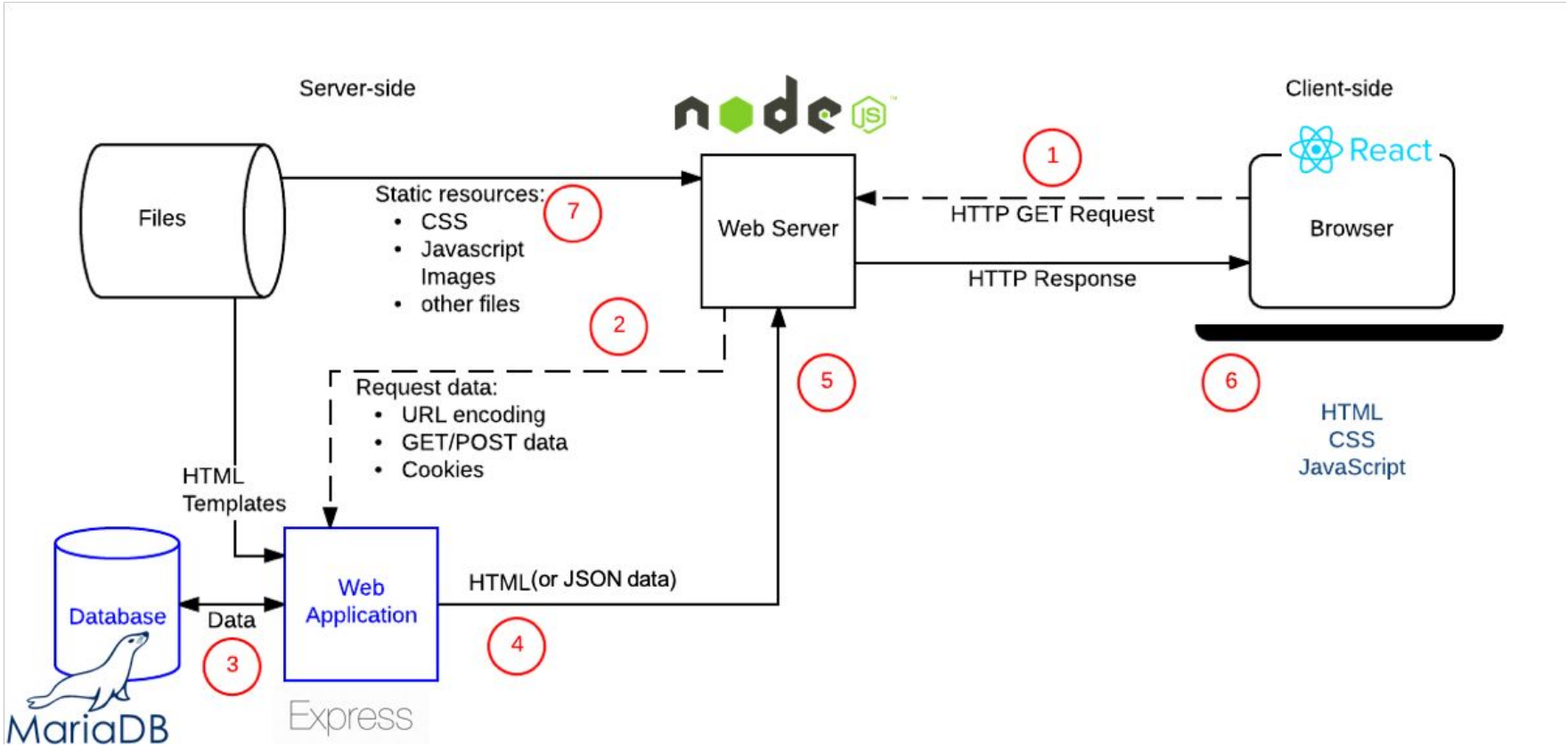
CIS 450/550

Recitation 3

Architecture Overview



Server

- index.js
  - Handles security permissions
    - Client & Server communication
  - Exposes routes to permitted origins
  - Binds a route handler to a route
  - Launches the express server

```
1   const bodyParser = require('body-parser');
2   const express = require('express');
3   var routes = require("./routes.js");
4   const cors = require('cors');
5
6   const app = express();
7
8   app.use(cors({credentials: true, origin: 'http://localhost:3000'}));
9   app.use(bodyParser.json());
10  app.use(bodyParser.urlencoded({extended: false}));
11
12  /* ------------------------------------------------------------ */
13  /* ------------------- Route handler registration ------------- */
14  /* ------------------------------------------------------------ */
15
16  /* ---- (Dashboard) ---- */
17  // The route localhost:8081/people is registered to the function
18  // routes.getAllPeople, specified in routes.js.
19  app.get('/people', routes.getAllPeople);
20
21  app.listen(8081, () => {
22      console.log(`Server listening on PORT 8081`);
23  });
```

Server

- db-config.js
  - Contains database credentials
- routes.js
  - Establishes connection to the database
  - Contains route handlers
    - Can query a database
    - Returns data in a JSON response
    - Handles errors

```
1   var config = require('./db-config.js');
2   var mysql = require('mysql');
3
4   config.connectionLimit = 10;
5   var connection = mysql.createPool(config);
6
7   /* ------------------------------------------------ */
8   /* ------------------- Route Handlers ------------- */
9   /* ------------------------------------------------ */
10
11  /* ---- (Dashboard) ---- */
12  function getAllPeople(req, res) {
13    var query = `
14      SELECT login, name, birthyear
15      FROM Person;
16    `;
17    connection.query(query, function(err, rows, fields) {
18      if (err) console.log(err);
19      else {
20        res.json(rows);
21      }
22    });
23  };
```

Client - Navigation

- App.js
  - Contains routing for the frontend on localhost

```
24                  <Route
25                      exact
26                      path="/dashboard"
27                      render={() => (
28                          <Dashboard />
29                      )}
30                  />
```

- PageNavbar.js
  - Associates routing with a GUI navbar

Client - React Components

- constructor()
  - `props` - can pass down values from a parent component
