Tech App Specification

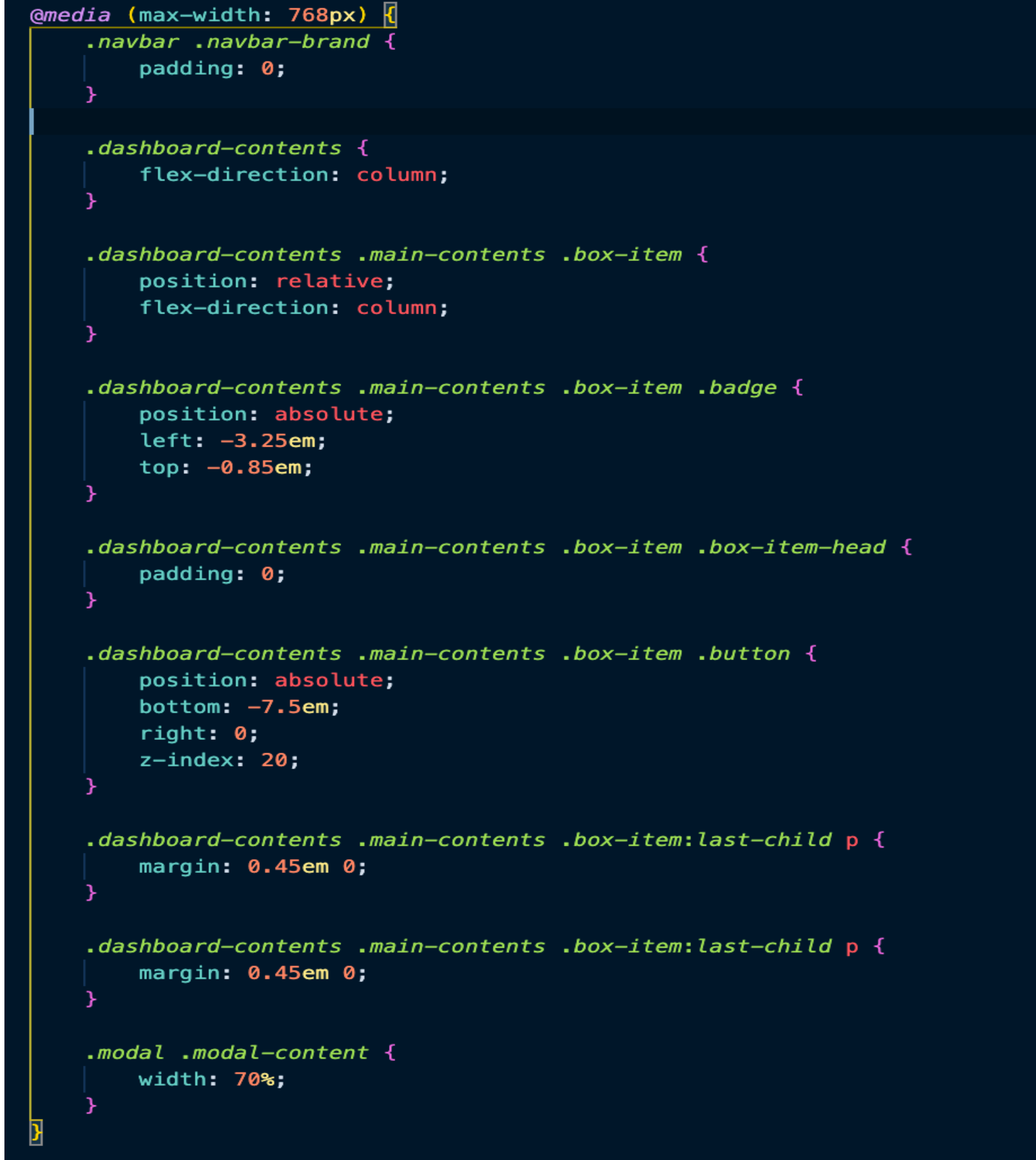
By, Mazahar Shaikh

Improvements Areas

Styles

Styles | Media Queries

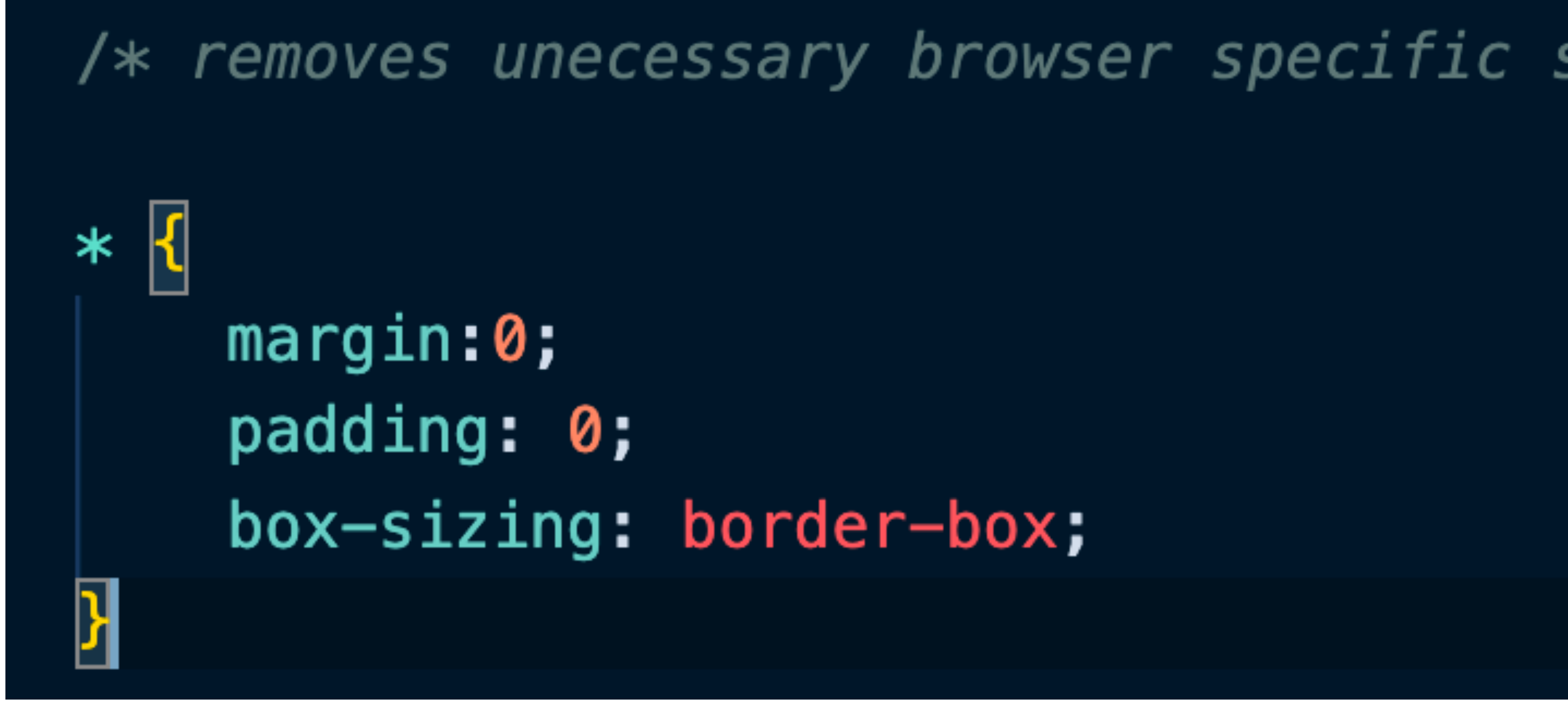
- Update media query, instead of different break points, we can play with specific width for anything < 768 px (iPad portrait) – It will support mobile as well
- Avoid render blocking CSS, not necessary every time
- *<link href="mobile.css" rel="stylesheet" media="(max-width: 768px)">*
- This would improve critical rendering path (optimization technique for critical path length)



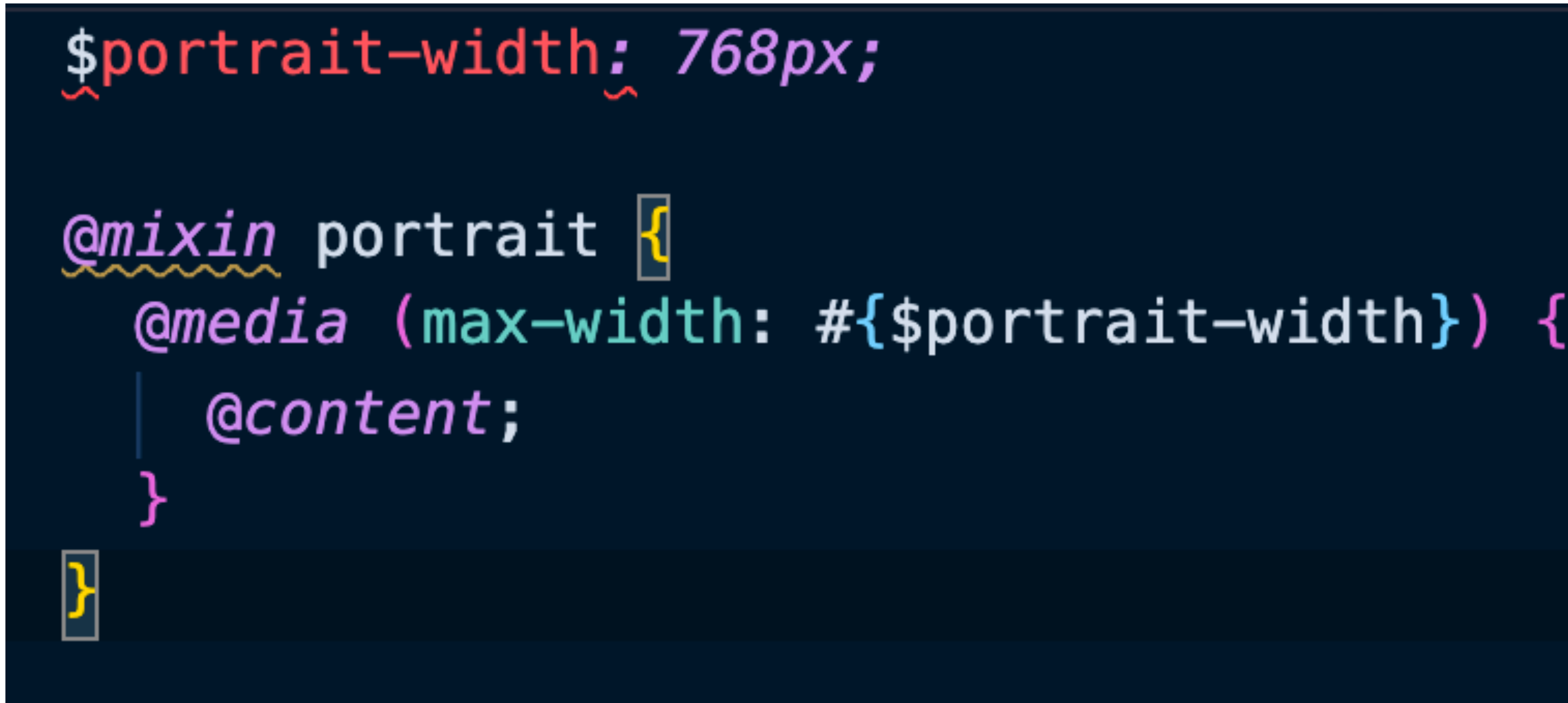
```css
@media (max-width: 768px) {
    .navbar .navbar-brand {
        padding: 0;
    }

    .dashboard-contents {
        flex-direction: column;
    }

    .dashboard-contents .main-contents .box-item {
        position: relative;
        flex-direction: column;
    }

    .dashboard-contents .main-contents .box-item .badge {
        position: absolute;
        left: -3.25em;
        top: -0.85em;
    }

    .dashboard-contents .main-contents .box-item .box-item-head {
        padding: 0;
    }

    .dashboard-contents .main-contents .box-item .button {
        position: absolute;
        bottom: -7.5em;
        right: 0;
        z-index: 20;
    }

    .dashboard-contents .main-contents .box-item:last-child p {
        margin: 0.45em 0;
    }

    .dashboard-contents .main-contents .box-item:last-child p {
        margin: 0.45em 0;
    }

    .modal .modal-content {
        width: 70%;
    }
}
```

