STW

Exercicis de NodeJS

Paula Sarqui

Universitat Autònoma
de Barcelona

Índex

1 Exercici 1 **3**

2 Exercici 2 **3**

3 Exercici 3 **3**

4 Exercici 4 **3**

5 Exercici 5 **3**

6 Exercici 6 **4**

7 Exercici 7 **4**

8 Exercici 8 **4**

9 Exercici 9 **5**

10 Exercici 10 **5**

11 Exercici 11 **5**

12 Exercici 12 **6**

13 Exercici 13 **7**

14 Exercici 14 **7**

15 Exercici 15 **8**

16 Exercici 16 **9**

17 Exercici 17 **9**

18 Exercici 18 **11**

19 Exercici 19 **12**

20 Exercici 20 **12**

21 Exercici 21 **13**

22 Exercici 22 **14**

23 Exercici 23 **14**

24 Exercici 24 **16**

25 Exercici 25 **17**

26 Exercici 26 **17**

27 Exercici 27 **18**

28 Exercici 28 **18**

29 Exercici 29 **19**

30 Exercici 30 **20**

31 Exercici 31 **20**

32 Exercici 32 **21**

33 Disclaimer **22**

34 Bibliografia **23**

1 Exercici 1

```
let f1 = function (a) {
    console.log(a);
};

f1(3);
```

2 Exercici 2

```
let f2 = function (a) {
    return (a >= 0) ? (2 * a) : -1;
};

console.log(f2(-2));
```

3 Exercici 3

```
let f3 = function (llista) {
    let llista2 = llista.map((x) => x + 23);
    return llista2;
};

let llista2 = f3([1,2,3]);
console.log(llista2);
```

4 Exercici 4

```
console.printaki = function () {
    console.log("aqui");
};

console.printaki();
```

5 Exercici 5

```
let f4 = function (a, b) {
    return (a + b);
};

let llistaA = [1,2,3,4];
let llistaB = llistaA.map((x) => f4(x,23));

console.log(llistaB);
```

6 Exercici 6

```
let f2 = function (a) {
    return (a >= 0) ? (2 * a) : -1;
};

let f5 = function (a, b, c) {
  c(b(a));
};

f5(1, f2, console.log);
```

7 Exercici 7

```
console.printaki2 = (function () {
    let count = 0;
    return (function () {
        count++;
        console.log("aqui " + count);
    });
})();

console.printaki2();
console.printaki2();
```

8 Exercici 8

```
const fs = require('fs');

let f6 = function (llista, callback_final) {
    let resultat = [];

    llista.forEach((element) =>
        fs.readFile('./' + element, 'utf8', (err, data) => {
        if (err) {
            throw err;
        }

        resultat.push(data);

        if (resultat.length === llista.length) {
            callback_final(resultat);
        }
    }));
};
```

```
let llista_arxius = ['a1.txt','a2.txt'];

f6(llista_arxius, function (res) {console.log(res);});
```

9 Exercici 9

```
const fs = require('fs');

let f7 = function (llista, callback_final) {
    let resultat = [];

    llista.forEach((element, index, array) =>
        fs.readFile('./' + element, 'utf8', (err, data) => {
        if (err) {
            throw err;
        }

        resultat.splice(index, 0, data);

        if (resultat.length === llista.length) {
            callback_final(resultat);
        }
    }));
};

let llista_arxius = ['a1.txt','a2.txt'];

f7(llista_arxius, function (res) {console.log(res);});
```

10 Exercici 10

Podría i segurament hauría problmes al utilitzar una variables de tipus 'var' com a comptador modificant-la adins del for each perque és una variable global a nivell de fils d'execusió. Si es modifica en un fil o *thread*, es modifica per tots els altres. En canvi, utilitzant la variable index proveïda per la funció 'forEach', interna de JavaScript, la condició de carrera[1] no es dona.

