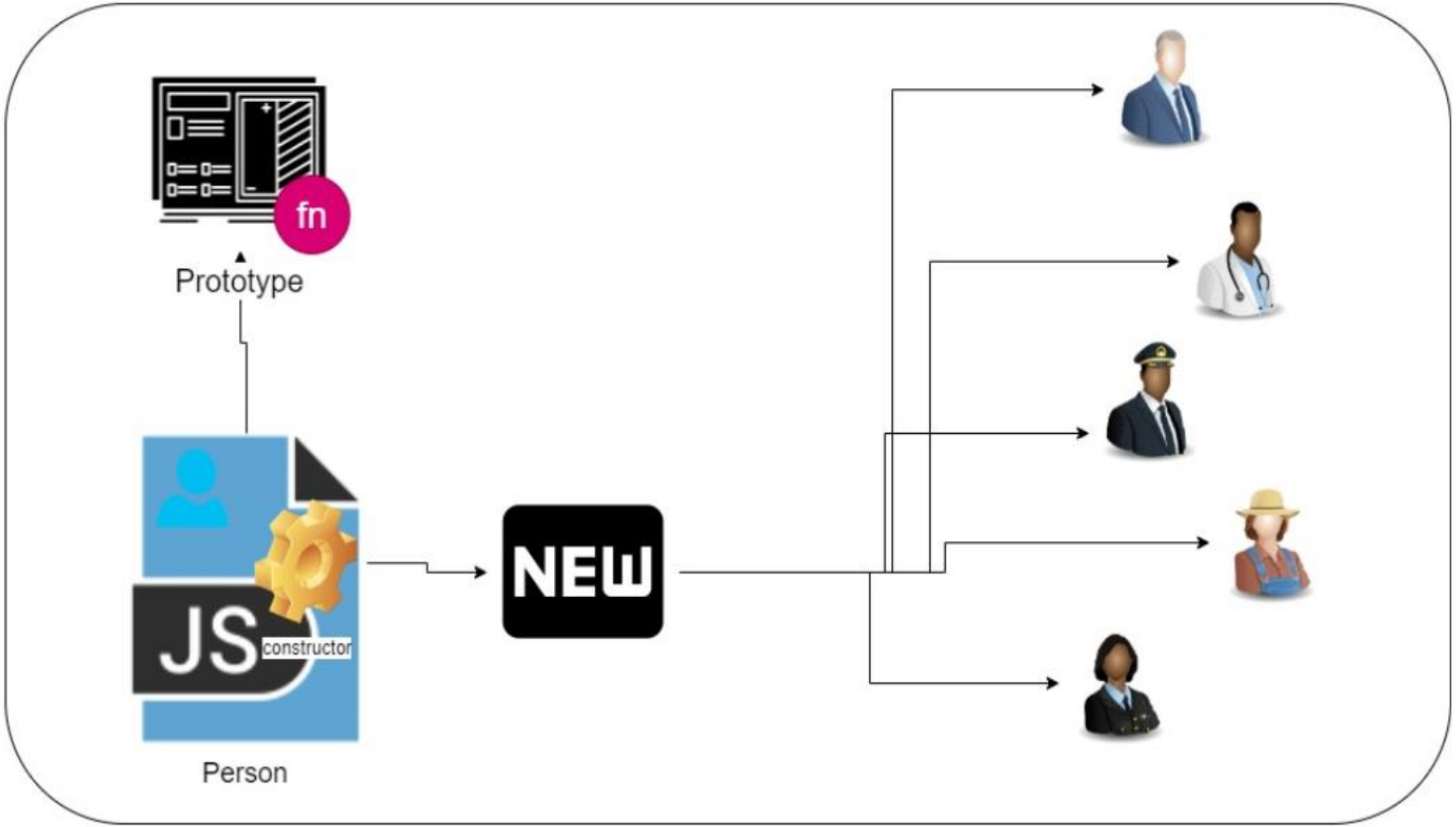
```
class Person {
   constructor(id){
      this.id = id;
   }
}

let me = new Person(1);
console.log(me);
```

Constructor

```
class Person {
   constructor(id){
      personId = id;
   }
}

let me = new Person(1);
console.log(me);
```

