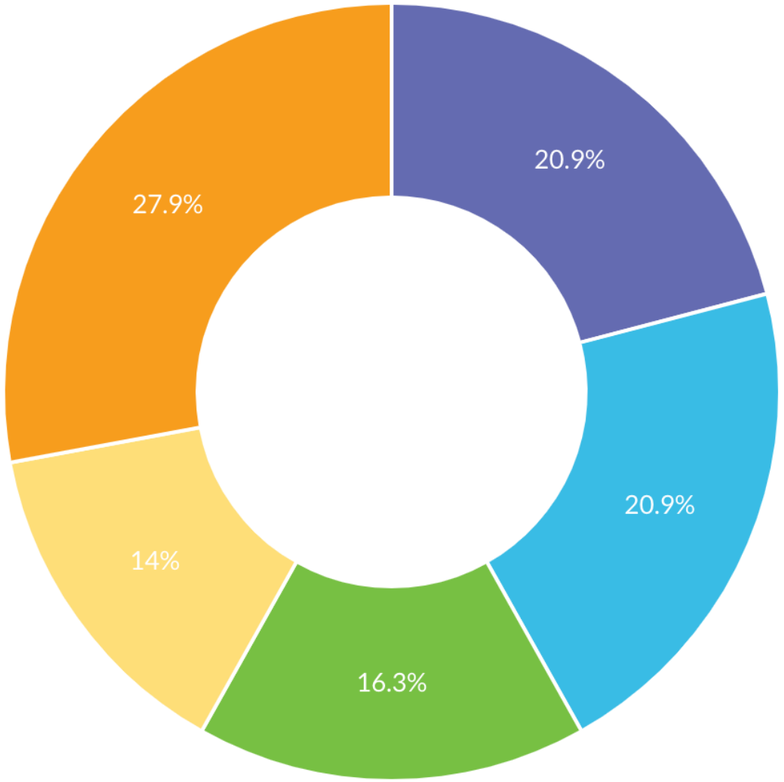
SELECTED CHAPTERS WEB

ABOUT/CONTACT

» Thomas Mayrhofer (@webpapaya)

» E-Mail: tmayrhofer.lba@fh-salzburg.ac.at

SURVEY RESULTS



Choice	Total
Error handling in React	9
Code Splitting with React	9
Client Side Debugging	7
Testing Redux	6
Portals and Refs	12

