

Intro til

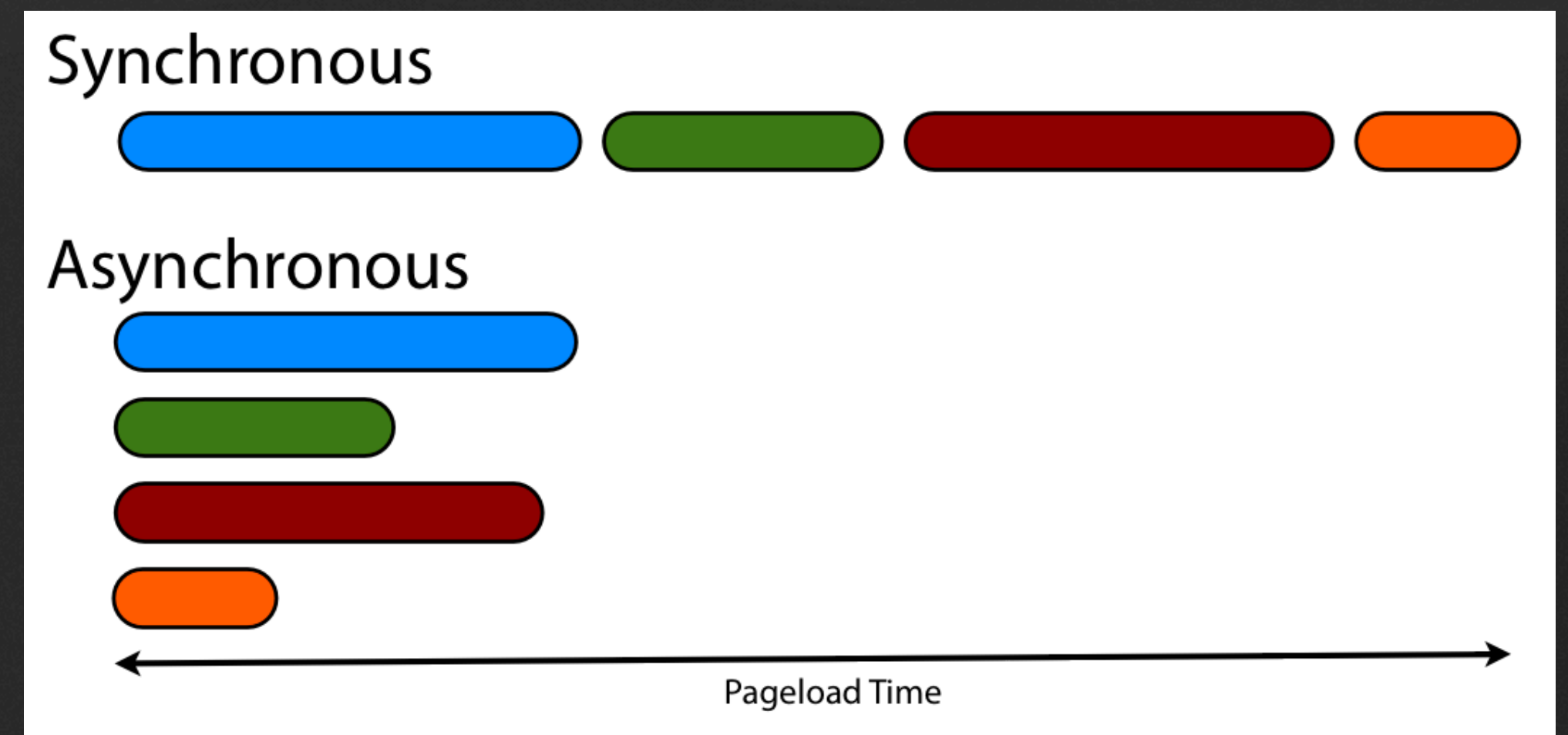
# async programming i nodejs

Nikolaj Schlüter Nielsen – CBS 2022



# Hvad er asynkrone tasks?

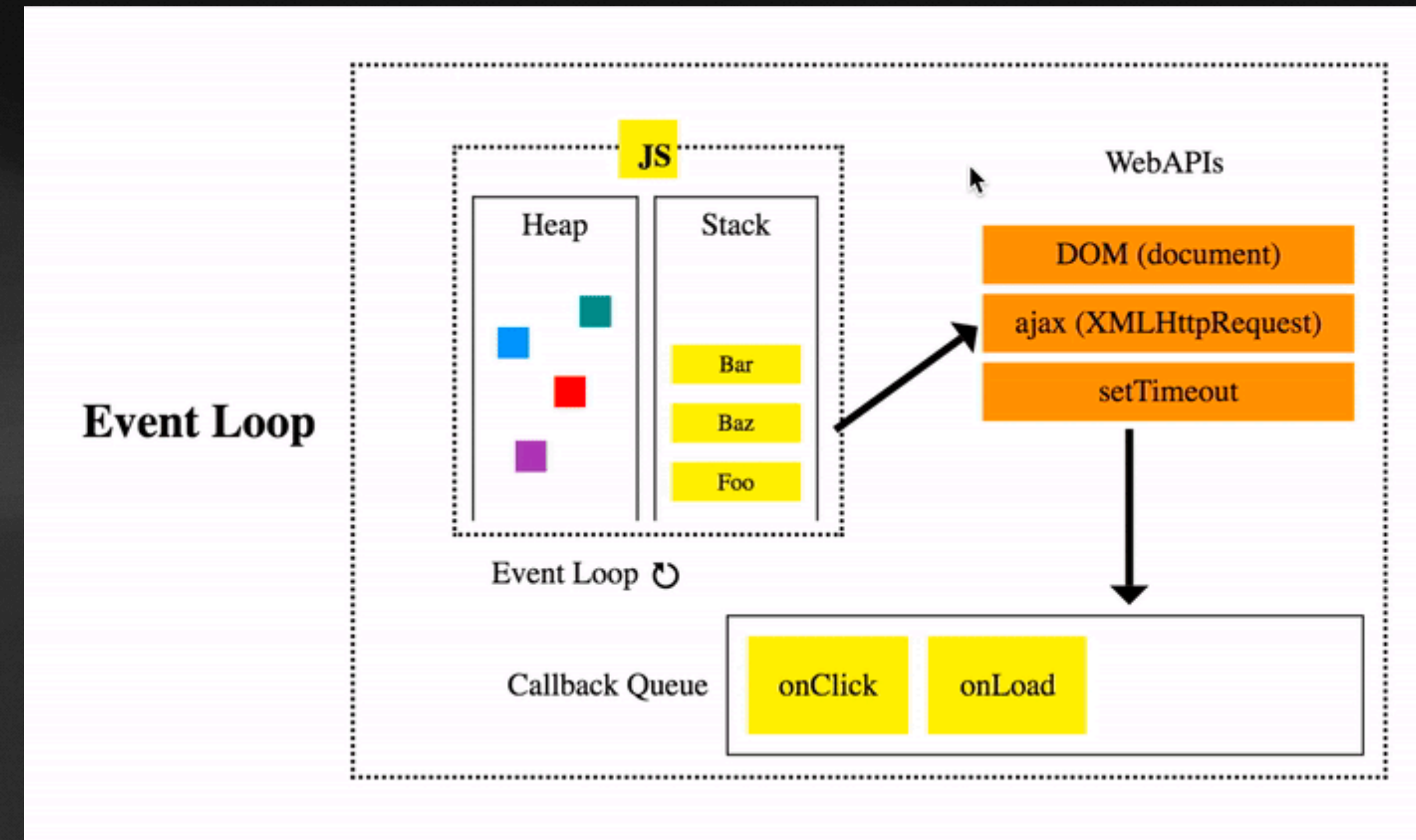
- Node.js / JavaScript er ikke multi-threaded
  - Bruger et event loop der kan *tilsidesætte* 'opgaver' til senere
  - Hvilke 'opgaver'?
    - API request
    - Andet?
- Hvorfor vil vi gerne kører ting async?





# JavaScript event loop


- **Heap** = Memory
- **Call stack**
  - Hvor kode bliver kørt linje for linje
  - Asynkrone opgaver bliver sent videre til:
- **Callback Queue**
  - Hvor async opgaver venter
- **Så hvad gør event loopet?**





# Promises

- En måde at håndtere asynkron kode
- .then()
- .catch()
- .finally()