Styles | Contin…

- Will add reset CSS
- Instead of verbose media-query we can use SASS mixins



```
/* removes unecessary browser specific

* {
    margin:0;
    padding: 0;
    box-sizing: border-box;
}
```



```
$portrait-width: 768px;

@mixin portrait {
  @media (max-width: #{$portrait-width}) {
    @content;
  }
}
```

HTML / Assets

HTML/Assets

- Accessibility metric: Instead of anchor will use Button
- Preload: Use preload for font awesome
- Usage of Semantic markups

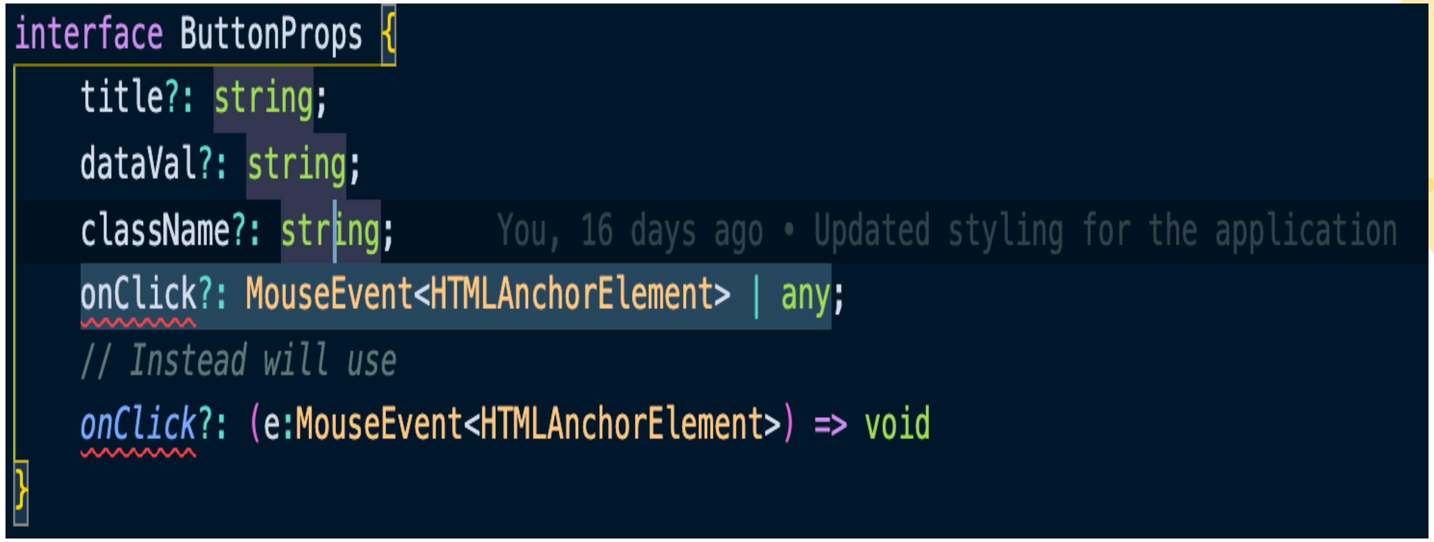
Image optimization

- Usage of SVG for logos and other icons, shapes. Instead of PNG/JPEG.
- It will take care of problems associated with Raster images and will not break on aspect ratios for different screen resolutions