ROADMAP

» Portals

» Refs

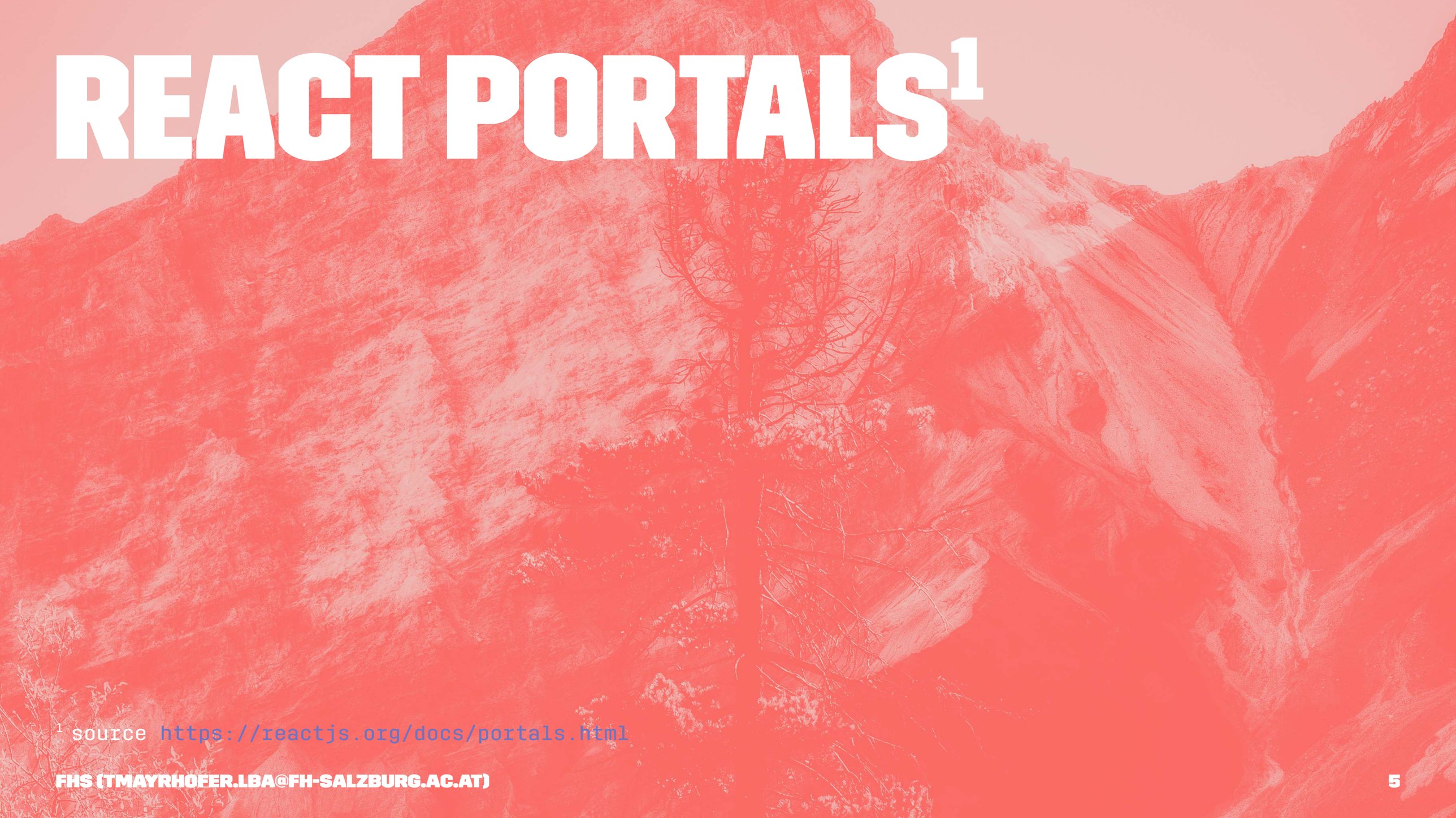
» Error Boundaries

» Code Splitting

» ~~Client Side Debugging~~

» ~~Testing Redux~~

REACT PORTALS[1]



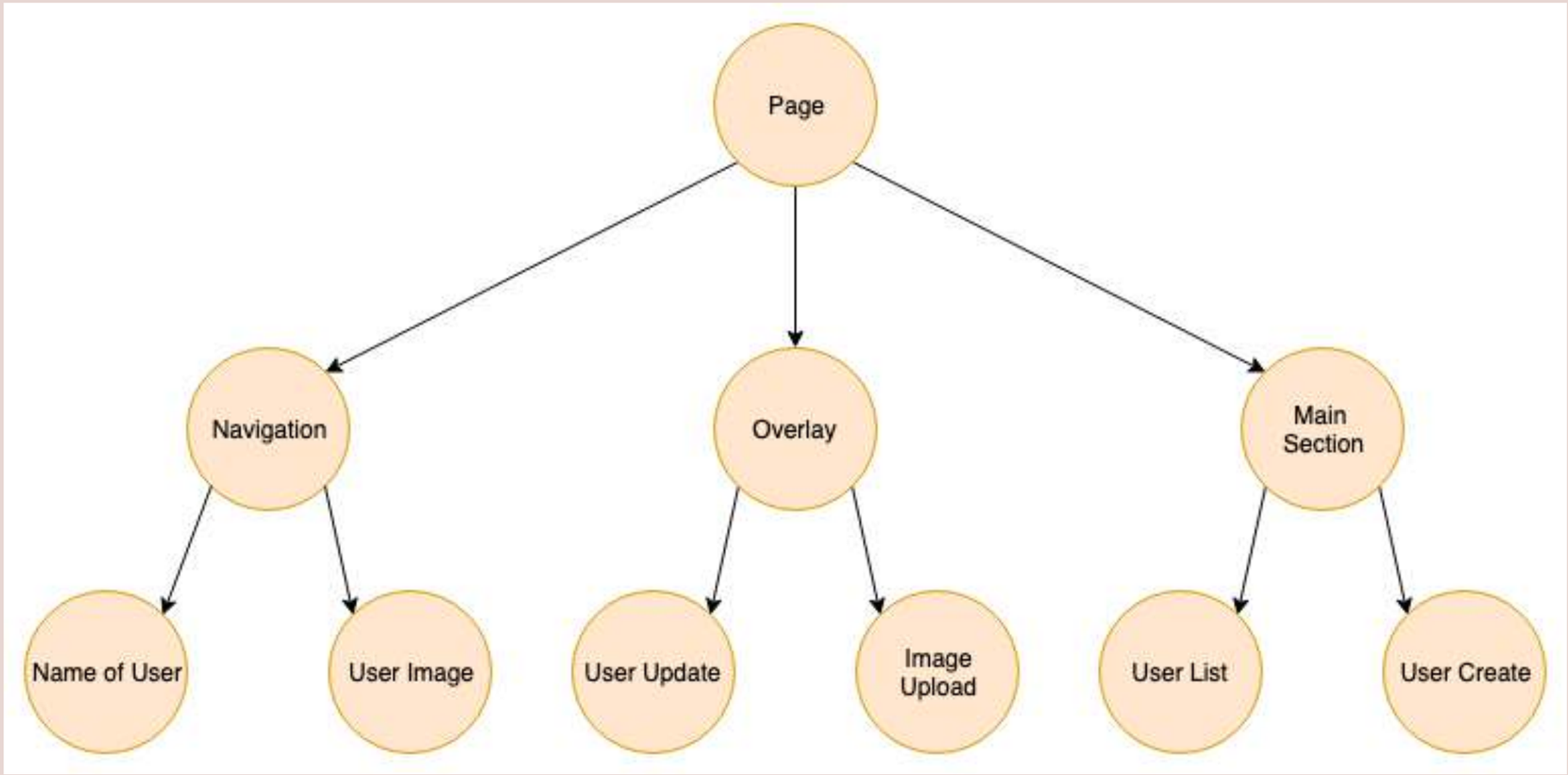
[1] source https://reactjs.org/docs/portals.html