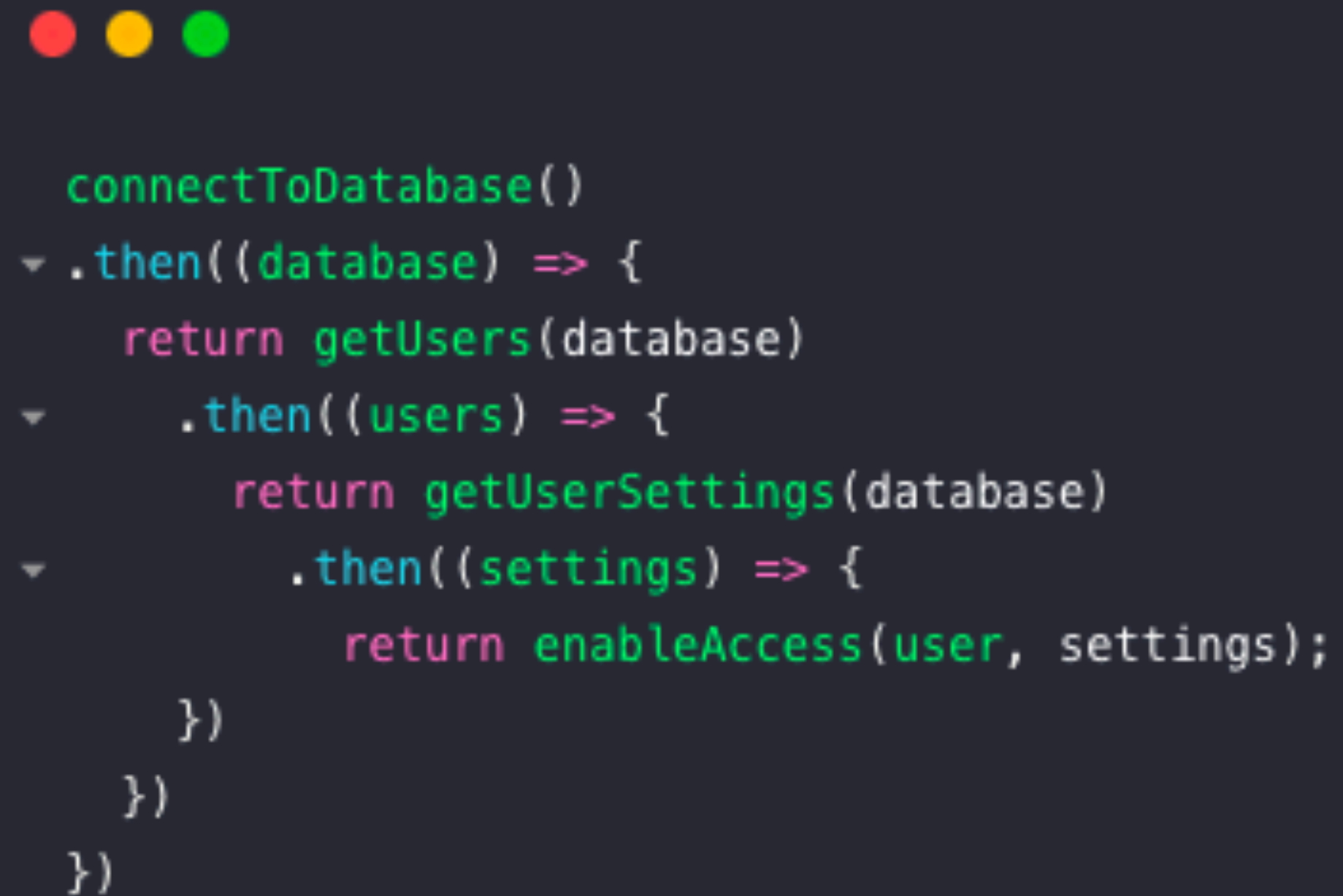


```
// readFile er asynkron og returnere et derfor Promise

readFile()
  .then((file) => console.log('Her er filen:', file))
  .catch((error) => console.log('Kunne ikke åbne filen: ', error))
  .finally(() => console.log('Slut med at læse fil'))
```



# .then hell



```
connectToDatabase()  
  .then((database) => {  
    return getUsers(database)  
      .then((users) => {  
        return getUserSettings(database)  
          .then((settings) => {  
            return enableAccess(user, settings);  
          })  
        })  
      })  
    })  
  })
```



# async / await

- Det samme som Promise- men pænere!

```
try {
  const file = await readFile()
  console.log('Her er filen:', file)
} catch (error){
  console.log('Kunne ikke åbne filen: ', error)
} finally {
  console.log('Slut med at læse fil')
}
```

```
async function getUsersFriends () {
  try {
    const user = await getUser()
    const userFriends = await getFriends(user.friends)
  }
  catch (err) {
    console.error('kunne ikke hente dine venner :(', err)
  }
}
```



# Opgaver om vejret



# Dagens opgaver

## Kald af vejr API'er med

- Hent koden på canvas Eller <https://github.com/slytter/cbs-async-lecture>
- Åben i jeres editor (vs code)
- Installer:
  - `npm install`
- Kør
  - `node opgave1`



# Opgave 1



# Opgave 1

Hent assignment koden fra canvas

- Kald vejr API'en 'open-meteo' med metoden **fetchRoute()**
  - Vælg enten CPH, Berlin, Paris, LA eller find jeres egen på [open-meteo.com](https://open-meteo.com)
- **fetchRoute** returner et Promise med et resultat fra **open-meteo**
  - **console.log** den nuværende temperatur i det valgte land



# Opgave 2



# Opgave 2

## Håndter fejl med catch

- Brug **fetchRoute()** til at kalde 'wrongRoute'
- Brug **.catch** til at fange og håndtere fejlen



