WP, predavanje 10, Promise, 2025./2026.

1. Inicijalizirati Node.js projekt, instalirati `nodemon`

2. U `index.js`:

```
const promise = new Promise((resolve, reject) => {

    // reject("Something's wrong");

    setTimeout(() => {
        const result = 10 / 0;
        resolve(result);
    }, 1000);
});

promise.then((result) => {
    console.log(result);
}, (error) => {
    console.error(error);
});

let faux_data = { user: "jmaltar", age: 32 };
let another_promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve(faux_data);
    }, 1500);
});

// kad se svi završe
Promise.all([ promise, another_promise ]).then((data) => {
    console.log(data);
}, err => { console.log(err) });

// čim se jedan završi
Promise.race([ promise, another_promise ]).then((data) => {
    console.log(data);
});
```

3. Trenutni kod spremiti u zasebnu datoteku. Kopirati `ms`, `threshold` i `UserDB` iz `copy.txt`.

4. Primjer loše prakse:

```
const ms = 500;
const threshold = 0.1;

class UserDB {
    static is_from_EU(id, s_callb, e_callb) {
        setTimeout(() => {
            let prob = Math.random();
            if (prob >= threshold)
                s_callb(true);
            else e_callb("Something went wrong (EU)");
        }, ms);
    }
    …
}

// callback hell + dugo čekamo

function get_user(id) {
    UserDB.get_username(id, (username) => {
        UserDB.is_legal_age(id, (legal_age) => {
            UserDB.is_from_EU(id, (from_EU) => {
                console.log(username, legal_age, from_EU);
            }, (error) => {
                console.log(error);
            });
        }, (error) => {
            console.log(error);
        });
    }, (error) => {
        console.log(error);
    });
}

get_user();
```

5. Dobra praksa:

```
class UserDB {
    static is_from_EU(id) {
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                let prob = Math.random();
                if (prob >= threshold) resolve(true);
                else reject("Something went wrong (EU)");
            }, ms);
        });
    }
    static is_legal_age(id) {
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                let prob = Math.random();
                if (prob >= threshold) resolve(true);
                else reject("Something went wrong (AGE)");
            }, ms);
        });
    }
    static get_username(id) {
        return new Promise((resolve, reject) => {
            setTimeout(() => {
                let prob = Math.random();
                if (prob >= threshold) resolve("jmaltar")
                else reject("Something went wrong (USER)");
            }, ms);
        });
    }
}

const get_user = (id) => {
    Promise.all([
        UserDB.is_from_EU(id),
        UserDB.is_legal_age(id),
        UserDB.get_username(id)
    ]).then((values) => {
        console.log(values);
    }).catch((error) => {
        console.error(error);
    });
}

get_user();
```

6. Dobra praksa (async + await):

```
const get_user = async (id) => {
    const user = await Promise.all([
        UserDB.is_from_EU(id),
        UserDB.is_legal_age(id),
        UserDB.get_username(id)
    ]);

    return user;
}

(async () => {
    try {
        const user = await get_user(0);
        console.log(user);
    } catch (error) {
        console.log(error);
    }
})();
```

7. Primjer učitavanja datoteke:

```
const fs = require("fs");
const fs_p = require("fs/promises");

const fs_synchronously = () => {
    const s = Date.now();
    const f = fs.readFileSync("../video.mp4");
    const e = Date.now();
    console.log(e - s);
    console.log("I'm another operation waiting to be executed");
}

const fs_asynchronously = () => {
    const s = Date.now();
    fs_p.readFile("../video.mp4").then(res => {
        const e_1 = Date.now();
        console.log(e_1 - s);
    });

    const e_2 = Date.now();
    console.log(e_2 - s);
    console.log("I'm another operation waiting to be executed");
}

// fs_synchronously();
fs_asynchronously();
```