REACT PORTALS[1]

“Portals provide a first-class way to render children into a DOM node that exists outside the DOM hierarchy of the parent component.”

[1] source https://reactjs.org/docs/portals.html

REACT PORTALS[1]



[1] source https://reactjs.org/docs/portals.html

REACT PORTALS[1]

» Components are rendered inside nearest parent DOM

```
const MyComponent = ({ children }) => {
  return (
    <div>
      {children}
    </div>
  );
}
```

[1] source https://reactjs.org/docs/portals.html

REACT PORTALS[1]

» Sometimes components need to render outside of DOM nodes

» eg. modals

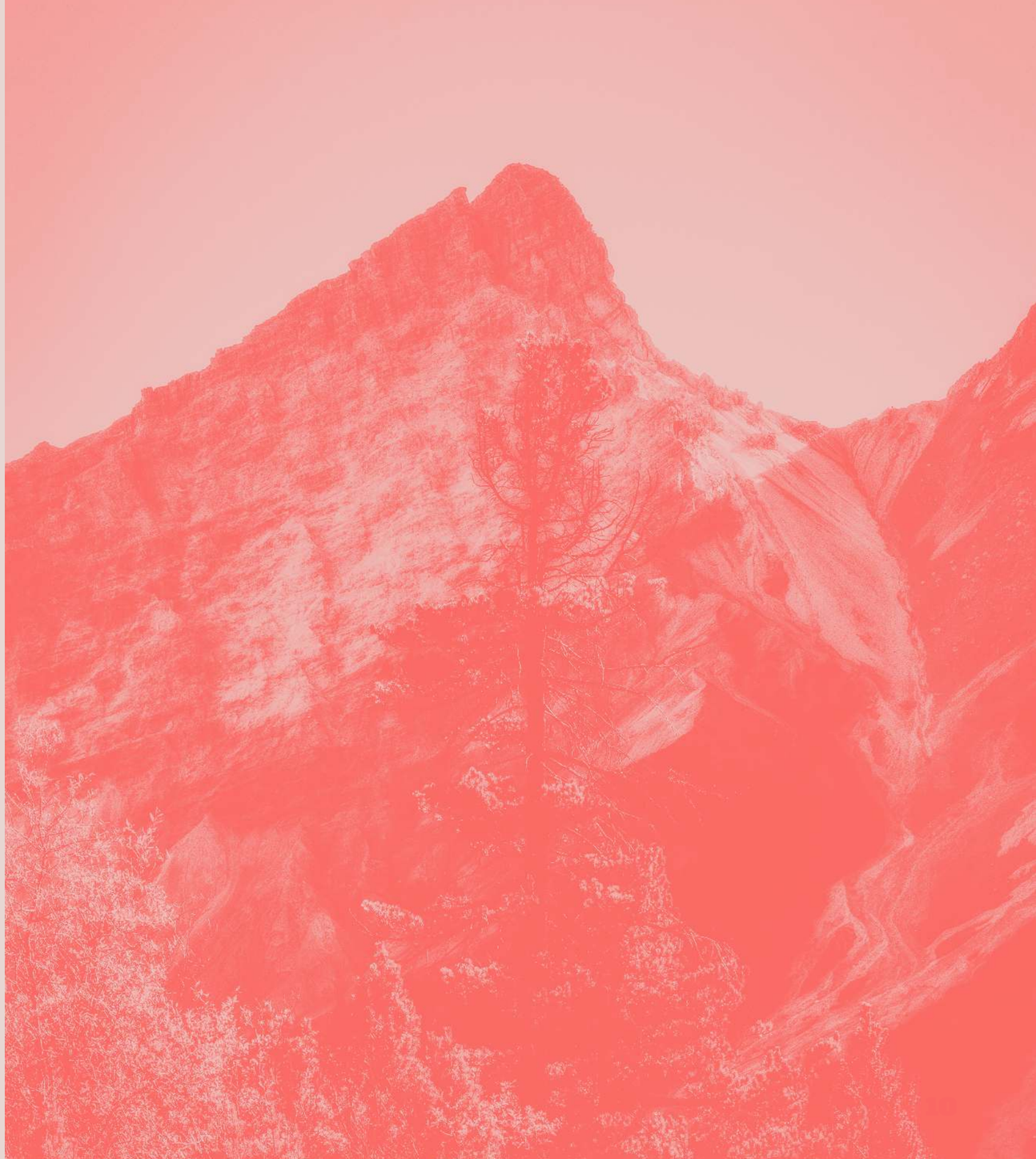
```
const MyComponent = ({ children }) => {
  return ReactDOM.createPortal(
    children,
    // specify elements to be rendered

    document.getElementById("myComponentRoot")
    // children will be rendered inside #myComponentRoot
  );
}
```

