

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

Modifiera data i  
server



# Agenda

- Skicka data till server
- Ändra en note i server
- Felhantering
- Övning (server i inlämningsuppgift)
  - Sätt upp mongoDB



# Skicka data till server

För att skicka data behöver vi en route som hanterar post

```
app.post("/notes", (req, resp) => {  
  notes.push(req.body);  
  resp.sendStatus(201);  
});
```



# Skicka data till server

Vi kan prova att skicka en note till servern och se vad som händer?

```
addNote = event => {  
  event.preventDefault()  
  const noteObject = {  
    content: newNote,  
    important: Math.random() < 0.5,  
  }  
  
  axios  
    .post('http://localhost:3001/notes', noteObject)  
    .then(response => {  
      console.log(response)  
    })  
}
```



# Skicka data till server

Vi bör i resultatet från  
servern bestämma när vi kan  
ändra state i frontend...

```
addNote = event => {  
  event.preventDefault()  
  const noteObject = {  
    content: newNote,  
    important: Math.random() > 0.5,  
  }  
  
  axios  
    .post('http://localhost:3001/notes', noteObject)  
    .then(response => {  
      setNotes(notes.concat(response.data))  
      setNewNote('')  
    })  
}
```



# Skicka data till server

## Övning

- Skapa backend route (post) för att ta emot noteobjekt
- Testa routen med Postman eller Insomnia
- Låt formulär (addNote) skicka en post till med note till backend
- Ändra så att frontend state ändras först när backend har skickat respons



# Ändra en note i server

Vi vill nu lägga till en knapp som styr om en note är viktig eller inte. För det syftet skickar vi in en funktion till `Note (toggleImportance)`

```
const Note = ({ note, toggleImportance }) => {  
  const label = note.important  
    ? 'make not important' : 'make important'  
  
  return (  
    <li>  
      {note.content}  
      <button onClick={toggleImportance}>{label}</button>  
    </li>  
  )  
}
```



# Ändra en note i server

Vi börjar med en enkel console.log för att testa

```
const App = () => {  
  const [notes, setNotes] = useState([])  
  const [newNote, setNewNote] = useState('')  
  const [showAll, setShowAll] = useState(true)  
  
  // ...  
  
  const toggleImportanceOf = (id) => {  
    console.log('importance of ' + id + ' needs to be toggled')  
  }  
  
  // ...  
  
  return (  
    <div>  
      <h1>Notes</h1>  
      <div>  
        <button onClick={() => setShowAll(!showAll)}>  
          show {showAll ? 'important' : 'all'}  
        </button>  
      </div>  
      <ul>  
        {notesToShow.map(note =>  
          <Note  
            key={note.id}  
            note={note}  
            toggleImportance={() => toggleImportanceOf(note.id)}  
          </>  
        )}  
      </ul>  
      // ...  
    </div>  
  )  
}
```





# Ändra en note i server

Övning: Sedan gör vi en uppdatering i server

- Skriv funktionen replace
- Testa route med Postman eller Insomnia

```
app.put("/notes/:id", (req, resp) => {  
  const id2Update = Number(req.params.id);  
  const ok = replace(notes, id2Update, req.body);  
  if (ok) {  
    resp.sendStatus(200); // OK  
  } else {  
    resp.sendStatus(404); // Not found  
  }  
});
```



# Ändra en note i server

Nu kan vi ändra toggleImportanceOf

```
const toggleImportanceOf = id => {  
  const url = `http://localhost:3001/notes/${id}`  
  const note = notes.find(n => n.id === id)  
  const changedNote = { ...note, important: !note.important }  
  
  axios.put(url, changedNote).then(response => {  
    setNotes(notes.map(n => n.id !== id ? n : response.data))  
  })  
}
```



# Ändra en note i server

## Övning

- Ändra toggleImportanceOf så att den anropar backend
- Ändra så att frontend state ändras först när backend har skickat respons



# Ändra en note i server

Vid det här laget passar det att extrahera en modul (noteservice) för kommunikationen med backend

```
import axios from 'axios'
const baseUrl = 'http://localhost:3001/notes'

const getAll = () => {
  return axios.get(baseUrl)
}

const create = newObject => {
  return axios.post(baseUrl, newObject)
}

const update = (id, newObject) => {
  return axios.put(`${baseUrl}/${id}`, newObject)
}

export default {
  getAll: getAll,
  create: create,
  update: update
}
```



# Ändra en note i server

Vår applikation får följande utseende

```
const App = () => {  
  // ...  
  
  useEffect(() => {  
    noteService  
      .getAll()  
      .then(response => {  
        setNotes(response.data)  
      })  
  }, [])  
  
  const toggleImportanceOf = id => {  
    const note = notes.find(n => n.id === id)  
    const changedNote = { ...note, important: !note.important }  
  
    noteService  
      .update(id, changedNote)  
      .then(response => {  
        setNotes(notes.map(note => note.id !== id ? note : response.data))  
      })  
  }  
  
  const addNote = (event) => {  
    event.preventDefault()  
    const noteObject = {  
      content: newNote,  
      important: Math.random() > 0.5  
    }  
  
    noteService  
      .create(noteObject)  
      .then(response => {  
        setNotes(notes.concat(response.data))  
        setNewNote('')  
      })  
  }  
  
  // ...  
}  
  
export default App
```



# Ändra en note i server

## Övning

- Skapa en noteService som hanterar alla backend-anrop
- Använd noteService i App istället för direkt anropa servern




# Ändra en note i server

Det vore nice om vi kunde gå från att arbeta med http responses till enbart notes

```
noteService
  .getAll()
  .then(response => {
    setNotes(response.data)
  })
```



```
noteService
  .getAll()
  .then(initialNotes => {
    setNotes(initialNotes)
  })
```



