TDP013 – Web Programming and Interactivity

Lecture 4: Client frameworks, templates and WebSockets, Project

Huanyu Li

Human-Centered Systems, Department of Computer and Information Science

LINKÖPING UNIVERSITY

Recap from lectures 1, 2 and 3

- JavaScript
  - Repetition of callbacks and promises
- Testing with Mocha and code coverage with Istanbul
- MongoDB
  - Database where data is stored as JSON objects
  - No schema
  - Scales well to large data volumes
- CORS
- AJAX
- Ethics assignment

Template engines and client frameworks

LINKÖPING UNIVERSITY

Backend and Client

- Backend
  - Node.js and Express
  - Handles communication with database
  - Provides an API for interaction
  - Runs on server and cannot be directly seen by user
- Client
  - Runs in browser (hosted via HTTP server)
  - HTML5, JavaScript, CSS
  - Often uses a framework to build the website
  - Communicates with backend
  - Runs on client and (almost) everything can be seen by user

Express for both backend and frontend

- Static web pages in Express (i.e. act as an HTTP server)
- Example:
  - app.use(express.static('public'))
  - The content will then be available under:
  - http://<my-domain>/index.html

```
// another middleware used to serve static files (e.g., css, images and scripts)
app.use(express.static('public', {
  maxAge: 5000 // how long should express cache each document?
}));
```

```
(ontomatch) huali50@mac01483 exress-example % ls -l
total 112
-rw-rw-r--@   1 huali50  staff    464 Oct  4  2023 README.md
-rw-rw-r--@   1 huali50  staff   1697 Sep  1 20:22 app.js
drwxr-xr-x  101 huali50  staff   3232 Sep  1 19:55 node_modules
-rw-rw-r--@   1 huali50  staff  41684 Sep  1 19:55 package-lock.json
-rw-rw-r--@   1 huali50  staff    338 Sep  1 19:55 package.json
drwxrwxr-x@   5 huali50  staff    160 Sep 14 13:10 public
(ontomatch) huali50@mac01483 exress-example % cd public
(ontomatch) huali50@mac01483 public % ls -l
total 8
drwxrwxr-x@ 3 huali50  staff   96 Oct  4  2023 css
-rw-rw-r--@ 1 huali50  staff  397 Oct  4  2023 index.html
drwxrwxr-x@ 3 huali50  staff   96 Oct  4  2023 js
```

Express for both backend and frontend

- Another option is to send the file as a response
  - res.sendFile(__dirname + '/index.html')
- If you use a client framework, you usually run the client and backend on different ports

```
// serve your index.html explicitly
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});
```

Template engines

Templates for generating pages

- Static template files where variables are replaced with actual values
  - … but more complicated functions are also supported!
- Some popular alternatives in Express:
  - Pug, default in Express
  - EJS
  - Handlebars
  - See more at https://expressjs.com/en/guide/using-template-engines.html

Templates for generating pages

- To use templates, you need to configure your Express server:
- Specify where the template files will be located
- Select template engine
- Install template engine via npm
- Render for specific routes

Pug

- A template engine for Node.js and Express
- Users do not have to write HTML code manually

- https://pugjs.org/api/getting-started.html

Example: Pug

```
app.set('views', './views')
app.set('view engine', 'pug')
app.get('/', (req, res) => {
  let data = { title: 'Hello', message: 'World!'}
  res.render('index', data)
})
```

```
html
  head
    title= title
  body
    h1= message
```

views/index.pug

Code example

Client framework

Templates for generating pages

- To use templates, you need to configure your Express server:
- Specify where the template files will be located
- Select template engine
- Install template engine via npm
- Render for specific routes

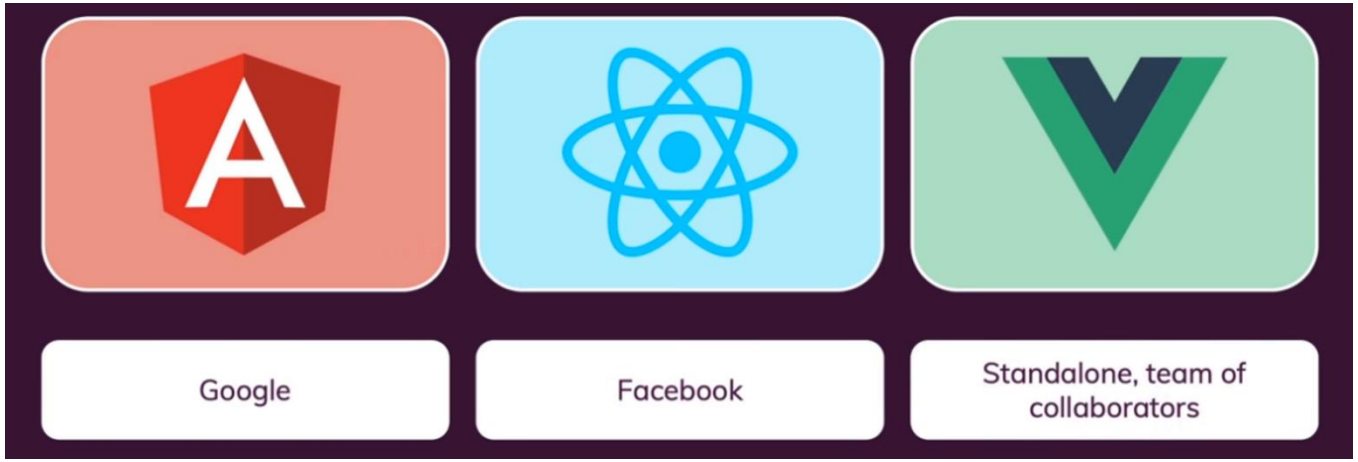
A simple plain HTML code

```
<!DOCTYPE html>
<html>
    <head>
        <title>Amazing Scientists</title>
    </head>
    <body>
        <section>
        <h1>Amazing scientists</h1>
            <img src="https://i.imgur.com/MK3eW3As.jpg" alt="Katherine Johnson">
            <img src="https://i.imgur.com/MK3eW3As.jpg" alt="Katherine Johnson">
            <img src="https://i.imgur.com/MK3eW3As.jpg" alt="Katherine Johnson">
        </section>
    </body>
</html>
```

