Förra veckans ämnen...

Introduktion till Node.js
Vad är Node.js? Översikt över ekosystemet. Varför använda det?

Installera och konfigurera Node.js
Hur man sätter upp en server för att köra JavaScript. Fokusera på olika operativsystem, inklusive skillnader och konsekvenser.

Grundläggande API-kommunikation
RESTful API, HTTP-metoder (GET, POST, PUT, DELETE). Bygga enkla API med Express.js.

Denna vecka tar upp...

Installera och konfigurera Node.js
Hur man sätter upp en server för att köra JavaScript. ~Fokusera på olika operativsystem, inklusive skillnader och konsekvenser.~

Grundläggande API-kommunikation
RESTful API, HTTP-metoder (GET, POST, PUT, DELETE). **Bygga enkla API med Express.js.**

Repetition

API:er – Vad är ett API?

- API står för **Application Programming Interface**.
- Det är ett sätt för program att kommunicera med varandra via definierade gränssnitt.
- Exempel på API-typer:
  - **Webb-API**: API:er som kommunicerar över webben.
  - **Biblioteks-API**: Kodpaket och funktioner som kan användas internt.

Vad är ett protokoll?

- Ett **protokoll** är en uppsättning regler och standarder som gör att olika enheter kan kommunicera med varandra.
- Exempel på protokoll:
  - **HTTP** (Hypertext Transfer Protocol) – Används för att överföra data över webben.
  - **TCP/IP** – Grunden för internetkommunikation.
  - **FTP** (File Transfer Protocol) – Används för filöverföring.

HTTP – Hypertext Transfer Protocol

- HTTP är ett **stateless** protokoll, vilket innebär att varje förfrågan och svar är oberoende av varandra.
- Används för att överföra webbsidor, API-data, bilder, etc.

HTTP-meddelanden

Kommunikation över HTTP-meddelande innebär alltid en **Request** (förfrågan) samt en **Response** (svar).

HTTP Request (Förfrågan)
Innehåller metod (t.ex. GET, POST), URL, headers + ev. body.

HTTP Response (Svar)
Innehåller statuskod (t.ex. 200 OK, 404 Not Found), headers och body om det finns data att skicka tillbaka.

- Detta gör HTTP **stateless** – varje förfrågan hanteras isolerat, utan kontext från tidigare förfrågningar.

HTTP Request & Response Flow

1. **Klient** skickar en request:
   - T.ex. GET `/index.html` HTTP/1.1
2. **Server** svarar:
   - HTTP/1.1 200 OK
   - Body: `<html>...</html>`

Varje interaktion sker som en fråga och ett svar, alltid.

HTTP-metoder

- **GET** – Hämta resurser
- **POST** – Skapa nya resurser
- **PUT/ PATCH** – Uppdatera resurser
- **DELETE** – Ta bort resurser