Constructor

```
class Person {
   constructor(id){
      let personId = id;
   }
}

let me = new Person(1);
console.log(me);
```

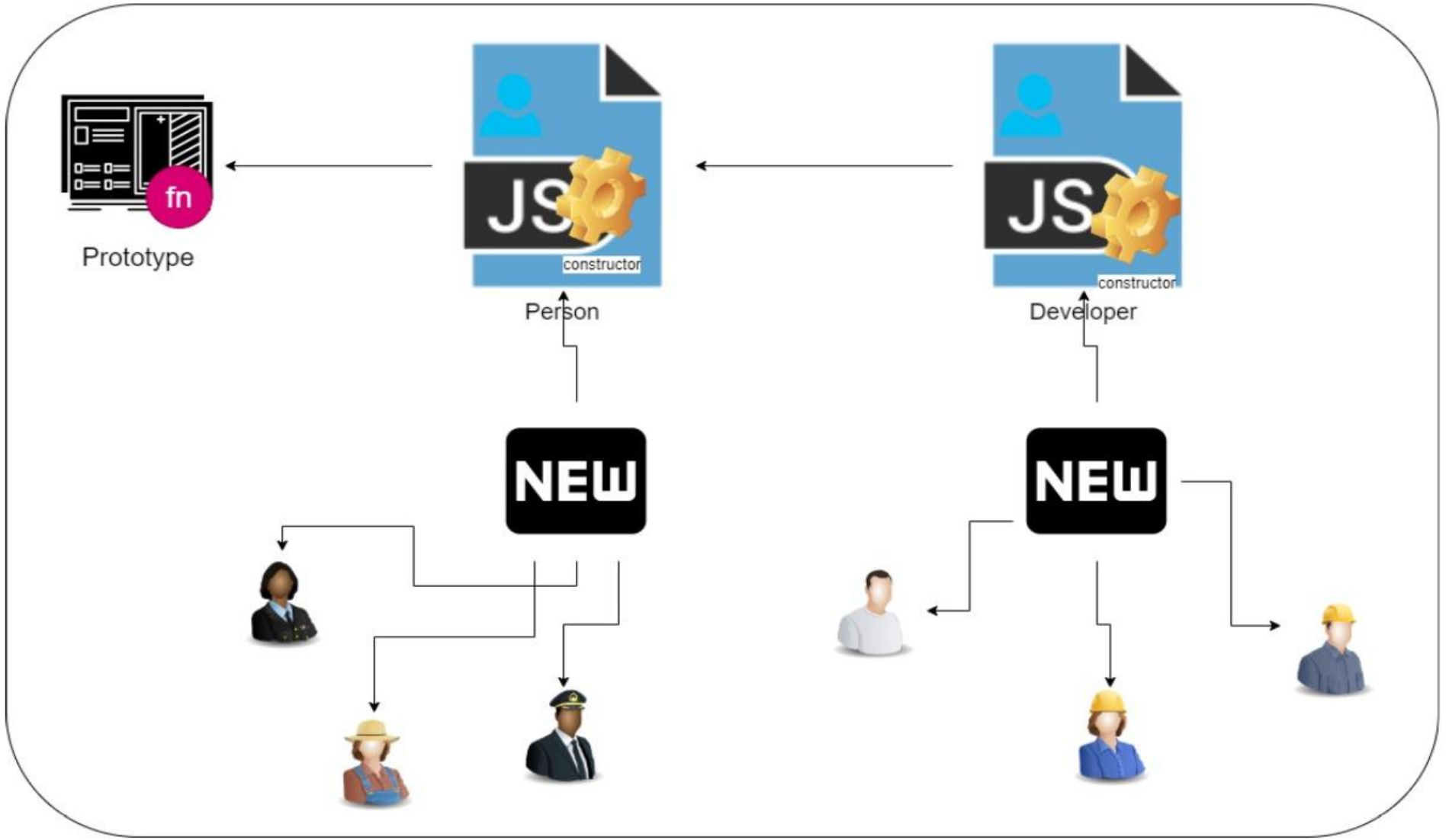

fn
Prototype
JS
constructor
Person
NEW

Methods

```
class Person {
    constructor(id){
        this.id = id;
    }
    showInfo(){
        return `Person Id: ${this.id}`
    }
}

let me = new Person(1);
console.log(me);
console.log(me.showInfo());
```