TypeScript

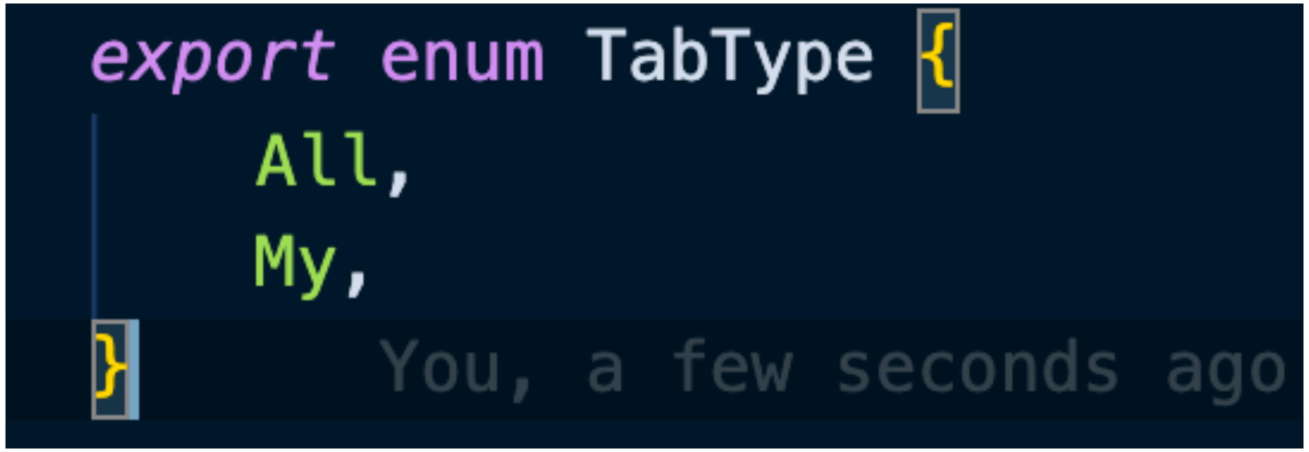
TypeScript

- Usage of specific type / union type / instead of any
- Usage of Function type:



```
interface ButtonProps {
    title?: string;
    dataVal?: string;
    className?: string;      You, 16 days ago • Updated styling for the application
    onClick?: MouseEvent<HTMLAnchorElement> | any;
    // Instead will use
    onClick?: (e:MouseEvent<HTMLAnchorElement>) => void
}
```

- Usage of enum for tabs specific value check
- Instead of explicit check "any"/"my" will use enum



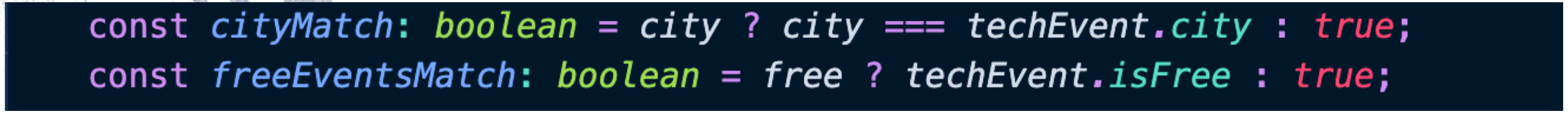
```
export enum TabType {
    All,
    My,
}       You, a few seconds ago
```

TypeScript practices

- Write utility function for error handling

```
function generateError(msg: string, code:
  number): never {
    throw {message: msg, errorCode: code
  })
}
// Usage:
generateError("and error occurred", 500);
```

- For specific values, will use Literal type
- As best practice will use automatic typing inference of typescript whenever required

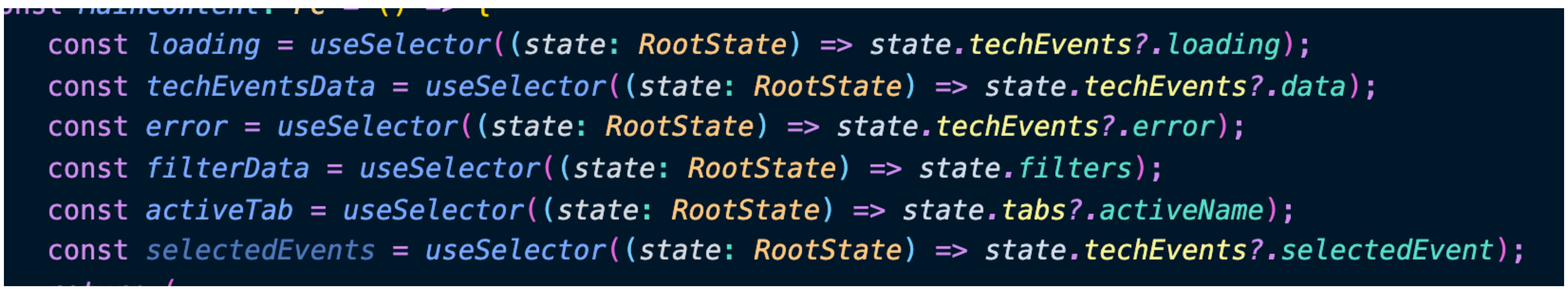


```
const cityMatch: boolean = city ? city === techEvent.city : true;
const freeEventsMatch: boolean = free ? techEvent.isFree : true;
```