# Opgave 3



# Opgave 3

## `async / await`

- Brug `async / await` som erstatning af `.then`
- Kald, og `console.log` **alle** vejr endpoints **en af gangen**



# Promise.all

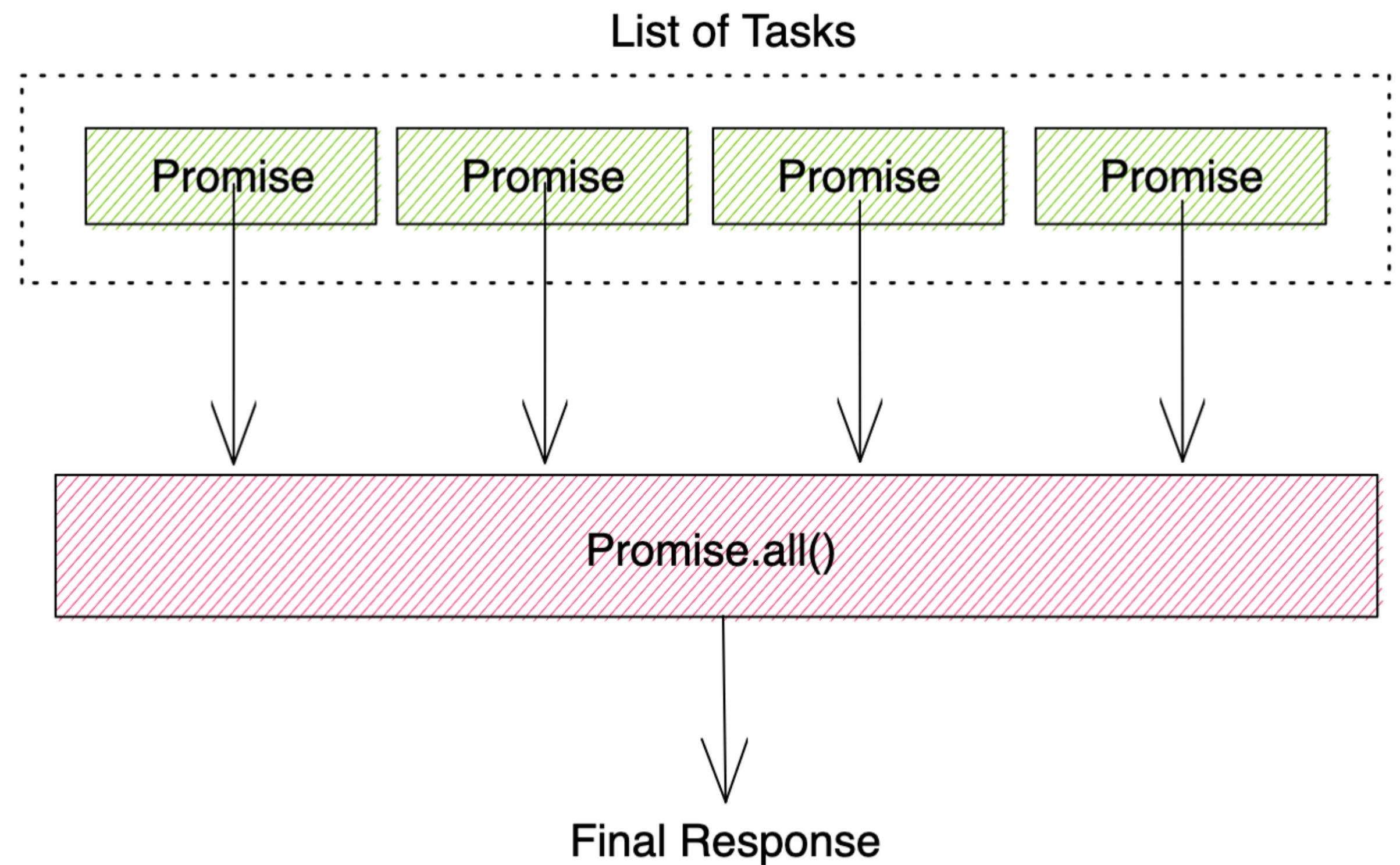
- Kør flere Promises parallelt



```
const promise1 = Promise.resolve(3) // success
const promise2 = 42 // success
const promise3 = new Promise(resolve => setTimeout(resolve, 100, 'foo')) // success

const allPromises = [promise1, promise2, promise3]

Promise.all(allPromises).then(values => console.log(values))
// Output: [3, 42, 'foo']
```





# Opgave 4



# Opgave 4

## Promise.all

- Brug **Promise.all** til at kalde **alle** vejr routes på samme tid
- Returner resultaterne i **det samme console.log()**



Tak for i dag!