Methods

```
class Person {
    constructor(id, firstName, lastName){
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    showInfo(){
        return `${this.firstName + ' ' + this.lastName} Id: ${this.id}`
    }
}

let me = new Person(1,'Karol','Rogowski');
console.log(me);
console.log(me.showInfo());
```


fn
Prototype
JS
constructor
Person
JS
constructor
Developer
NEW
NEW

Inheritance

```
class Person {
    constructor(){
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type}`
    }
}

class JsDeveloper extends Person{
}

let jsDev = new JsDeveloper();
console.log(jsDev);
console.log(jsDev.showInfo());
```

Inheritance

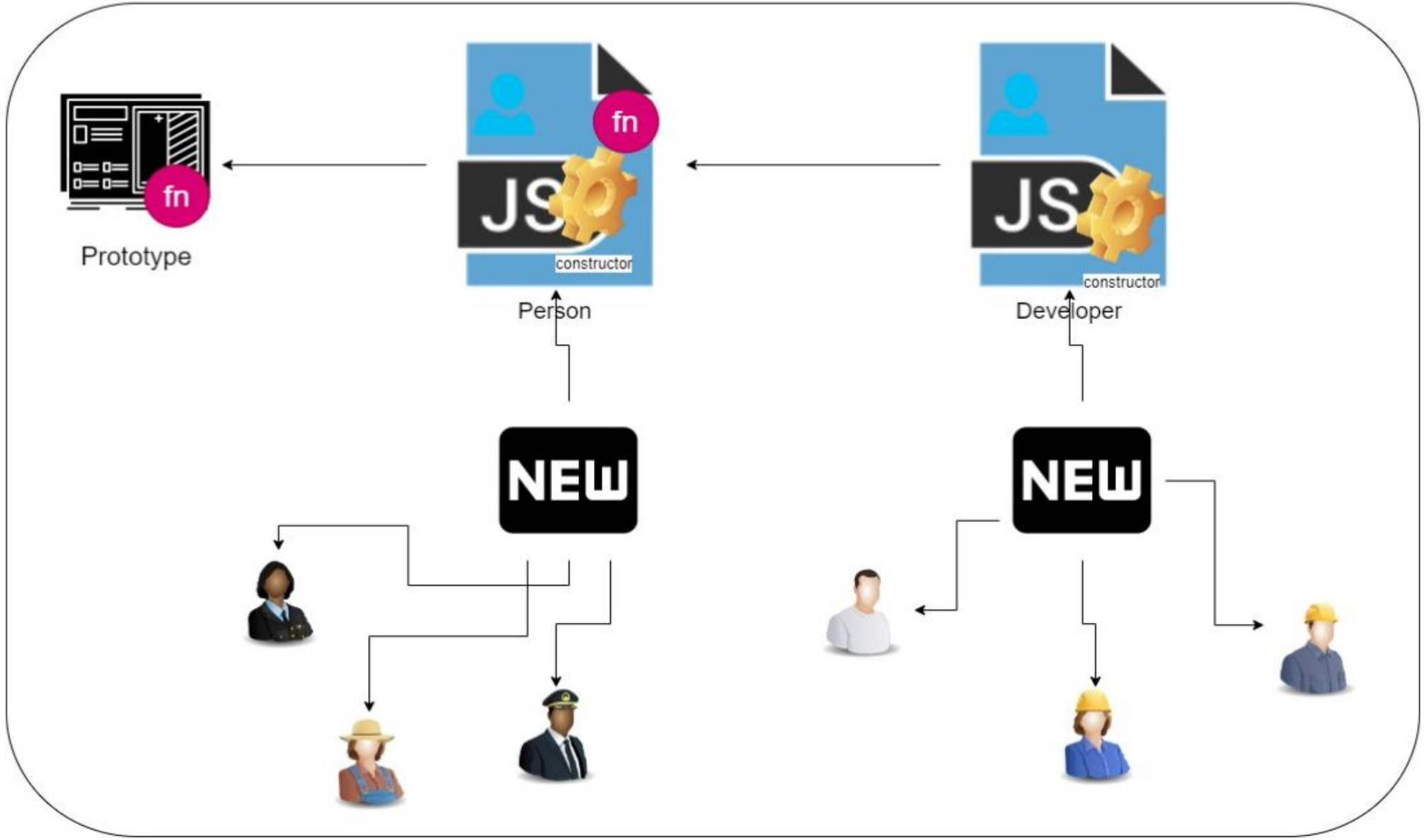
```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
    return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id){
        super(id);
    }
}

let jsDev = new JsDeveloper(5);
console.log(jsDev);
console.log(jsDev.showInfo());
```