  - State initialization - set the default value when the page loads
  - Function bindings - sets the context of your function to `this`

```
6   export default class FindFriends extends React.Component {
7       constructor(props) {
8           super(props);
9
10          // State maintained by this React component is the inputted login,
11          // and the list of friends of that login.
12          this.state = {
13              login: "",
14              foundFriends: []
15          }
16
17          this.handleLoginChange = this.handleLoginChange.bind(this);
18          this.submitLogin = this.submitLogin.bind(this);
19      }
```

- Parent/Child Components
  - Create a component and use it like an HTML tag in another component
  - Pass props like HTML attributes
- this.setState()
  - Used to update any state variable within your component
  - NO: this.state.login = "mylogin";
- render() - returns the HTML to be displayed

Client - React Components

- componentDidMount()
  - Built-in React function that is called when a page loads
  - Used in this app to make an HTTP GET request to load data from our database
  - Fetch is an *asynchronous* function that returns a *promise*
    - Handle async functions by chaining .then() to create a guaranteed execution order
  - Map is used as an iterator
    - Used here to create an HTML block for all rows of data

```
17  // React function that is called when the page loads.
18  componentDidMount() {
19    // Send an HTTP request to the server.
20    fetch("http://localhost:8081/people",
21    {
22      method: 'GET' // The type of HTTP request.
23    }).then(res => {
24      // Convert the response data to a JSON.
25      return res.json();
26    }, err => {
27      // Print the error if there is one.
28      console.log(err);
29    }).then(peopleList => {
30
31      // Map each attribute of a person in
32      // this.state.people to an HTML element
33      let peopleDivs = peopleList.map((person, i) =>
34      <div key={i} className="person">
35        <div className="login">{person.login}</div>
36        <div className="name">{person.name}</div>
37        <div className="birthyear">{person.birthyear}</div>
38      </div>);
39
40      // Set the state of the person list to the value
41      // returned by the HTTP response from the server.
42      this.setState({
43        people: peopleDivs
44      });
45    }, err => {
46      // Print the error if there is one.
47      console.log(err);
48    });
49  }
```