11 Exercici 11

```
const fs = require('fs');

function asyncMap(list, f, callback_final) {
    let resultList = [];
```

```
    let counter = 0;
    let callback_error = false;

    list.map((file, index) => f('./' + file, 'utf8', (err, data) => {
        if (err && !callback_error) {
            callback_final(err, null);
            callback_error = true;
            throw err;
        } else {
            resultList[index] = data;
            counter++;

            if (counter === list.length) {
                callback_final(resultList);
            }
        }
    }));
}

asyncMap(['a1.txt'], fs.readFile, function (a) { console.log(a) });
```

12 Exercici 12

```
let o = class {
    constructor() {
        this.count = 0;
        this.notify = null;
    }

    inc() {
        this.count++;

        if ((this.notify !== null) && (this.notify instanceof Function)) {
            this.notify(this.count);
        }
    }
}

let o1 = new o();

o1.count = 1;

o1.notify = function (a) {
    console.log(a);
};
```

```
o1.inc();
```

13 Exercici 13

```
var o2 = (function () {
    this.count = 1;
    this.notify = null;

    this.setNotify = function (f) {
        notify = f
    };

    this.increment = function () {
        count++;

        if ((notify !== null) && (notify instanceof Function)) {
            notify(count);
        }
    }

    this.getCount = function () {
        return count;
    };

    return {
        inc: increment,
        count: getCount,
        setNotify: setNotify
    }
})();

o2.setNotify(function (a) { console.log(a) });

o2.inc();
```

14 Exercici 14

```
function Counter () {
    this.a = 1;
    this.inc = function () {
        this.a++;

        if (this.notify !== null) {
```

```
            this.notify(this.a);
        }
    };
    this.count = function () { return this.a };
    this.notify = null;
}

let o3 = new Counter();

o3.notify = console.log;

o3.inc();
```

15 Exercici 15

```
function Counter (){
    this.a = 1;
    this.inc = function () {
        this.a++;

        if (this.notify !== null) {
            this.notify(this.a);
        }
    };
    this.count = function () { return this.a };
    this.notify = null;
}

function DecreasingCounter () {
    this.inc = function () {
        this.a--;

        if (this.notify !== null) {
            this.notify(this.a);
        }
    };
}
DecreasingCounter.prototype = new Counter();

let o4 = new DecreasingCounter();

o4.notify = console.log;

o4.inc();
```

16 Exercici 16

```
const fs = require('fs');

function aFuture (data) {
    this.result = ((data === undefined) || (data === null))? null:data;
    this.isDone = this.result !== null;

    return {
        isDone: this.isDone,
        result: this.result
    }
}

let future = null;

function readIntoFuture(filename) {
    fs.readFile('./' + filename, 'utf-8', function (err, data) {
        if (err) {
            throw err;
        }
        future = new aFuture(data);
    });

    return new aFuture();
}

future = readIntoFuture('a1.txt');
console.log(future);

setTimeout(function () {
    console.log(future);
}, 1000);
```

17 Exercici 17

```
const fs = require('fs');

function aFuture (data) {
    this.result = ((data === undefined) || (data === null))? null:data;
    this.isDone = this.result !== null;

    return {
        isDone: this.isDone,
        result: this.result
```

```
    }
}

let future = null;

function asyncToFuture(f) {
    return (function readIntoFuture(filename) {
        f('./' + filename, function (err, data) {
            if (err) {
                throw err;
            }
            future = new aFuture(data);
        });

        return new aFuture();
    });
}

function rIFuture2 () {
    let readIntoFuture2 = asyncToFuture(fs.readFile);

    future = readIntoFuture2('a1.txt');
    console.log("ReadIntoFuture 2: ");
    console.log(future);

    setTimeout(function () {
        console.log(future);
    }, 1000);
}

function rISFuture () {
    let statIntoFuture = asyncToFuture(fs.stat);
    future = statIntoFuture('a1.txt');

    console.log("Stat future: ");
    console.log(future);

    setTimeout(function () {
        console.log(future);
    }, 1000);
}

rIFuture2();
//rISFuture();
```

18 Exercici 18

```
const fs = require('fs');

let EnhancedFuture = (function () {
    let singleInstance = null;
    let callback = null;

    return (function (data) {
        this.isDone = false;
        this.result = null;
        this.registerCallback = (function (f) {
            callback = f;

            if (this.result) {
                f(singleInstance);
            }
        });

        if (data) {
            this.result = data;
            this.isDone = true;

            singleInstance = this;

            if (this.result && (typeof callback === 'function')) {
                console.log("dataEnhancedFuture: " + this.result);
                callback(singleInstance);
            }
        }

        singleInstance = this;

        if (singleInstance) {
            return singleInstance;
        }
    });
})();

let enhancedFuture = new EnhancedFuture();

function asyncToEnhancedFuture (f) {
    return (function (filename) {
        f("./" + filename, "utf-8", function (err, data) {
            if (err) {
                throw err;
```

```
            }
            enhancedFuture = new EnhancedFuture(data);
        });

        return new EnhancedFuture();
    });
}

let readIntoEnhancedFuture = asyncToEnhancedFuture(fs.readFile);
enhancedFuture = readIntoEnhancedFuture('a1.txt');
enhancedFuture.registerCallback(console.log);
```