Code

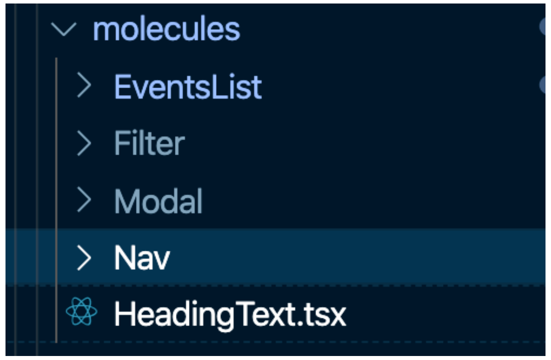
Code | Separation of concerns

- Did some separation of concerns (Container and Presentation components) but need to improve more on this
- Need ask myself do I have good separation of concerns in my application? Yes
- Remove un-necessary data from EventList.tsx

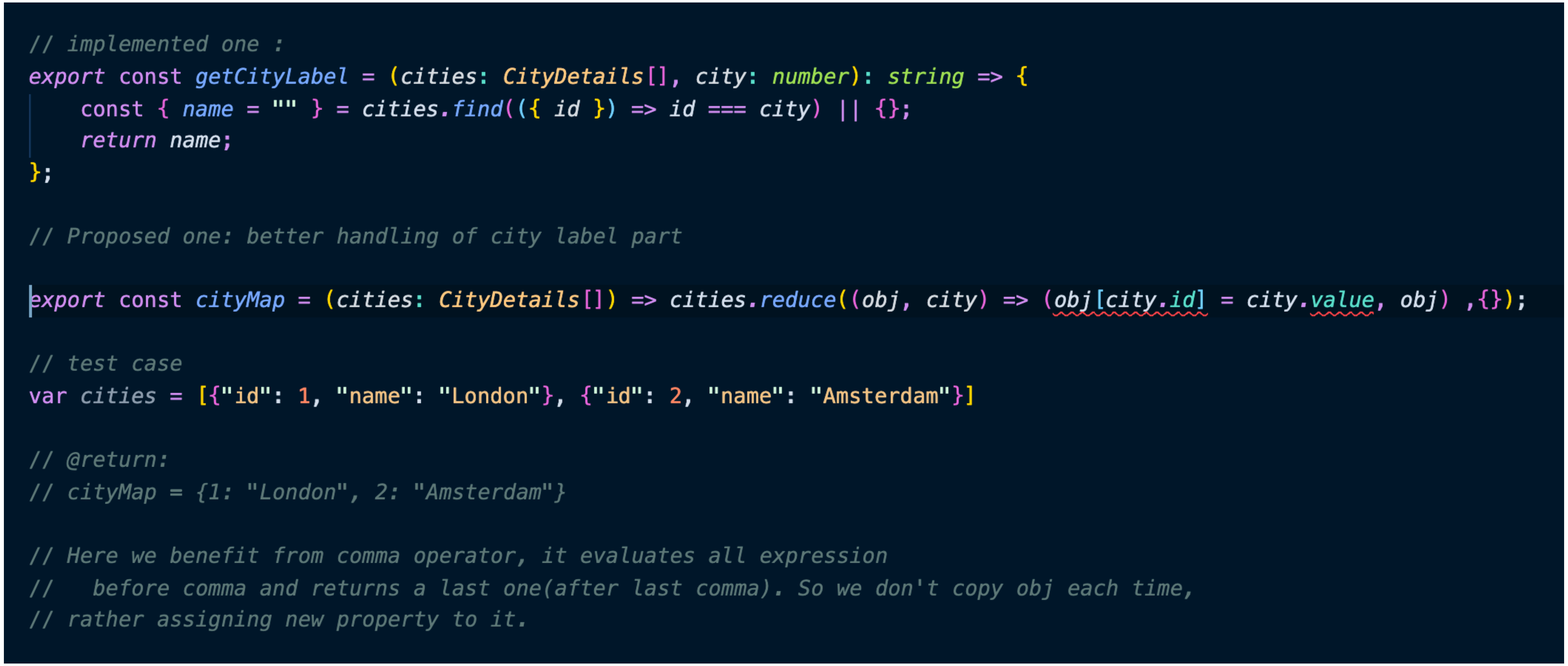


```
  const loading = useSelector((state: RootState) => state.techEvents?.loading);
  const techEventsData = useSelector((state: RootState) => state.techEvents?.data);
  const error = useSelector((state: RootState) => state.techEvents?.error);
  const filterData = useSelector((state: RootState) => state.filters);
  const activeTab = useSelector((state: RootState) => state.tabs?.activeName);
  const selectedEvents = useSelector((state: RootState) => state.techEvents?.selectedEvent);
```