Inheritance

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id){
        super(id);
        this.type = 'JS Developer'
    }
}

let jsDev = new JsDeveloper(5);
console.log(jsDev);
console.log(jsDev.showInfo());
```

Inheritance

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id,framework){
        super(id);
        this.type = 'JS Developer';
        this.framework = framework;
    }
}

let jsDev = new JsDeveloper(5,'React');
console.log(jsDev);
console.log(jsDev.showInfo());
```


fn
Prototype
fn
JS
constructor
Person
JS
constructor
Developer
NEW
NEW

Inheritance

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id,framework){
        super(id);
        this.type = 'JS Developer';
        this.framework = framework;
    }

    showDeveloperInfo(){
        return `Of type ${this.type}, id ${this.id} and favourite frameworks is ${this.framework}`
    }
}

let jsDev = new JsDeveloper(5,'React');
console.log(jsDev);
console.log(jsDev.showDeveloperInfo());
```

Inheritance

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id,framework){
        super(id);
        this.type = 'JS Developer';
        this.framework = framework;
    }

    showInfo(){
        return `Of type ${this.type}, id ${this.id} and favourite frameworks is ${this.framework}`
    }
}

let jsDev = new JsDeveloper(5,'React');
console.log(jsDev);
console.log(jsDev.showInfo());
```

Inheritance

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person';
    }
    showInfo(){
        return `Of type ${this.type} and id ${this.id}`
    }
}

class JsDeveloper extends Person{
    constructor(id,framework){
        super(id);
        this.type = 'JS Developer';
        this.framework = framework;
    }

    showInfo(){
        return super.showInfo() + ` and favourite frameworks is ${this.framework}`
    }
}

let jsDev = new JsDeveloper(5,'React');
console.log(jsDev);
console.log(jsDev.showInfo());
```


Classes
People
JsDeveloper.js
Person.js
node_modules
1)Basic.js
2)Constructor.js
3)Methods.js
4)Inheritance.js

Module 1/4

Module 2/4 (*Person.js*)

```
class Person {
    constructor(id){
        this.id = id;
        this.type = 'basic person’;
    }

    showInfo(){
    return `Of type ${this.type} and id ${this.id}`
    }
}

module.exports = Person;
```

Module 3/4 (*JsDeveloper.js*)

```
var Person = require("./Person");

class JsDeveloper extends Person{
    constructor(id,framework){
        super(id);
        this.type = 'JS Developer';
        this.framework = framework;
    }

    showInfo(){
        return super.showInfo() + ` and favourite frameworks is ${this.framework}`
    }
}

module.exports = JsDeveloper;
```

Module 4/4 (*Module.js*)

```
let JsDeveloper = require("./Classes/People/JsDeveloper");

let jsDev = new JsDeveloper(5,'React');
console.log(jsDev);
console.log(jsDev.showInfo());
```

Errors

1

Error

2

Try / Catch

3

Finally

4

Custom Error

Errors

Error