HTTP-Statuskoder

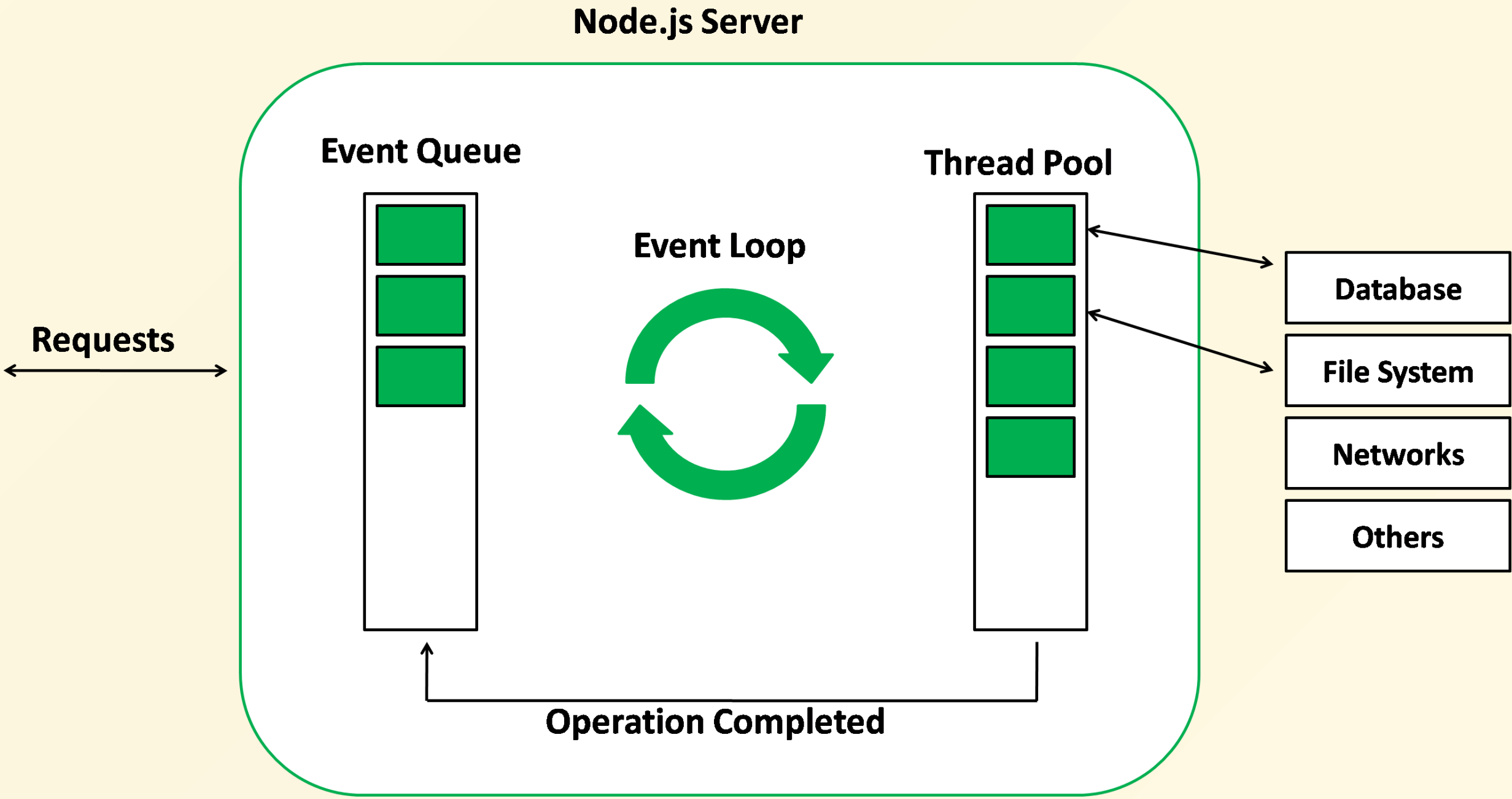
- **200 OK** – Förfrågan lyckades
- **404 Not Found** – Resurs hittades inte
- **500 Internal Server Error** – Något gick fel på servern

Node.js

- Node.js är en **JavaScript-runtime** byggd på **Chrome's V8 JavaScript-motor**.
- Låter dig köra JavaScript på serversidan.
- **Event-driven**, asynkron, non-blocking I/O-modell gör den perfekt för skalbara applikationer.

Event Loop i Node.js

- Node.js är **single-threaded**, men hanterar asynkrona operationer via en **event loop**.
- Exekverar kod, samlar och bearbetar händelser, samt kör asynkrona funktioner.


Node.js Server
Event Queue
Thread Pool
Event Loop
Requests
Database
File System
Networks
Others
Operation Completed

JavaScript på Serversidan

Fördelar med JavaScript på serversidan:

- **Samma språk** på både klient- och serversidan.
- Snabbt tack vare konkurrans mellan webbläsare.
- **Asynkron programmering** ger bättre prestanda för I/O-tunga uppgifter.
- Stort ekosystem av bibliotek och moduler via **NPM** (Node Package Manager).

Vad är en teknikstack?

- En **teknikstack** är en kombination av verktyg, språk och teknologier som används för att bygga en applikation.
- Exempel:
  - **Frontend**: React, Redux, MUI etc
  - **Backend**: Node.js, Express, Databaser (MySQL, MongoDB).

  Kan vara helt annat...
  - **DevOps**: Docker, Kubernetes, Jenkins.

CRUD

- CRUD står för:
  - **Create** – Skapa ny data.
  - **Read** – Läsa data.
  - **Update** – Uppdatera existerande data.
  - **Delete** – Ta bort data.
- Dessa operationer är grunden för hur data hanteras i databaser och REST-API:er.

REST – Representational State Transfer

- REST är en arkitekturstil som bygger på **stateless** interaktioner mellan klient och server.
- Använder HTTP-metoder för CRUD-operationer:
  - **GET** – Hämta resurser
  - **POST** – Skapa nya resurser
  - **PUT/PATCH** – Uppdatera resurser
  - **DELETE** – Ta bort resurser
- Skalbart, flexibelt och enkelt att förstå.

REST API Design

Viktiga principer:

Stateless & Idempotent: Ingen server-session. All information kommer med varje begäran.

Bygger vidare på HTTP: Varje resurs representeras som en HTTP URL, Samma **statuskoder**, samma **verb**

Vecka 2

Programmerinsmönster och design för REST
Data entiter, intefaces och designmönster

Programmering med Node.js
Använda typescript, mer om moduler, använda portar och express.js

Pakethantering med NPM
Hur man installerar och hanterar paket. Viktigt att tänka på vid pakethantering.

Praktiskt API-kommunikation
Kommunikation från ett front-end projekt till ett API - uppdelning av ansvar och felhantering