## Ändra en note i server

För detta modifierar vi vår noteservice till följande utseende

```
import axios from 'axios'
const baseUrl = 'http://localhost:3001/notes'

const getAll = () => {
  const request = axios.get(baseUrl)
  return request.then(response => response.data)
}

const create = newObject => {
  const request = axios.post(baseUrl, newObject)
  return request.then(response => response.data)
}

const update = (id, newObject) => {
  const request = axios.put(`${baseUrl}/${id}`, newObject)
  return request.then(response => response.data)
}

export default {
  getAll: getAll,
  create: create,
  update: update
}
```





# Ändra en note i server

```
const App = () => {  
  // ...  
  
  useEffect(() => {  
    noteService  
      .getAll()  
      .then(initialNotes => {  
        setNotes(initialNotes)  
      })  
  }, [])  
  
  const toggleImportanceOf = id => {  
    const note = notes.find(n => n.id === id)  
    const changedNote = { ...note, important: !note.important }  
  
    noteService  
      .update(id, changedNote)  
      .then(returnedNote => {  
        setNotes(notes.map(note => note.id !== id ? note : returnedNote))  
      })  
  }  
  
  const addNote = (event) => {  
    event.preventDefault()  
    const noteObject = {  
      content: newNote,  
      important: Math.random() > 0.5  
    }  
  
    noteService  
      .create(noteObject)  
      .then(returnedNote => {  
        setNotes(notes.concat(returnedNote))  
        setNewNote('')  
      })  
  }  
  
  // ...  
}
```



# Ändra en note i server

Vi kan också använda en mer kompakt syntax

```
import axios from 'axios'
const baseUrl = 'http://localhost:3001/notes'

const getAll = () => {
  const request = axios.get(baseUrl)
  return request.then(response => response.data)
}

const create = newObject => {
  const request = axios.post(baseUrl, newObject)
  return request.then(response => response.data)
}

const update = (id, newObject) => {
  const request = axios.put(`${baseUrl}/${id}`, newObject)
  return request.then(response => response.data)
}

export default { getAll, create, update }
```



# Ändra en note i server

## Övning

- Modifiera er noteservice så att den endast hanterar notes och gömmer response och requests...



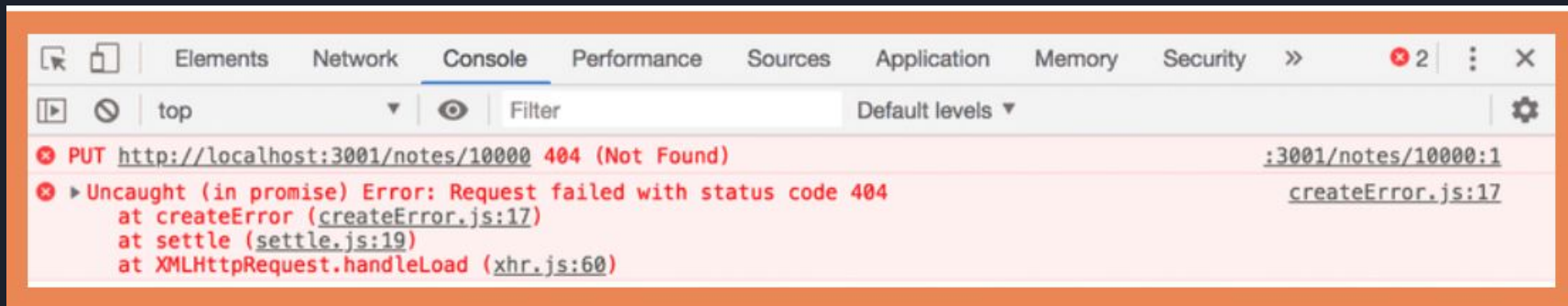
# Felhantering i frontend

Vi vill nu hantera fel i backend koden. Vi simulerar ett fel genom att introducera en påhittad note som backenden inte känner till

```
const getAll = () => {  
  const request = axios.get(baseUrl)  
  const nonExisting = {  
    id: 10000,  
    content: 'This note is not saved to server',  
    important: true,  
  }  
  return request.then(response => response.data.concat(nonExisting))  
}
```

# Felhantering i frontend

Vi får nu följande fel om vi tittar i konsollen





# Felhantering i frontend

För att hantera fel i frontendkoden lägger vi till ett catch block

```
axios
  .get('http://example.com/probably_will_fail')
  .then(response => {
    console.log('success!')
  })
  .catch(error => {
    console.log('fail')
  })
```



# Felhantering i frontend

Med en promise kedja så kan vi hantera alla fel med ett catch block

```
axios
  .put(`${baseUrl}/${id}`, newObject)
  .then(response => response.data)
  .then(changedNote => {
    // ...
  })
  .catch(error => {
    console.log('fail')
  })
```



# Felhantering i frontend

Vi kan nu modifiera vår kod till följande kod

```
const toggleImportanceOf = id => {  
  const note = notes.find(n => n.id === id)  
  const changedNote = { ...note, important: !note.important }  
  
  noteService  
    .update(id, changedNote).then(returnedNote => {  
      setNotes(notes.map(note => note.id !== id ? note : returnedNote))  
    })  
    .catch(error => {  
      alert(  
        `the note '${note.content}' was already deleted from server`  
      )  
      setNotes(notes.filter(n => n.id !== id))  
    })  
}
```





# Övning (server i inlämningsuppgift)

- Arbeta med inlämningsuppgift server
- Sätt upp mongoDB (genomgång)
  - <https://blog.stackademic.com/setting-up-mongodb-with-express-is-a-step-by-step-guide-8eb8fd80d5ff>