- Instead of molecule “HeadingText” it should be atom



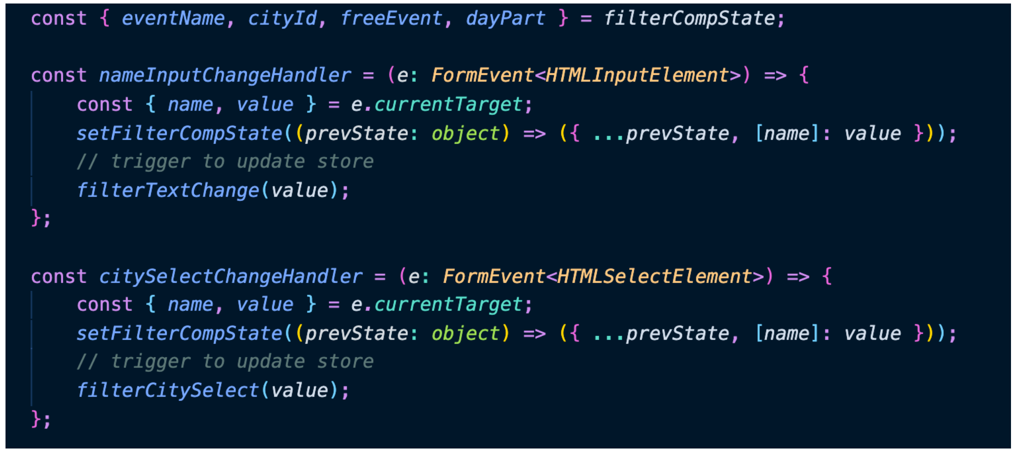
Code | Improve City Label method



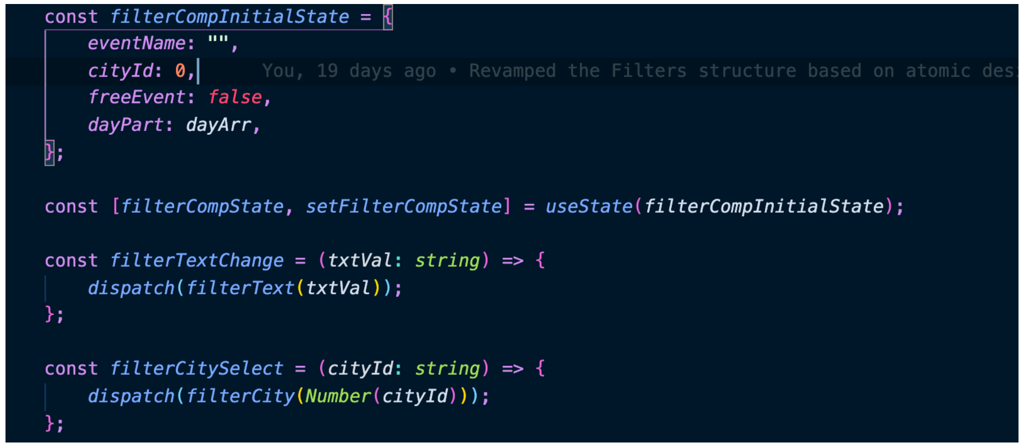
```
// implemented one :
export const getCityLabel = (cities: CityDetails[], city: number): string => {
    const { name = "" } = cities.find(({ id }) => id === city) || {};
    return name;
};

// Proposed one: better handling of city label part

export const cityMap = (cities: CityDetails[]) => cities.reduce((obj, city) => (obj[city.id] = city.value, obj) ,{});

// test case
var cities = [{"id": 1, "name": "London"}, {"id": 2, "name": "Amsterdam"}]

// @return:
// cityMap = {1: "London", 2: "Amsterdam"}

// Here we benefit from comma operator, it evaluates all expression
//   before comma and returns a last one(after last comma). So we don't copy obj each time,
// rather assigning new property to it.
```

Code | Better Form state handling | FilterContent

- Lot of code mess for similar operation of state and its update.
- At Filters.tsx (image 1)
- At FilterContent.tsx (image 2)