[1] source https://reactjs.org/docs/portals.html

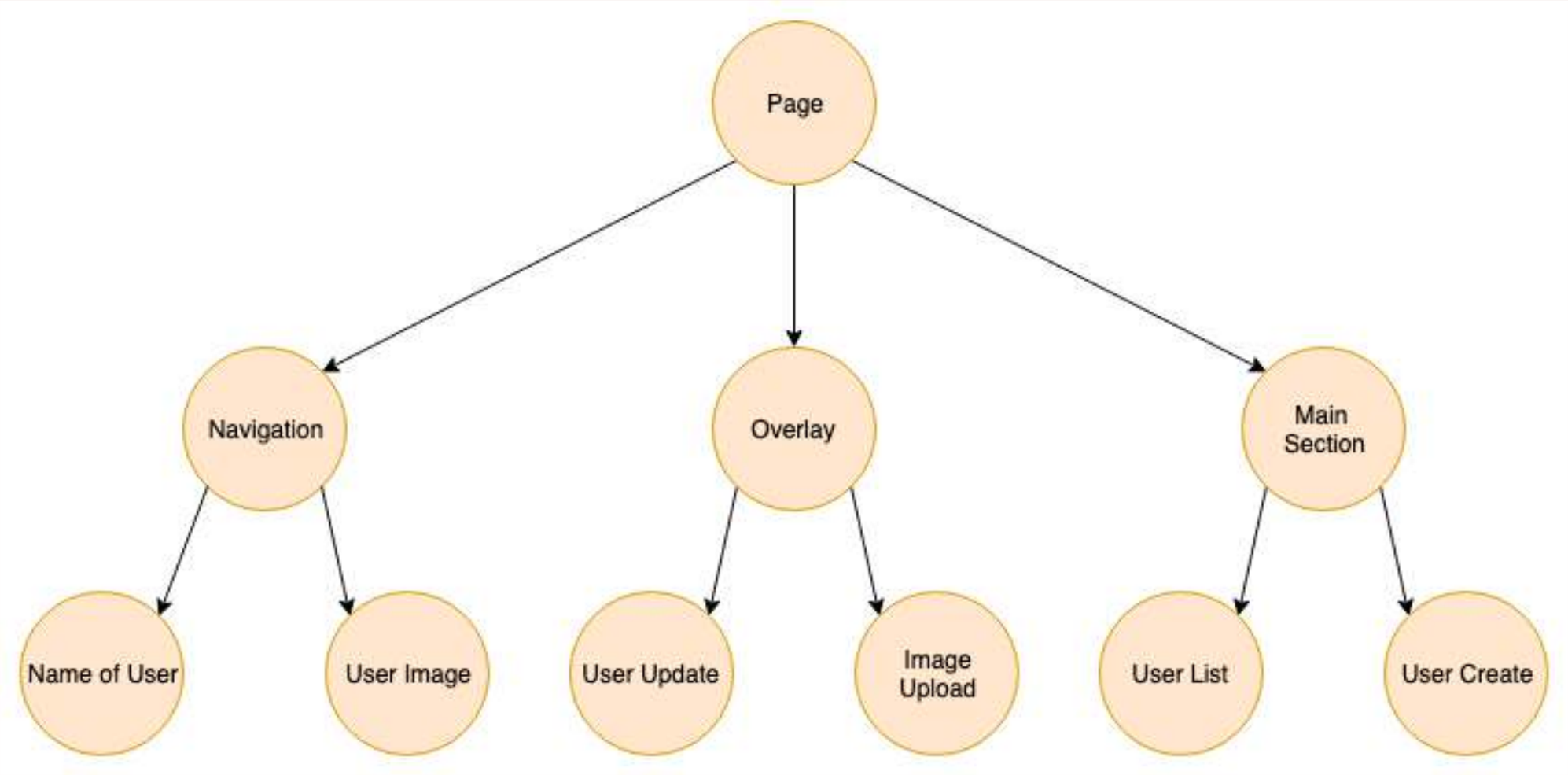
REFS

“Refs provide a way to access DOM nodes or React elements created in the render method.”

FHS (TMAYRHOFER.LBA@FH-SALZBURG.AC.AT)



REFS


Page
Navigation
Overlay
Main Section
Name of User
User Image
User Update
Image Upload
User List
User Create

REFS [2]

- » props are used to interact with child components
  - » to modify children rerender with different props
  - » might be triggered via setState/useState
- » refs are used to access DOM elements directly
  - » setting focus
  - » trigger imperative animations

[2] source https://reactjs.org/docs/refs-and-the-dom.html