React, Vue, Angular

- Data binding
- Single page application

- Different syntactic sugars



React

- Free and open-source front-end JavaScript library
- Component-based
  - A component is a JavaScript function combined with markup
  - e.g., button, webpage
- Immutable states

- https://react.dev

React 18.3.1

- react DOM
  - create a root to display React components inside a browser DOM node
- JSX:
  - HTML-like syntax inside JavaScript

```
function Profile() {
  return (
    <img
    src="https://i.imgur.com/MK3eW3As.jpg"
    alt="Katherine Johnson"
    />
  );
}
```

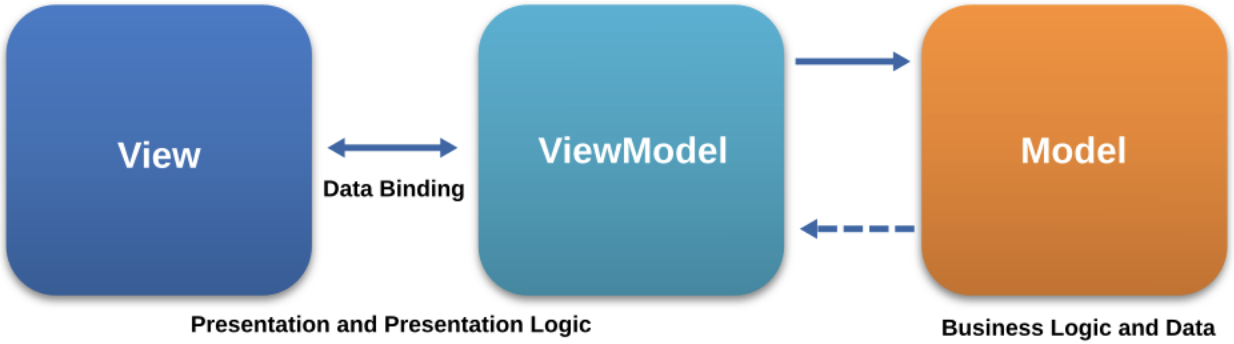
```
export default function Gallery() {
  return (
    <section>
      <h1>Amazing scientists</h1>
        <Profile />
        <Profile />
        <Profile />
    </section>
);
}
```

```
// Render App into the root div
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

Code Example

Vue

- Open-source and Model-View-ViewModel
- Vue components (reusable UI blocks)
- Mutable state
- https://vuejs.org



https://en.wikipedia.org/wiki/Model–view–viewmodel

Vue

- index.html
- main.js
- app.vue
- component.vue
- ...

Code Example

Angular and AngularJS

- https://angular.dev
- AngularJS
  - JavaScript-based web framework for developing web applications
- Angular
  - TypeScript-based free and open-source web application framework
  - A complete rewrite from the same team building AngularJS
- Immutable state

Angular and AngularJS

- Use components with templates
  - component.ts
  - component.html
  - app.html
  - ...

```
import { Component } from '@angular/core';


@Component({
selector: 'app-profile',
imports: [],
templateUrl: './profile.html',
styleUrl: './profile.css'
})
export class ProfileComponent {}
```

```
<p>profile works!</p>
<img
src="https://i.imgur.com/MK3eW3As.j
pg"
alt="Katherine Johnson"
/>
```