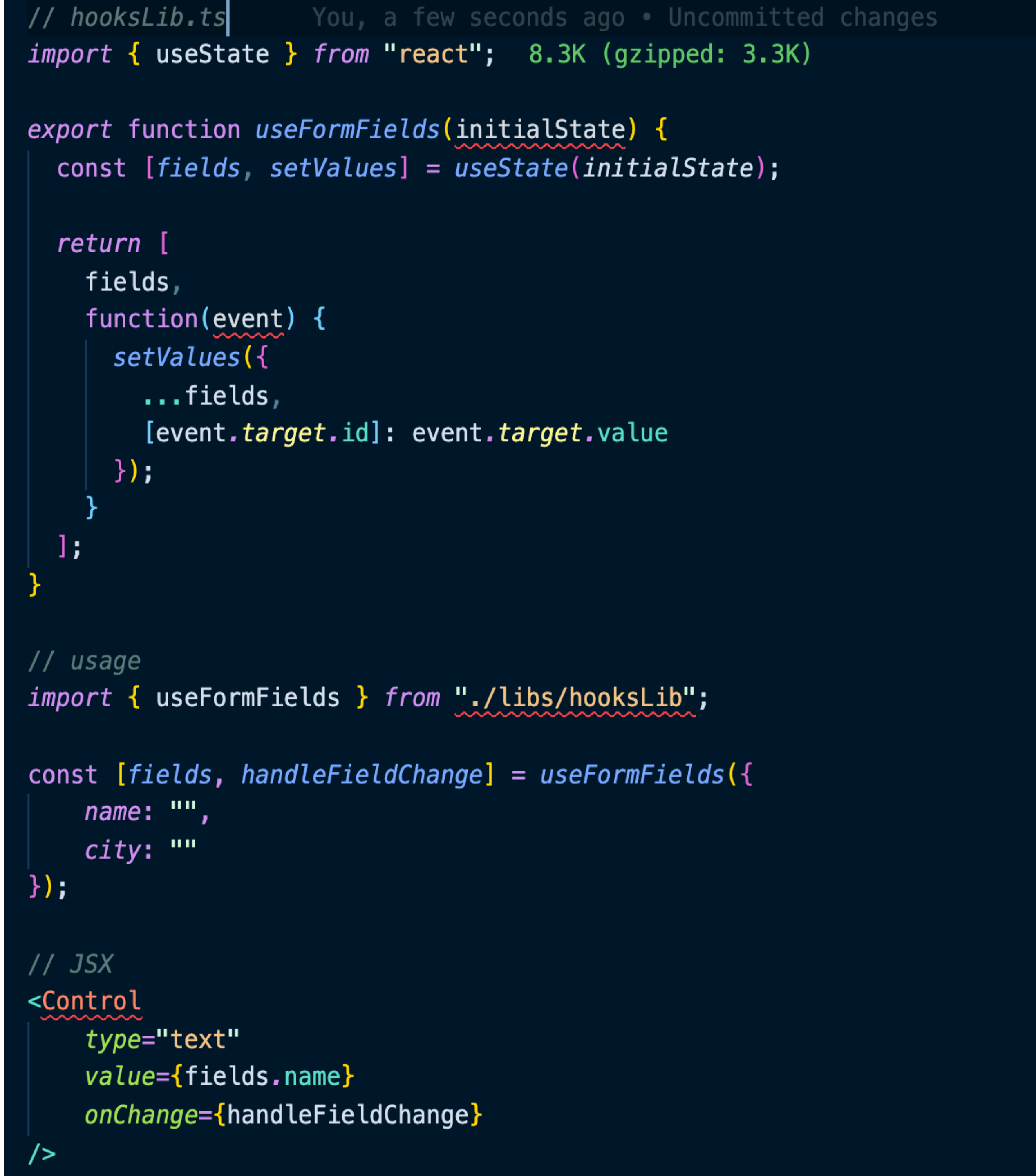
```
const { eventName, cityId, freeEvent, dayPart } = filterCompState;

const nameInputChangeHandler = (e: FormEvent<HTMLInputElement>) => {
    const { name, value } = e.currentTarget;
    setFilterCompState((prevState: object) => ({ ...prevState, [name]: value }));
    // trigger to update store
    filterTextChange(value);
};

const citySelectChangeHandler = (e: FormEvent<HTMLSelectElement>) => {
    const { name, value } = e.currentTarget;
    setFilterCompState((prevState: object) => ({ ...prevState, [name]: value }));
    // trigger to update store
    filterCitySelect(value);
};
```



```
const filterCompInitialState = {
    eventName: "",
    cityId: 0,      You, 19 days ago • Revamped the Filters structure based on atomic des
    freeEvent: false,
    dayPart: dayArr,
};

const [filterCompState, setFilterCompState] = useState(filterCompInitialState);

const filterTextChange = (txtVal: string) => {
    dispatch(filterText(txtVal));
};

const filterCitySelect = (cityId: string) => {
    dispatch(filterCity(Number(cityId)));
};
```

Code | Better Form state handling | FilterContent Conti...

- Instead we can implement simple hooksLib.ts and utilize it effectively



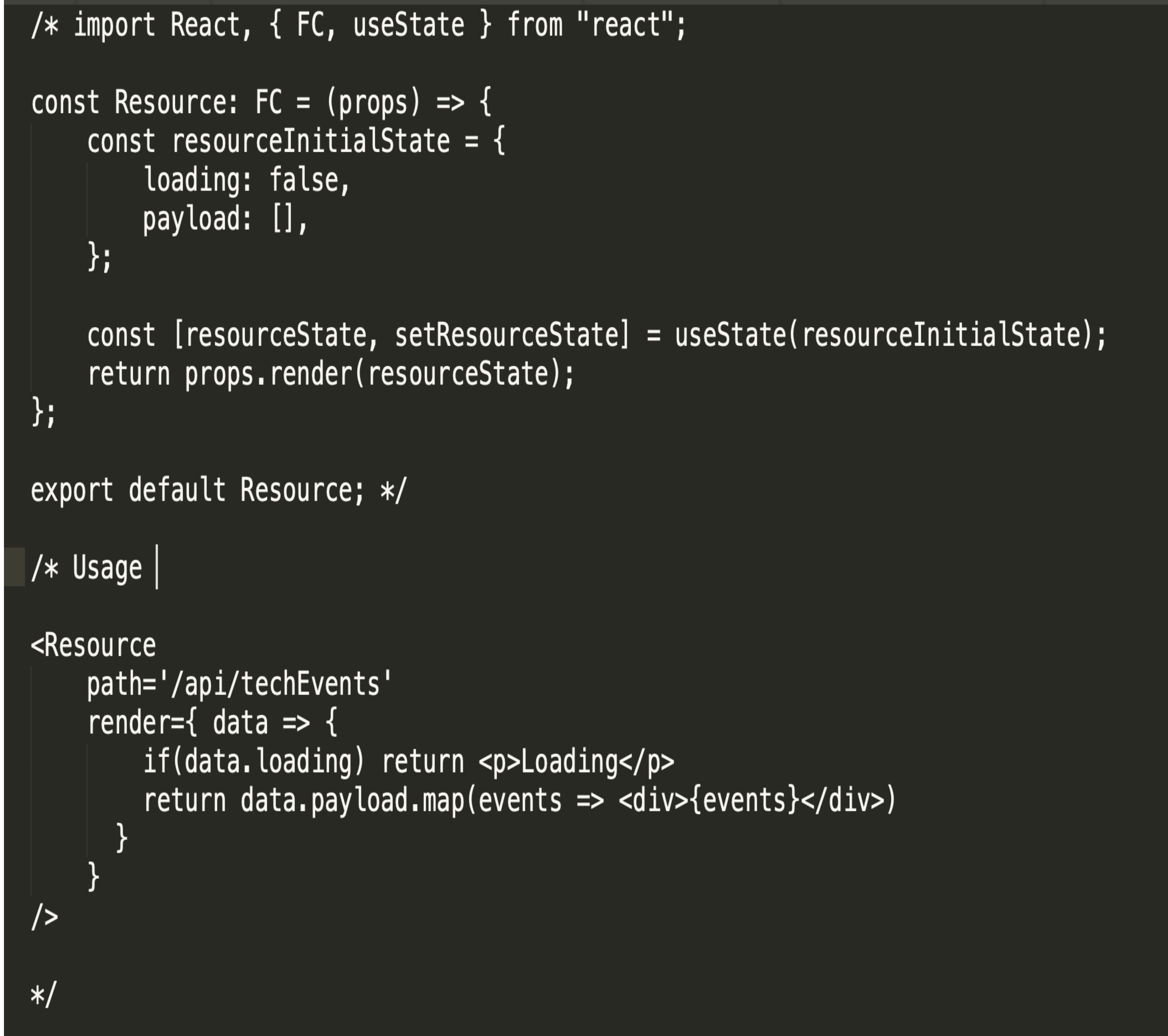
```
// hooksLib.ts        You, a few seconds ago • Uncommitted changes
import { useState } from "react";   8.3K (gzipped: 3.3K)

export function useFormFields(initialState) {
  const [fields, setValues] = useState(initialState);

  return [
    fields,
    function(event) {
      setValues({
        ...fields,
        [event.target.id]: event.target.value
      });
    }
  ];
}

// usage
import { useFormFields } from "./libs/hooksLib";

const [fields, handleFieldChange] = useFormFields({
    name: "",
    city: ""
});

// JSX
<Control
    type="text"
    value={fields.name}
    onChange={handleFieldChange}
/>
```