REFS [2]

```
const MyInputComponent = ({ autofocus }) => {
    const inputRef = useRef(null);
    //                 ^^^^^^^^^^^^^
    // specify a reference with an initial value of null

    useEffect(() => {
        inputRef.current?.focus();
        //       ^^^^^^^^
        // current might be <input /> or null
        // when <input /> then focus input
    }, [inputRef.current])


    return <input ref={inputRef} />
    //             ^^^^^^^^^^^^^^^
    // specify to which element the ref should be bound
}
```

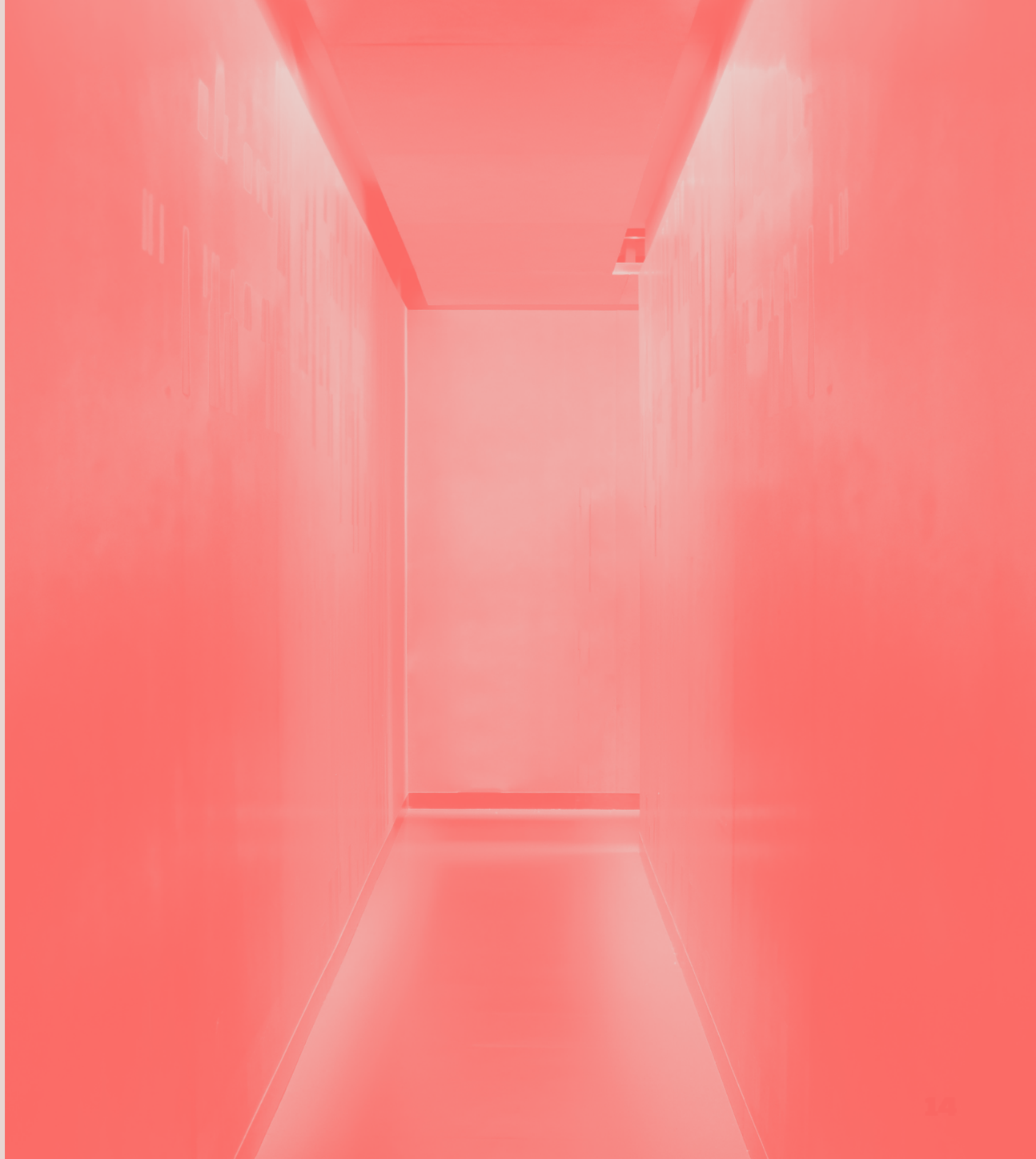
[2] source https://reactjs.org/docs/refs-and-the-dom.html