19 Exercici 19

```
const fs = require('fs');

let when = function (f) {
    this. do = function (g) {
        f(g);
    };

    return {
        do: this.do
    };
};

let f1 = function (callback) { fs.readFile('a1.txt', 'utf-8', callback) };
let f2 = function (error, result) { console.log(result) };

when(f1).do(f2);
```

20 Exercici 20

```
const fs = require('fs');

let when = function (f) {
    this.promises = [];
    let err1;
    let err2;
    let res1;
    let res2;

    this.promises.push(new Promise((resolve) => {
        f((error, result) => {
```

```
            err1 = error;
            res1 = result;

            resolve();
        });
    }));

    this.and = function (g) {

        this.promises.push(new Promise((resolve) => {
            g((error, result) => {
                err2 = error;
                res2 = result;

                resolve();
            });
        }));

        return this;
    };

    this.do = function (h) {
        Promise.all(this.promises).then(() => {
            h(err1, err2, res1, res2);
        });
    };

    return this;
};

let f1 = function (callback) { fs.readFile('a1.txt', 'utf-8', callback) };
let f2 = function (callback) { fs.readFile('a2.txt', 'utf-8', callback) };
let f3 = function (err1, err2, res1, res2) { console.log(res1, res2) };

when(f1).and(f2).do(f3);
```

21 Exercici 21

```
let composer = function (f1, f2) {
    return (function (a) {
        return f1(f2(a));
    });
};

let f1 = function (a) { return a + 1; };
```

```
let f3 = composer(f1, f1);

console.log(f3(3));

let f4 = function (a) { return a * 3; };
let f5 = composer(f3, f4);

console.log(f5(3));
```

22 Exercici 22

```
let asyncComposer = function (f1, f2) {
    return function (a, b) {
        f1(a, function (error, result) {
            f2(result, function (err, res) {

                b(err, res);
            });
        });
    };
};

let f1 = function (a, callback) { callback(null, a + 1); };
let f3 = asyncComposer(f1, f1);

f3(3, function (error, result) { console.log(result) });

f1 = function (a, callback) { callback(null, a + 1) };
let f2 =  function (a, callback) { callback("error", "") };
f3 = asyncComposer(f1, f2);
f3(3, function (error, result) { console.log(error, result); });
```

23 Exercici 23

```
let p;

/**
 * Apartat 'a'
 *
 * La promessa 'p' es resol amb x = 0;
 * Al primer 'then', tenim que 'x' ens arriba a 0 i fem:
 * x = x + 1 = 1
 * Al segon 'then' fem:
 * x = x + 2 = 3
```

```
 * Al tercer 'then' fem:
 * x = x + 4 = 7
 * Es finalitza l'execusió i es printa el valor de 'x' (x = 7);
 */
p = Promise.resolve(0)
    .then(x => x + 1)
    .then(x => x + 2)
    .then(x => x + 4);
p.then((x) => { console.log("a: " + x) });
/**
 * Apartat 'b'
 *
 * La promesa 'p' es rebutja amb x = 0;
 * Al primer 'then' tenim que, com s'ha rebutjat la promesa,
 * ens el saltem i anem al 'catch'.
 * Al 'catch' tenim que ens arriba 'x' amb valor 0 i fem:
 * x = x + 2 = 2
 * Al segon 'then' entrem perque ja hem fet un 'catch' i aquest
 * 'then' s'ha d'executar a continuació. Fem:
 * x = x + 4 = 6
 * Es finalitza l'execusió i es printa el valor de 'x' (x = 6);
 */
p = Promise.reject(0)
    .then(x => x + 1)
    .catch(x => x + 2)
    .then(x => x + 4);
p.then((x) => { console.log("b: " + x) });
/**
 * Apartat 'c'
 *
 * La promessa 'p' es resol amb x = 0;
 * Al primer 'then', tenim que 'x' ens arriba a 0 i fem:
 * x = x + 1 = 1
 * Al segon 'then' fem:
 * x = x + 2 = 3
 * El 'catch' no s'execute perque no ha hagut 'reject'.
 * Al tercer 'then', tenim que 'x' ens arriba a 3 i fem:
 * x = x + 8 = 11
 * Es finalitza l'execusió i es printa el valor de 'x' (x = 11);
 */
p = Promise.resolve(0)
    .then(x => x + 1)
    .then(x => x + 2)
    .catch(x => x + 4)
    .then(x => x + 8);
p.then((x) => { console.log("c: " + x) });
```

```
/**
 * La promesa 'p' es rebutja amb x = 0;
 * Ni el primer 'then' ni el segon s'executen;
 * Al 'catch' tenim que x = 0 i fem:
 * x = x + 4;
 * Al tercer 'then' tenim que 'x' ens arriba a 4 i fem:
 * x = x + 8 = 12;
 * Es finalitza l'execusió i es printa el valor de 'x' (x = 12);
 */
p = Promise.reject(0)
    .then(x => x + 1)
    .then(x => x + 2)
    .catch(x => x + 4)
    .then(x => x + 8);
p.then((x) => { console.log("d: " + x) });
/**
 * La promesa 'p' es rebutja amb x = 0;
 * Al primer 'then' fem una operació 'null' per promessa rebutjada,
 * és a dir, no fem res;
 * En aquest moment tenim que x = null;
 * Al primer 'catch' fem:
 * x = x + 2 = null + 2 = 2;
 * El segon 'catch' ens el saltem;
 * Es finalitza l'execusió i es printa el valor de 'x' (x = 2);
 */
p = Promise.reject(0)
    .then(x => x + 1, null)
    .catch(x => x + 2)
    .catch(x => x + 4);
p.then((x) => { console.log("e: " + x) });
```