Code | Few more points

- Instead of utils complete file, will have Selectors pattern for filtering events (single selectors.ts) – It can refined more
- Instead of calling every time mock API, will store in LocalStorage(LS) and update if new data comes. Perform operations on LS data and consume it in reducers via utility
- Instead of props.children plainly for Modal, we can use React.Children.map more effectively. Especially with elements, provides array kind methods etc.
- We can use currying for signUpClickHandler in EventItem.tsx
- Implement lazy loading for events list - Usage of React virtualization also helps (https://bvaughn.github.io/react-virtualized/#/components/List)

Code | Few more points conti...

Will implement <Resource /> component for API call, loading state and payload which will be reusable across application (Pattern render Props)



```
/* import React, { FC, useState } from "react";

const Resource: FC = (props) => {
    const resourceInitialState = {
        loading: false,
        payload: [],
    };

    const [resourceState, setResourceState] = useState(resourceInitialState);
    return props.render(resourceState);
};

export default Resource; */

/* Usage

<Resource
    path='/api/techEvents'
    render={ data => {
        if(data.loading) return <p>Loading</p>
        return data.payload.map(events => <div>{events}</div>)
      }
    }
/>

*/
```

Development Config steps

- Creation:

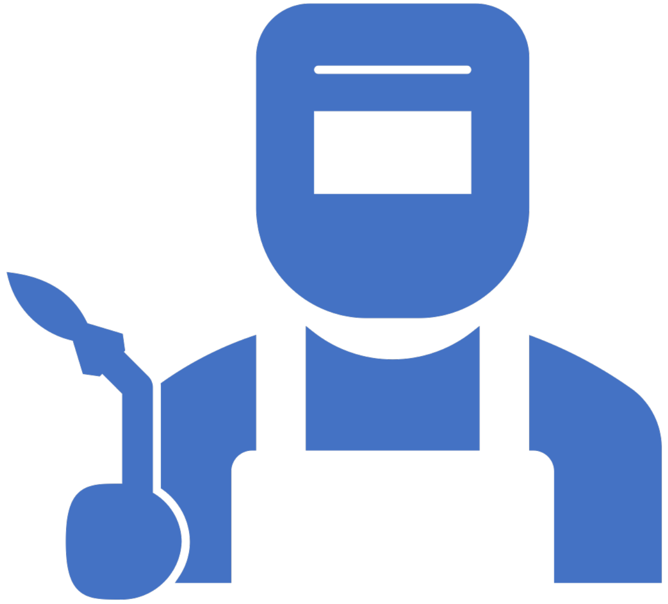
```
$ npx create-react-app techevents-app --template typescript
```

- Dev tools:

```
$ npm i --save-dev prettier tslint tslint-config-prettier tslint-plugin-prettier tslint-react
$ touch tslint.json // Added custom configurations
$ touch .prettierrc // Added rules for prettier
```

In package.json added commands:

  - tslint
  - tslint:fix
  - tslint:check
  - prettier



- Redux:
  - $ npm i redux-thunk
  - $ npm i redux react-redux @types/react-redux redux-devtools-extension @fortawesome/fontawesome-free
  - Middleware: $ npm i redux-thunk
- Deleted unnecessary files:
  - App.test.tsx,
  - logo.svg,
  - index.css,
  - serviceWorker.ts and
  - setupTests.ts files.
- Added moment.js: $ npm i moment
- For Deployment over free Git pages:
  - $ npm install gh-pages --save-dev
  - Updated package.json - homepage: <url>
- At last added media queries

Thank you