REFS [2]

» Don’t Overuse Refs

» prefer using props and lift up state

[2] source https://reactjs.org/docs/refs-and-the-dom.html



ERROR BOUNDARIES [3]

[3] source https://reactjs.org/docs/error-boundaries.html

ERROR BOUNDARIES [3]

- an error in a component might break the whole app
  - react mixes up state
  - reloading page is the only way to recover
  - bad UX
- Error boundaries handle errors of components

[3] source https://reactjs.org/docs/error-boundaries.html

ERROR BOUNDARIES [3]

- static getDerivedStateFromError
  - invoked after an error in child has been thrown
  - should return new component state
- componentDidCatch
  - invoked after an error in child has been thrown
  - used to trigger side effects (eg. server logging)

[3] source https://reactjs.org/docs/error-boundaries.html

ERROR BOUNDARIES [3]

```
class ErrorBoundary extends React.Component {
    state = { hasError: false }
    static getDerivedStateFromError(error) {
        // Update state so the next render will show the fallback UI.
        return { hasError: true };
    }

    componentDidCatch(error, errorInfo) {
        // You can also log the error to an error reporting service
        logErrorToMyService(error, errorInfo);
    }

    render() {
        if (this.state.hasError) {
            // You can render any custom fallback UI
            return <h1>Something went wrong.</h1>;
        }

        return this.props.children;
    }
}
```

[3] source https://reactjs.org/docs/error-boundaries.html

ERROR BOUNDARIES [3]
USING ERROR BOUNDARIES

- using boundaries is up to the developer
- place boundaries around use-cases [4] eg:
  - sign-up form
  - create money-transactions
  - money-transaction list

