STANDARDI U PRIMJENI INTERNETSKE TEHNOLOGIJE

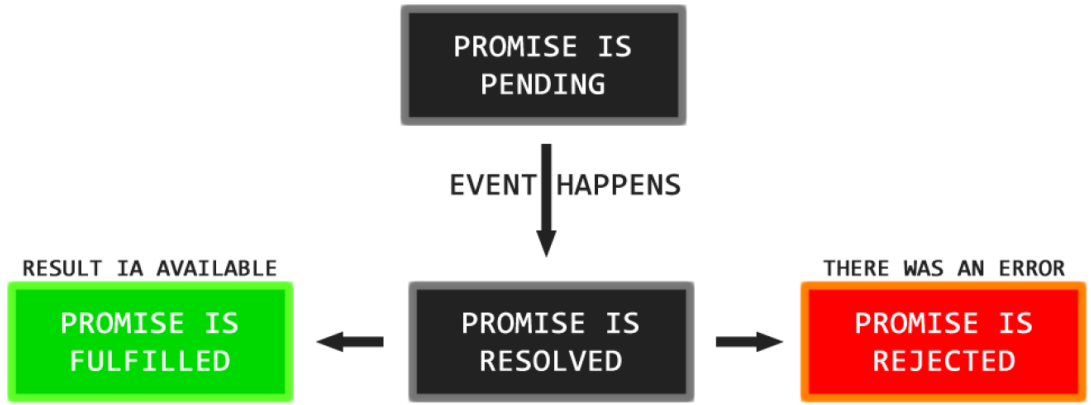
Predavanje 13

Promise

- Promise objekti su ključni dio asinkronog programiranja u JavaScriptu.
- Predstavljaju operaciju koja nije još završena, ali se očekuje da će biti u budućnosti.

Stanja Promise objekta

- **Pending**: Inicijalno stanje, ni ispunjeno (fulfilled) ni odbačeno (rejected).
- **Fulfilled**: Znači da je operacija uspješno završena.
- **Rejected**: Znači da je došlo do greške tijekom operacije.



Promise

- **Kreiranje Promise objekta**
  - Pomoću konstruktora **new Promise()**. Konstruktor prima executor funkciju koja se poziva odmah, a prima kao parametre dvije funkcije: **resolve** i **reject**.

```
const promise = new Promise((resolve, reject) => {
    // Asinkrona operacija
    if (success) {
        resolve(value);
    } else {
        reject(error);
    }
});
```

- **Korištenje Promise objekta**
  - **then**(): za raspoređivanje callbackova za ispunjene (fulfilled) ili odbačene (rejected) promise-e.

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

  - **catch**(): za hvatanje odbačenih (rejected) promise-a.

```
promise
    .then(response => {
        console.log(response);
    })
    .catch(err => console.log(err));
```

  - **finally**(): za izvršavanje koda nakon što je Promise ispunjen ili odbačen.

JSON - JavaScript Object Notation

- Predstavlja lagani format za razmjenu podataka
- Lak za čitanje i pisanje
- Koristi strukturu ključ/vrijednost koja je poznata svim programerima
- Koristi se u web aplikacijama za slanje podataka između klijenta i poslužitelja.
- Popularan je format za konfiguracijske datoteke
- Jednostavnost i jasnoća čine JSON boljim izborom od XML-a za razmjenu podataka

XMLHttpRequest

- XMLHttpRequest (XHR) je JavaScript API koji omogućuje web stranicama da asinkrono komuniciraju s poslužiteljima, tj. da šalju i primaju podatke bez potrebe za osvježavanjem stranice.
- Iako XMLHttpRequest i dalje široko koristi, postepeno ga zamjenjuje moderniji fetch API, koji nudi *promises* i čišću, fleksibilniji sintaksu.

```
const xhr = new XMLHttpRequest();
xhr.open('GET', endpoint, true);
xhr.onload = (e) => {
    if (xhr.status === 200) {
        console.log(xhr.responseText);
    }
}
xhr.onerror = (e)=>{
    console.log('error');
}
xhr.send();
```

fetch

- Predstavlja moderni standard ugrađen u web preglednike, koji omogućava jednostavno izvršavanje mrežnih zahtjeva. Nudi fleksibilniju alternativu starijem XMLHttpRequest.
- fetch vraća Promise koji se razrješava (resolves) u odgovoru (Response) kada se zahtjev dovrši.
- Response objekt sadrži informacije o odgovoru, uključujući status, zaglavlja, i metode za obradu tijela odgovora (kao što su **json**(), **text**(), **blob**()).

```
fetch(endpoint)
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.log(error));
```

- Za prilagodbu zahtjeva, fetch prima drugi argument, objekt s konfiguracijama, koji može uključivati metodu (method), zaglavlja (headers), tijelo (body) i druge postavke.

```
fetch('https://api.example.com/submit', {
    method: 'POST',
    body: JSON.stringify(requestPayload),
    headers: {
        'Content-Type': 'application/json'
    }
})
```

jQuery.ajax

- Metoda unutar jQuery biblioteke koja omogućuje izvršavanje asinkronih HTTP zahtjeva.
- jQuery.ajax prima objekt s mnogim konfiguracijskim opcijama, uključujući:
  - **url**: URL zahtjeva.
  - **method** (ili type): HTTP metoda zahtjeva, npr. "GET", "POST".
  - **data**: Podaci koji se šalju s zahtjevom, korisno za POST zahtjeve.
  - **dataType**: Očekivani tip podataka odgovora, npr. "json", "xml", "script".
  - **success**: Callback funkcija koja se poziva kada zahtjev uspije.
  - **error**: Callback funkcija koja se poziva kada zahtjev ne uspije.
  - **complete**: Callback funkcija koja se poziva kada zahtjev završi, bilo uspješno ili neuspješno.
  - **headers**: Objekt za postavljanje prilagođenih zaglavlja zahtjeva.

```
$.ajax({
    url: endpoint,
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    data: JSON.stringify(requestPayload),
    success: (response) => {
        console.log(response);
    },
    error: (jqXHR, textStatus, errorThrown) => {
        console.log(jqXHR, textStatus, errorThrown);
    },
})
```