COMP6080 WK4 Tutorial 🗺️

Joanna He

Agenda

- Week 3 overview
- Data fetching
- Promises
- Demo

Reminder

Assignment 2 is due tomorrow 5pm 😮

No late submissions are accepted (unless with special consideration)

Any questions?

Last week we covered...

- JavaScript
- Event listeners
- DOM manipulation

Onto data fetching...

HTTP methods

Four main methods to communicate with a server:

- GET
- POST: upload new data
- PUT: replace data
- DELETE

How do we fetch data?

```
function getUsers() {
  return [
    { username: 'kirby', email: 'kirby@test.com' },
    { username: 'charmander', email: 'charmander@test.com' },
  ];
}

function findUser(username) {
  const users = getUsers();
  const user = users.find((user) => user.username === username);
  return user;
}

console.log(findUser('kirby'));
```

But API's are asynchronous...

```
function getUsers() {
  let users = [];

  // delay 1 second (1000ms)
  setTimeout(() => {
    users = [
      { username: 'bulbasaur', email: 'bulbasaur@test.com' },
      { username: 'charmander', email: 'charmander@test.com' },
    ];
  }, 1000);

  return users;
}

function findUser(username) {
  const users = getUsers();
  const user = users.find((user) => user.username === username);
  return user;
}

console.log(findUser('bulbasaur'));
```

Callbacks????

```
function getUsers(callback) {
  setTimeout(() => {
    callback([
      { username: 'baulbasaur', email: 'baulbasaur@test.com' },
      { username: 'charmander', email: 'charmander@test.com' },
    ]);
  }, 1000);
}

function findUser(username, callback) {
  getUsers((users) => {
    const user = users.find((user) => user.username == username);
    callback(user);
  });
}

findUser('baulbasaur', console.log);
```

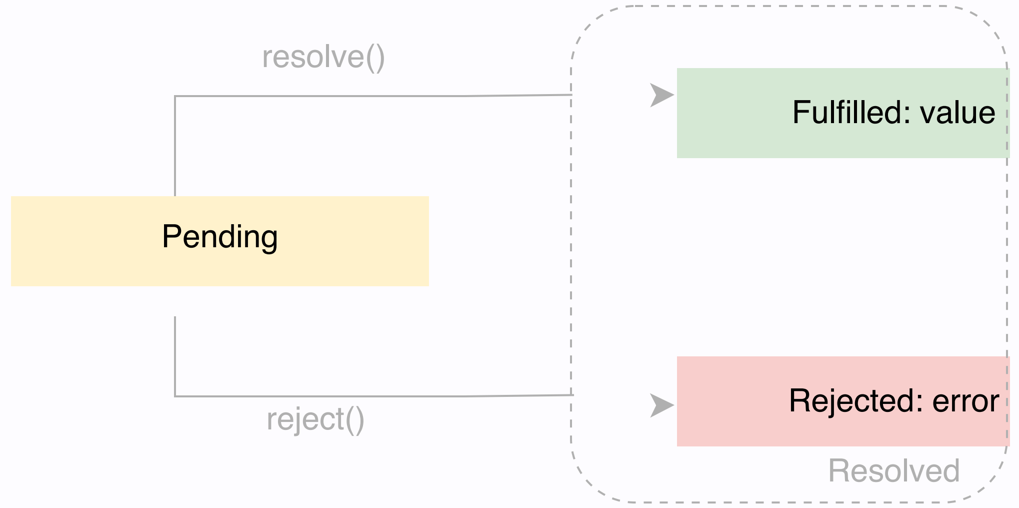
W😮W so beautiful but why?

```javascript
function getUsers() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve([
        { username: 'bulbasaur', email: 'bulbasaur@test.com' },
        { username: 'charmander', email: 'charmander@test.com' },
      ]);
    }, 1000);
  });
}

getUsers().then((users) => {
  console.log(users.find((user) => user.username === 'bulbasaur'));
}).catch((error) => {
  console.error('Error:', error);
});
```

Promises

- An object that encapsulates the result of an asynchronous operation
- Three states - pending, rejected and resolved / fulfilled
- Avoids callback hell



Fetch

- Creates network requests and returns a promise

```
fetch(apiUrl, {
  method: "GET", // by default, sends a get request
  headers: {
    'Content-type': 'application/json',
    'Authorization': `Bearer ${userToken}`
  },
  body: JSON.stringify({  // what we want to send to the api
    username: name.value,
  }),
}).then((res) => res.json()) // resolved case returns another promise
.then((data) => {
  if (!data.ok) doSomething
})
.catch(() => doSomething)  // rejected case
```

Demo

- Let's fetch the first 20 pokemon and append the pokemon names as list tags to the DOM!

Tutorial code can be found at

https://github.com/joanna209/tutoring/tree/main/comp6080/23T3