[3] source https://reactjs.org/docs/error-boundaries.html

[4] personal opinion

CODE SPLITTING

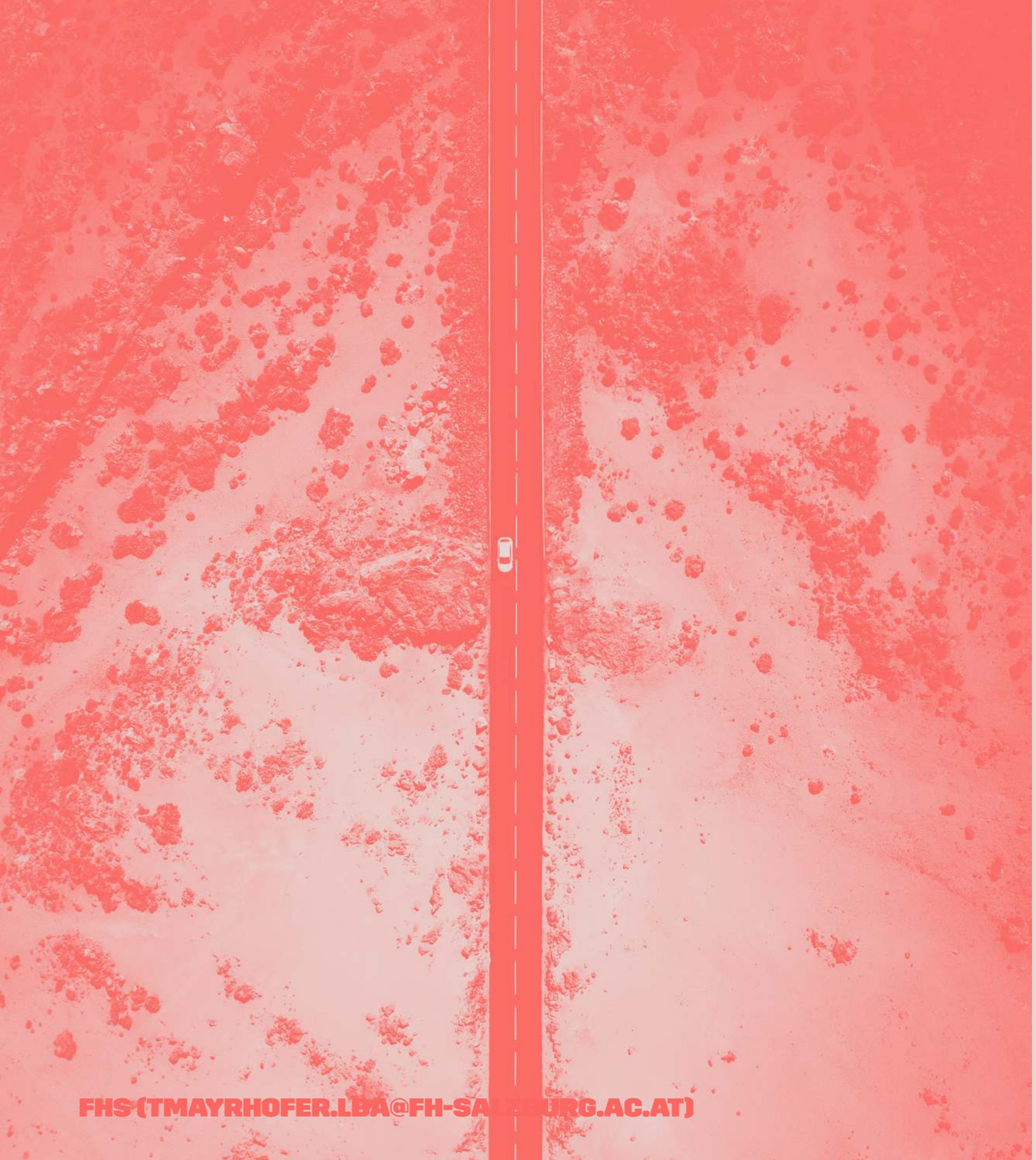
CODE SPLITTING
BUNDLING

“combine multiple JS modules into a single bundle”

» when apps grow the bundle will grow

» large libraries are added to bundle

  » increases download/parse time



CODE SPLITTING [5]

“split bundle into multiple bundles which can be loaded lazily during runtime”

[5] source https://reactjs.org/docs/code-splitting.html

CODE SPLITTING [5]
DYNAMIC IMPORTS

```
const add = async (a, b) => {
    const math = await import("./math")
    //                ^^^^^^^^^^^^^^^^^^^^^
    // load "./math" module only when `add` is executed
    // for the first time. Bundlers will split the bundle
    // automatically. (2 files will be generated)

    return math.add(a, b)
});
```

