Intermediate React

tsevdos.me / @tsevdos

Agenda

All the content can be found here: https://github.com/codehub-learn/react-bootcamp.

- Bootstrap a React app (create-react-app)
- HTTP requests and API
- Asynchronous JavaScript
- Forms and events
- useEffect hook
- UI and Charts libraries

Rules

Feel free to interrupt me for:

- questions
- relevant comments

Create react app

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

Create react app

```
npx create-react-app my-app
cd my-app
npm i antd react-router-dom recharts
npm start
```

Create react app

demo

UI library

- Ant Design
- reactstrap
- Material UI

React Router (Web)

- React Router (Web)

Recharts

- Recharts

Create react app

- files and directory structure
- create/import components
- import ant design components
- client-side router with react router

HTTP

- protocol
- methods
- status codes
- API
- JSON

HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

HTTP

- HTTP is an "application layer" protocol
- HTTP follows a classical "client-server" model
- HTTP is a TCP/IP based communication protocol
- The standard port for HTTP connections is port 80
- The version of HTTP in common use is HTTP/1.1 or HTTP/2

HTTP

HTTP is used to transmit resources. A resource is some chunk of information that can be identified by a URL (it's the R in URL). The most common kind of resource is a file, but a resource may also be a dynamically-generated query result.

HTTP Methods

- GET
- POST
- PUT
- DELETE
- PATCH
- HEAD

REST (Representational state transfer) API

- method
- endpoint
- headers
- data (or body)

CRUD in REST

- Create - POST `/books`
- Read - GET `/books` or `/books/12`
- Update - PUT `/books/12`
- Delete - DELETE `/books/12`

Status codes

- 1xx - Informational
- 2xx - Success (200)
- 3xx - Redirection (301)
- 4xx - Client error (404)
- 5xx - Server error (500)

JSON

- JSON (JavaScript Object Notation) is a lightweight data-interchange format
- It is easy for humans to read and write
- It is easy for machines to parse and generate
- It is completely language independent

JSON allowed values

- string
- number
- object
- array
- boolean
- null

An example

```
{
  "firstname": "John",
  "lastname": "Tsevdos",
  "age": 38,
  "address": {
    "city": "Athens",
    "street": "My street",
    "number": 12
  },
  "isOlympiakos": true,
  "pet": null,
  "hobbies": [
    "football",
    "movies",
    "coding"
```

Convert an object to JSON (JSON.stringify)

```
const me = {
  firstname: 'John',
  lastname: 'Tsevdos',
  age: 31,
  address: {
    city: 'Athens',
    street: 'my street',
    number: 12
  },
  isOlympiakos: true,
  pet: null,
  hobbies: [
    "football", "movies", "coding"
  ]
};
```

Convert a JSON to object (JSON.parse)

```
const myJSON = '{"firstname":"John", "lastname":"Tsevdos", "ag
const myObj = JSON.parse(myJSON);

console.log(myJSON);
console.log(typeof myJSON);
console.log(myObj);
console.log(typeof myObj);
```

Asynchronous JavaScript

AJAX basics

- Asynchronous JavaScript and XML
- Set of web technologies
- Send / receive data asynchronously

AJAX technologies

- HTML / CSS
- JavaSrcipt / DOM
- XMLHttpRequest object
- XML / JSON

Asynchronous JavaScript

- Promises
- Fetch API
- JSON

Asynchronous JavaScript

```
setTimeout(() => {
  console.log("Hello from line 6");
}, 2000);

console.log("Hello from line 9");
```

Promises

```
const myPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    if (true) {
      resolve("Happy path!");
    } else {
      reject(Error("Something went wrong."));
    }
  }, 2000);
});

myPromise
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
```

Promises

```
const myPromise1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    if (true) {
      resolve("Happy path!");
    } else {
      reject(Error("Something went wrong."));
    }
  }, 2000);
});

const myPromise2 = new Promise((resolve, _reject) => {
  setTimeout(() => {
    resolve("My promise2 resolved!!!");
  }, 4000);
});
```

Fetch API

```
fetch("http://api.icndb.com/jokes/random")
  .then((res) => res.json())
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });
```

Async/await

```
async function getJoke() {
  try {
    const res = await fetch("http://api.icndb.com/jokes/random
    const data = await res.json();
    console.log(data);
  } catch (error) {
    console.log(error);
  }
}

getJoke();
```

Forms and events

- inputs events
- form events
- ant design components

Forms and events

examples

React useEffect hook

accepts an effect "action" as an anonymous function as the first argument. Skip applying an effect if certain values haven’t changed between re-renders. To do so, pass an array as an optional second argument to useEffect. Finally, some effects might require cleanup so they return a function.

useEffect hook

examples

Events and hooks

Events run code when the user / browser interacts with the page, useEffect hook runs code depending the component's rendering status.

Charts library

Recharts

Recharts library

examples

Recap:

- Bootstrap a React app (create-react-app)
- HTTP requests and API
- Asynchronous JavaScript
- Forms and events
- useEffect hook
- UI and Charts libraries

That's all folks

Questions / Discussions?