```
let person = Karol
console.log(person);
```

Try / Catch

```
try {
   let person = Karol
} catch (error) {
   console.log('error: ', error);
}

console.log('done');
```

Finally

```
try {
   let person = Karol
} catch (error) {
   console.log('error: ', error);
} finally {
   console.log('finally block reasech’)
}

console.log('done');
```

Custom Error

```
try {
   throw new Error('Custom application error')
} catch (error) {
   console.log('error: ', error);
} finally {
   console.log('finally block reasech’)
}

console.log('done');
```

Promises

01 Basic

02 Static Result

03 Generator

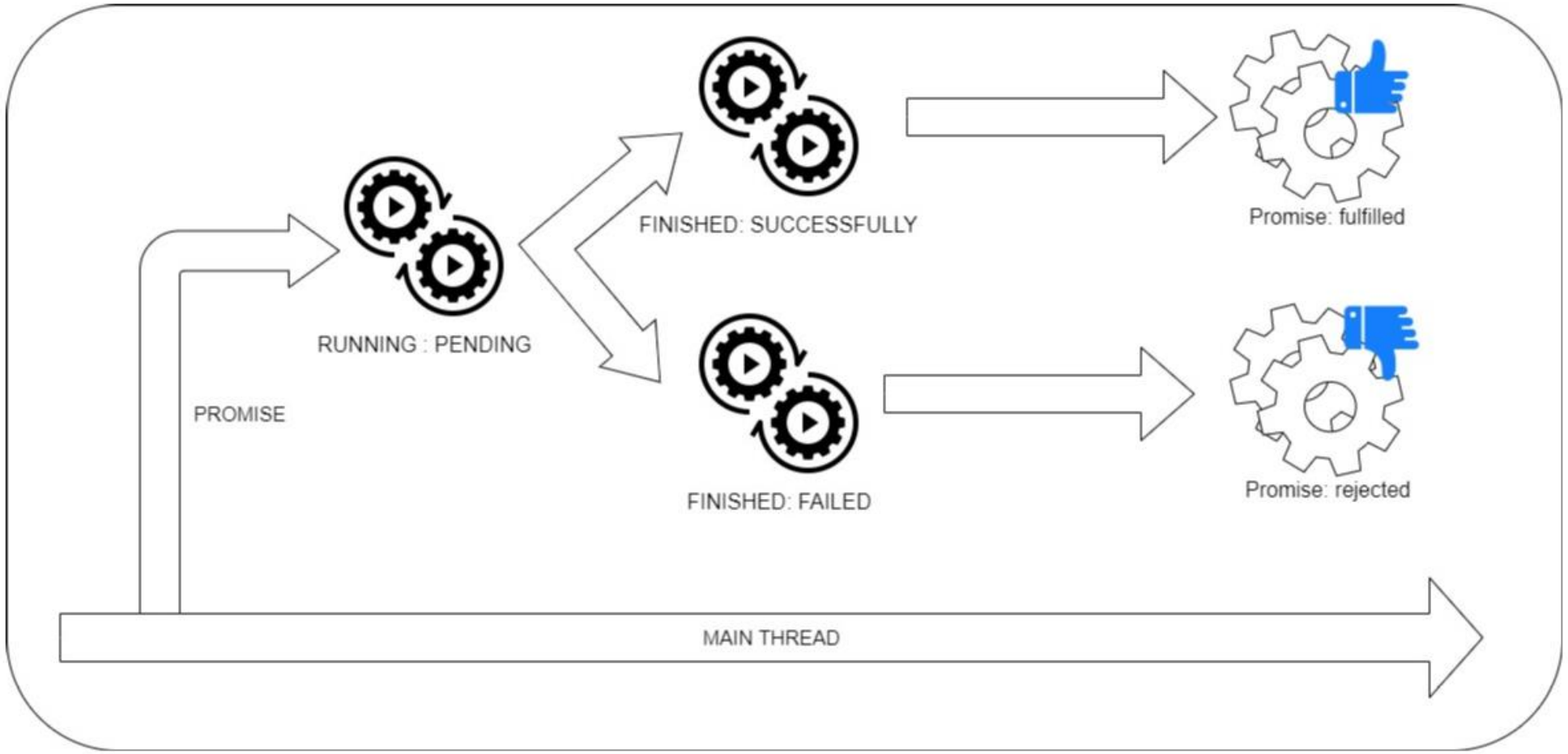
04 Many Promises

05 Chaining

06 All

07 Race

Promises


FINISHED: SUCCESSFULLY
Promise: fulfilled
RUNNING : PENDING
PROMISE
FINISHED: FAILED
Promise: rejected
MAIN THREAD


- Are we there yet?
- No!

Basic