[5] source https://reactjs.org/docs/code-splitting.html

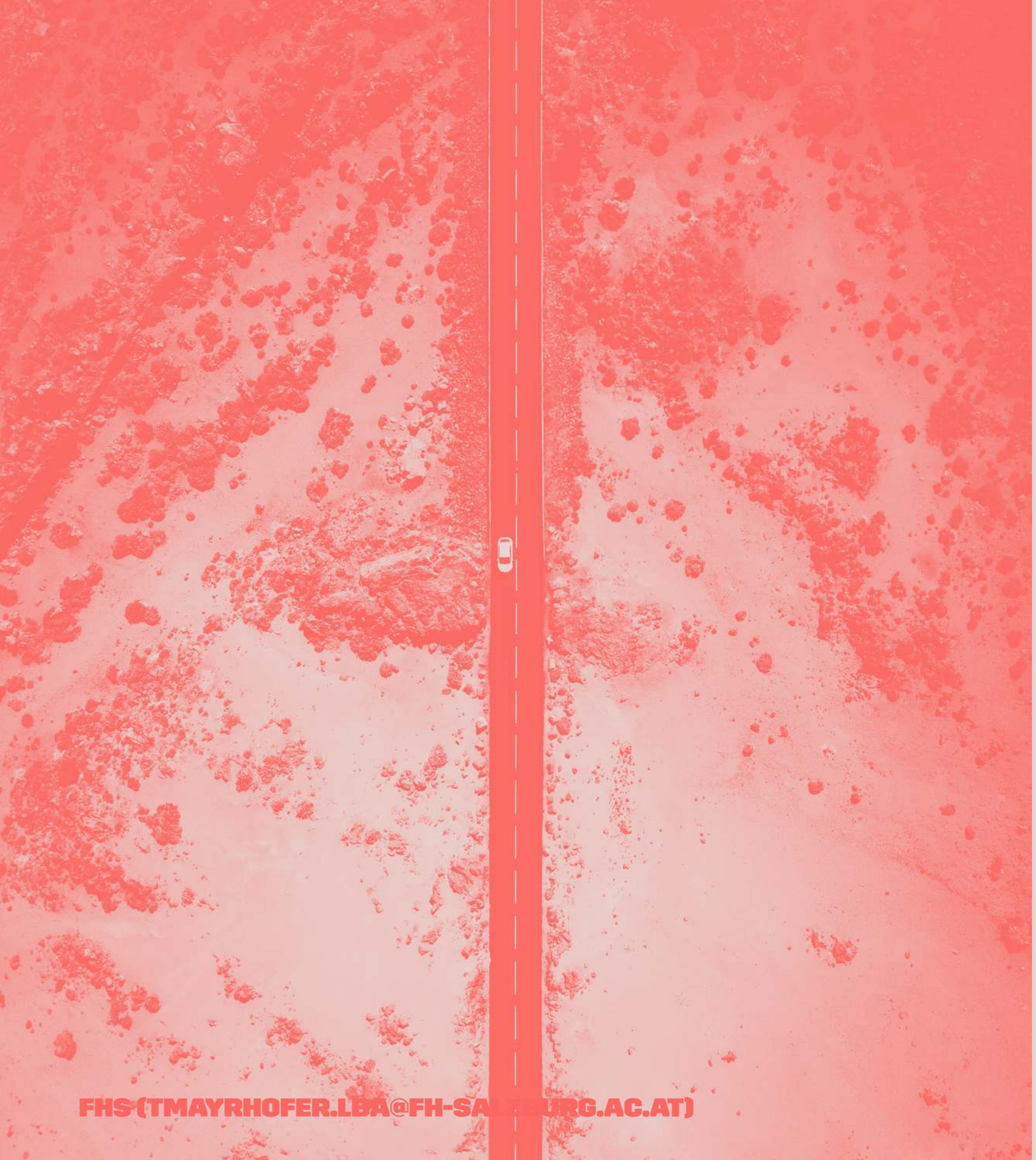
CODE SPLITTING [5]

React.lazy

“The React.lazy function lets you render a dynamic import as a regular component.”

```
const OtherComponent = React.lazy(() => import('./OtherComponent'));
//                                       ^^^^^^
// dynamically import `./OtherComponent` when the component is rendered
// for the first time.
// `./OtherComponent` needs to be exported as default
```

[5] source https://reactjs.org/docs/code-splitting.html



CODE SPLITTING [5]

React.Suspense

» importing modules dynamically is an async operation

» a loading screen needs to be displayed

“React.Suspense lets components “wait” for something before rendering.”

[5] source https://reactjs.org/docs/code-splitting.html

CODE SPLITTING [5]

React.Suspense

```
import React, { Suspense } from 'react';

const OtherComponent = React.lazy(() => import('./OtherComponent'));

function MyComponent() {
  return (
      <Suspense fallback={<div>Loading...</div>}>
        <OtherComponent />
      </Suspense>
  );
}
```

[5] source https://reactjs.org/docs/code-splitting.html

OTHER MASTER TOPICS

» Testing Client Side Apps Deep Dive

» Performance Tuning in React/Redux

» Functional Programming in JS

» Scaling React/Redux

» Debugging in JS

» Localization

» ...

LETS KEEP IN TOUCH

» thomas@mayrhofer.at

» Twitter @webpapaya

» LinkedIn