24 Exercici 24

```
let antipromise = function (promise) {
    return new Promise((resolve, reject) => {
        promise.then((x) => {
            reject(x);
        }).catch((x) => {
            resolve(x);
        });
    });
};

antipromise(Promise.reject(0))
    .then((x) =>{ console.log("Antipromise resolved, x = " + x) });
```

```
antipromise(Promise.resolve(1))
    .catch((x) => { console.log("Antipromise rejected, x = " + x) });
```

25 Exercici 25

```
let promiseToCallback = function (f) {
    return (function (x, callback) {
        f(x).then((res) => callback(null, res), (res) =>
            callback(res, null));
    });
};

let isEven = x => new Promise((resolve, reject) => {
    x % 2 ? reject(x) : resolve(x);
});

let isEvenCallback = promiseToCallback(isEven);

isEven(2).then(() => console.log("OK"), () => console.log("KO"));
isEvenCallback(2, (err, res) => console.log(err, res));
isEven(3).then(() => console.log("OK"), () => console.log("KO"));
isEvenCallback(3, (err, res) => console.log(err, res));
```

26 Exercici 26

```
const fs = require('fs');

let readToPromise = function (file) {
    return new Promise((resolve, reject) => {
        fs.readFile(file, (err, data) => {
            if (err) {
                reject(err);
            } else {
                resolve(data)
            }
        });
    });
};

readToPromise("a1.txt")
    .then(x => console.log("Contents: ", x))
    .catch(x => console.log("Error: ", x));
readToPromise("notfound.txt")
    .then(x => console.log("Contents: ", x))
```

```
    .catch(x => console.log("Error: ", x));
```

27 Exercici 27

```
let fs = require('fs');

let callbackToPromise = function (f) {
    return (function (file) {
        return new Promise((resolve, reject) => {
            f(file, (err, data) => {
                if (err) {
                    reject(err);
                } else {
                    resolve(data);
                }
            });
        });
    });
};

let readToPromise2 = callbackToPromise(fs.readFile);

readToPromise2("a1.txt")
    .then(x => console.log("Contents: ", x))
    .catch(x => console.log("Error: ", x));
```

28 Exercici 28

```
const fs = require('fs');

function asyncToEnhancedFuture (f) {
    return (function (file) {
        let callback = null;
        let future = {
            isDone: false,
            result: null,
            registerCallback: function (cb) {
                if (future.isDone) {
                    cb(future);
                } else {
                    callback = cb;
                }
            }
        };
```

```
        f("./" + file, "utf-8", function (err, data) {
            if (err) {
                throw err;
            } else {
                future.result = data;
                future.isDone = true;

                if (callback != null) {
                    callback(future);
                }
            }
        });

        return future;
    });
}

let enhancedFutureToPromise = function (enhancedFuture) {
    return new Promise((resolve, reject) => {
        enhancedFuture.registerCallback((future) => {
            resolve(future.result);
        });
    });
};

readIntoEnhancedFuture = asyncToEnhancedFuture(fs.readFile);
let enhancedFuture = readIntoEnhancedFuture('a1.txt');
let promise = enhancedFutureToPromise(enhancedFuture);
promise.then(console.log);
```