```
let promise = new Promise(
    function(resolve, reject){
        console.log('promise code executed');
        setTimeout(resolve, 500, 'Karol Rogowski');
    }
);
```

Basic

```
let promise = new Promise(
    function(resolve, reject){
        console.log('promise code executed');
        setTimeout(resolve, 500, 'Karol Rogowski');
    }
);

promise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```

Basic

```
let promise = new Promise(
    function(resolve, reject){
        console.log('promise code executed');
        setTimeout(reject, 500, 'Karol Rogowski');
    }
);

promise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```

Basic

```
let promise = new Promise(
    function(resolve, reject){
        console.log('promise code executed');
        setTimeout(reject, 500, 'Karol Rogowski');
    }
);

promise.catch(
    error => console.log('rejected: ' + error)
);

promise.then(
    value => console.log('fullfilled: ' + value)
);

promise.catch(
    error => console.log('rejected2: ' + error)
);
```

Basic

```
let promise = new Promise(
    function(resolve, reject){
        console.log('promise code executed');
        setTimeout(reject, 500, 'Karol Rogowski');
    }
);

promise.catch(
    error => console.log('rejected: ' + error)
);

promise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected3: ' + error)
);

promise.catch(
    error => console.log('rejected2: ' + error)
);
```

Basic

```
let promise = new Promise(
    function(resolve, reject){
        setTimeout(resolve, 1000, 'Karol Rogowski');
    }
);

console.log('before handle');

promise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);

console.log('after handle');
```

Basic

```
var trustworthy = false; // true
let promise = new Promise(function(resolve, reject) {
    if (trustworthy) {
        resolve("The person is trustworthy");
    } else {
        reject("The person can't be trusted");
    }
});


promise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```

Basic

```
var trustworthy = true;
let promise = new Promise(function(resolve, reject) {
setTimeout(function() {
    if (trustworthy) {
        resolve(
            {
                value: "The person is trustworthy",
                code: "CD1_TPIT"
            });
    } else {
        reject(
        {
            value:"The person can't be trusted",
            code: "CD2_TPCNBT"
        });
    }
    }, 1000);
});

promise.then(
value => console.log('fullfilled1: ' + JSON.stringify(value)),
error => console.log('rejected1: ' + JSON.stringify(error))
)
```

Basic

```
                value:"The person can't be trusted",
                code: "CD2_TPCNBT"
            });
        }
    }, 1000);
});

promise.then(
    value => console.log('fullfilled1: ' + JSON.stringify(value)),
    error => console.log('rejected1: ' + JSON.stringify(error))
)

promise.then(
    value => console.log('fullfilled2: ' + JSON.stringify(value)),
    error => console.log('rejected2: ' + JSON.stringify(error))
);
```

Static Result

```
var resolvedPromise = Promise.resolve(123);

resolvedPromise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```

Static Result

```
var rejestedPromise = Promise.reject(321);

rejestedPromise.then(
    value => console.log('fullfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```

Promise Generator

```
var promiseRes = function(n = 0) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve({
                resolvedAfterNSeconds: n
            });
        }, n * 1000);
    });
};


let promiseResolved = promiseRes(2);
promiseResolved.then(function(value) {
    console.log("Value when promise is resolved : ", value);
},function(reason) {
    console.log("Reason when promise is rejected : ", reason);
});
```

Promise Generator

```
var promiseRej = function(n = 0) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            reject({
                rejectedAfterNSeconds: n
            });
        }, n * 1000);
    });
};

let promiseRejected = promiseRej(2);
promiseRejected.then(function(value) {
    console.log("Value when promise is resolved : ", value);
},
function(reason) {
    console.log("Reason when promise is rejected : ", reason);
});
```

Many promises 1/2

```
var generatePromise = function(id) {
    return new Promise(function(resolve, reject) {
        let randomNumberOfSeconds = getRandomNumber(2, 10);
        setTimeout(function() {
            let randomiseResolving = getRandomNumber(1, 10);
            if (randomiseResolving > 5) {
                resolve({
                    ordernumber: id,
                    randomNumberOfSeconds: randomNumberOfSeconds,
                    randomiseResolving: randomiseResolving
                });
            } else {
                reject({
                    ordernumber: id,
                    randomNumberOfSeconds: randomNumberOfSeconds,
                    randomiseResolving: randomiseResolving
                });
            }
        }, randomNumberOfSeconds * 1000);
    });
};
```

