UNIVERSITY OF SZEGED
Department of Software Engineering
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS

Practice 5
Promise, asynchronous operations

Topics

- Introducing async operations and the methods to handle them
- Exercise to introduce projects consisting of more modules and async methods

Request to a server

- Operation is required from the server – send user data, store input
- Slow, busy server => should we wait for it's answer?
- Async processing: continue executing our code, we only handle the received data when it's ready
- We handle the request as a promise from the server

Promise

- We can define the methods and processing to do if the data is ready, or if the request failed
- then, catch routes – we can work with the data sent back from the server as arguments
- While the Promise is being answered by the server, the user doesn't have to wait, we can keep working
- RxJs and Angular2 works with even better solutions (Observable) – upgraded the idea of the simple promise

Example

```
site.promiseMe().then(data => {
        changeNumField(<string>data);
 }).catch(error => {
        console.log(error);
 });
```

- The promiseMe() method returns a Promise
- If it was successful (resolved), we change a field to the return value (handled as a string)
- If it wasn't suffessful (rejected), we log the received error object in the console

Example 2

- We can write a Promise on the client side too (we need to wait for a slower processing method for example)
- Important setting in TypeScript, the target must be ES6 (in earlier versions it wasn't native, external libs like Q or BlueBird were needed).

```
return new Promise((resolve, reject) => {
        if(this.num < 10) reject(„Number is too low.");
        resolve(„Number is good.");
})
```

- Resolve sends back success, a reject will result in the catch function running.