29 Exercici 29

```
let mergedPromise = function (promise) {
    return new Promise((resolve) => {
        promise.then(resolve, resolve);
    });
};

mergedPromise(Promise.resolve(0)).then(console.log);
mergedPromise(Promise.reject(1)).then(console.log);
```

30 Exercici 30

```
let functionPromiseComposer = function (f1, f2) {
    return function (x) {
        return new Promise((resolve, reject) => {
            f2(x).then((res) => {
                f1(res).then(resolve).catch(reject);
            }).catch(reject);
        });
    };
};

let f1 = x => new Promise((resolve, reject) =>
    resolve(x + 1));
let f2 = x => new Promise((resolve, reject) =>
    reject('always fails'));
let f3 = x => new Promise((resolve, reject) =>
    setTimeout(() => resolve(x * 2), 500));

functionPromiseComposer(f1, f1)(3).then(console.log);
functionPromiseComposer(f1, f2)(3).catch(console.log);
functionPromiseComposer(f1, f3)(3).then(console.log);
```

31 Exercici 31

```
let parallelPromise = function (promise1, promise2) {
    return new Promise((resolve) => {
        Promise.all([promise1, promise2]).then((results) => {
            resolve(results);
        });
    });
};

let p1 = parallelPromise(Promise.resolve(0), Promise.resolve(1));

p1.then(console.log);

let plast = new Promise((resolve) => {
    setTimeout(() => { resolve(0) }, 200);
});

let pfirst = new Promise((resolve) => {
    setTimeout(() => { resolve(1) }, 100);
});
```

```
let p2 = parallelPromise(plast, pfirst);

p2.then(console.log);
```

32 Exercici 32

```
let promiseBarrier = function(x) {
    let list = [];
    let counter = 0;
    let params = [];
    let executedFunctions = [];

    for (let i = 0; i < x; i++) {
        list[i] = function (n) {

            return new Promise((resolve) => {
                counter++;
                executedFunctions[i] = resolve;
                params[i] = n;

                if (counter === x) {
                    for(let j = 0; j < x; j++) {
                        executedFunctions[j](params[j]);
                    }
                }
            });
        };
    }

    return list;
};

let test1 = function () {
    let [f1, f2, f3] = promiseBarrier(3);

    Promise.resolve(0)
        .then(x => { console.log("c1 s1"); return x; })
        .then(x => { console.log("c1 s2"); return x; })
        .then(x => { console.log("c1 s3"); return x; })
        .then(f1)
        .then(x => { console.log("c1 s4"); return x; })
    Promise.resolve(0)
        .then(x => { console.log("c2 s1"); return x; })
        .then(f2)
        .then(x => { console.log("c2 s2"); return x; })
```

```
    Promise.resolve(0)
        .then(f3)
        .then(x => { console.log("c3 s1"); return x; })
        .then(x => { console.log("c3 s2"); return x; })
        .then(x => { console.log("c3 s3"); return x; })
};

let test2 = function () {
    let [f4, f5] = promiseBarrier(2);
    Promise.resolve(1).then(f4).then(console.log);
    Promise.resolve(2).then(f5).then(console.log);
};

test1();
//test2();
```

33 Disclaimer

A aquesta secció es troben comentaris necessaris a tenir en compte. Primer de tot, he de dir que, per facilitar la disponibilitat del codi per la seva correcció, he anar publicant-los a un repositori públic[2]. També cal mencionar que aquests exercicis ja els vaig realitzar l'any passat, amb contribució del meu company de pràctiques, la versió de l'any passat també es troba disponible *on-line*[3], els vaig utilitzar de referència en alguns casos però, fent la comparació es pot veure la diferència.

Per últim l'únic exercici que no es un arxiu amb extensió '.js' és l'exercici 10 perquè he considerat que, com era d'explicació no calia, pel que té un format '.txt'.

34 Bibliografia

[1] Condición de carrera. URL https://github.com/miguelinux314/experiment-Notebook.

[2] P. Sarqui. Stw-problemes-nodejs. URL https://github.com/AlysH/STW-Problemes-nodejs.

[3] P. Sarqui A. Hurtado. node. URL https://github.com/ahurtado92/ProblemesSTW/tree/master/node.