Many promises 2/2

```
for (i=1; i<=10; i++) {
    let promise = generatePromise(i);

    promise.then(function(value) {
        console.log("Value when promise is resolved : ", value);
    },function(reason) {
        console.log("Reason when promise is rejected : ", reason);
    });
}
```

Chaining 1/2

```
var promiseRes = function(n = 0, info) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve({
                resolvedAfterNSeconds: n,
                info: info
            });
        }, n * 1000);
    });
};
```

Chaining 2/2

```
let promise1 = promiseRes(2, 'Main level');
promise1.then(
    function(value){
        console.log(value);
        return promiseRes(1,'FirstLevel’);
    }
).then(
    function(value){
        console.log(value);
        return promiseRes(3,'Second Level’);
    }
).then(
    function(value){
        console.log(value);
        return promiseRes(1,'Final Level’);
    }
).then(
    function(value){
        console.log(value);
    }
)
```

All 1/4

```
var promiseRes = function(n = 0, info) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve({
                resolvedAfterNSeconds: n,
                info:info
            });
        }, n * 1000);
    });
};

var promiseRej = function(n = 0, info) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            reject({
                rejectedAfterNSeconds: n,
                info:info
            });
        }, n * 1000);
    });
};
```

All 2/4

```
var promises = [];
promises.push(promiseRes(2, 'Promise 1'));
promises.push(promiseRes(1, 'Promise 2'));
promises.push(promiseRes(3, 'Promise 3'));
promises.push(promiseRes(4, 'Promise 4'));

var handleAllPromises = Promise.all(promises);
handleAllPromises.then(function(values) {
        console.log("All the promises are resolved", values);
    },
    function(reason) {
        console.log("One of the promises failed with the following reason", reason);
});
```

All 3/4

```
var promises = [];

var handleAllPromises = Promise.all(promises);
handleAllPromises.then(function(values) {
        console.log("All the promises are resolved", values);
    },
    function(reason) {
        console.log("One of the promises failed with the following reason", reason);
});
```

All 4/4

```
var promises = [];
promises.push(promiseRes(2, 'Promise 1'));
promises.push(promiseRes(1, 'Promise 2'));
promises.push(promiseRej(3, 'Promise 3'));
promises.push(promiseRej(4, 'Promise 4'));
promises.push(promiseRes(2, 'Promise 5'));

var handleAllPromises = Promise.all(promises);
handleAllPromises.then(function(values) {
        console.log("All the promises are resolved", values);
    },
    function(reason) {
        console.log("One of the promises failed with the following reason", reason);
});
```

Race 1/3

```
var promiseRes = function(n = 0, info) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve({
                resolvedAfterNSeconds: n,
                info:info
            });
        }, n * 1000);
    });
};

var promiseRej = function(n = 0, info) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            reject({
                rejectedAfterNSeconds: n,
                info:info
            });
        }, n * 1000);
    });
};
```

Race 2/3

```
var promises = [];
promises.push(promiseRes(2, 'Promise 1'));
promises.push(promiseRes(1, 'Promise 2'));
promises.push(promiseRej(3, 'Promise 3'));
promises.push(promiseRej(4, 'Promise 4'));
promises.push(promiseRes(2, 'Promise 5'));

var handleRacePromises = Promise.race(promises);
handleRacePromises.then(function(values) {
        console.log("First resolve", values);
    },
    function(reason) {
        console.log("First reject", reason);
});
```

Race 3/3

```
var promises = [];
promises.push(promiseRes(2, 'Promise 1'));
promises.push(promiseRej(1, 'Promise 2'));
promises.push(promiseRes(3, 'Promise 3'));
promises.push(promiseRej(4, 'Promise 4'));
promises.push(promiseRes(2, 'Promise 5'));

var handleRacePromises = Promise.race(promises);
handleRacePromises.then(function(values) {
        console.log("First resolve", values);
    },
    function(reason) {
        console.log("First reject", reason);
});
```