```
<div>
<h1>Amazing scientists</h1>
<app-profile></app-profile>
<app-profile></app-profile>
<app-profile></app-profile>
</div>
```

Code Example

WebSockets

LINKÖPING UNIVERSITY

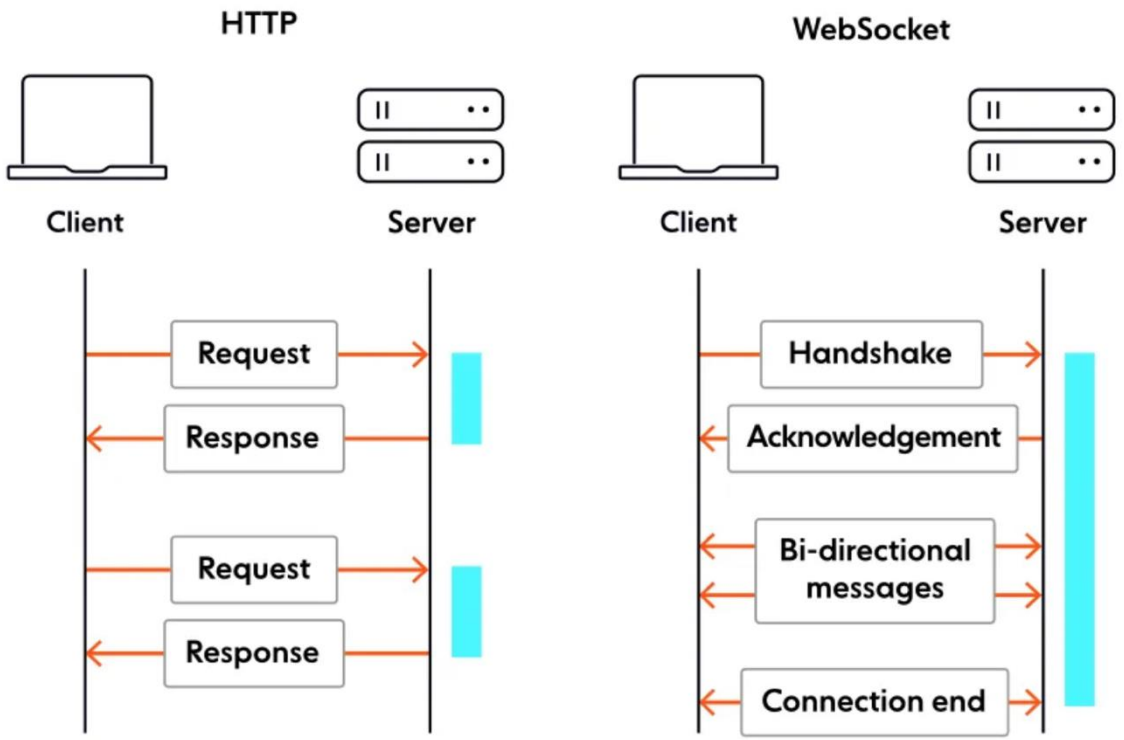
WebSocket

- A computer communications protocol
  - provides a bidirectional communication channel over a single TCP connection
- A living standard
- https://websockets.spec.whatwg.org

HTTP communication vs. WebSocket

- unidirectional protocol
  - port:80
  - multiple connections for multiple data requests
- bidirectional communication protocol
  - port:443
  - real-time data exchange

HTTP communication vs. WebSocket



https://websocket.org/guides/road-to-websockets/

WebSocket API

- Enable bidirectional communication between a client and a server
  - Socket.IO
  - SocketCluster
  - ...

Socket.IO server side

```
import express from 'express';
import http from 'http';
import { Server } from "socket.io";

const app = express();
const server = http.createServer(app);
const io = new Server(server);
app.use(express.static('client'))
app.get("/test", (req, res) => {
        res.send("Express is still working as expected!")
})
// on connect
io.on('connection', (socket) => {
        console.log('A user connected');
}
server.listen(3000, () => console.log('Server is running'));
```

Socket.IO Client side

```
socket.on('message', (msg) => {
        console.log(msg)
});
```

Code Example

Project

LINKÖPING UNIVERSITY

Project: a social website

- The project is carried out in pairs (or individually) according to webreg
  - *https://www.ida.liu.se/webreg3/TDP013-2025-1/PRA1*
- For a pass, the basic requirements must be met
- For higher grades, there are additional requirements
- Deadlines:
  - Tuesday, October 14th (submission of code)
  - Wednesday, October 15th 1–5 pm (project presentations)
  - Friday, October 17th 8-10am (project presentations)
    - you will be assigned a group for project presentations
  - See the website for other reporting options and supplements

Project

- A social website (a simplified version of Facebook)
- Users should be able to
  - register with a username and password,
  - log in and log out on a personal page
- Passwords should not be sent/saved in plain text
- Friends should only be added after a friend request has been accepted
- A friend relationship applies in both directions
- Users should only be able to post on their own and their friends' pages
- Data should be saved in a database (MongoDB)
- Backend testing should be done with Mocha/Istanbul

Project

- The website should be clearly structured and use approximate styles (e.g., colors, element sizes, well-chosen spacing and margins)
- To demonstrate you understanding of CSS!

The Project: Higher Grade

- 1. Allowing friends to chat with each other in real time using HTML5 WebSockets and socket.io plugin for Node.js
- 2. Using a client framework to build your application
  - React is recommended but other frameworks are OK to use after discussing with your lab assistant

- Requirements for Grade 4: basic Requirements + 1 or 2
- Requirements for Grade 5: basic Requirements + 1 and 2

LINKÖPING UNIVERSITY