Callback function – invoked by the eventhandler **onFormSubmit** in the SearcBar Component

The **onFormSubmit** method is attach as EventHandler to the **onSubmit** event

React App Pics - Event handling overview Part 1

C/Henrik Høltzer Zealand 12.02.2020

https://api.unsplash.com/search/photo/term

**Unsplash WS**

class **App** extends React.Component {

  state = {**images**: []};

**onSearchSubmit** = async (**term)** =>{

    const response = await unsplash.get('/search/photos', {params: {query: **term**}});

    this.setState({**images**: response.data.results});

  }

  render(){

   return (

      <div className="ui container" style={{marginTop: '10px'}}>

        <**SearchBar** **onSubmit**={this.**onSearchSubmit**}/>

        <ImageList **images**={this.state.**images**}/>

        Found: {this.state.**images**.length} images

      </div>

    );

  }

}

**onSubmit** invokes the handler method **onFormSubmit**

class **SearchBar**extends React.Component {

   state = { **term**:  ''};

**onFormSubmit** = (event) =>{

        event.preventDefault();

        this.props.**onSubmit**(this.state.**term**);

    }

    render(){

        return (

        <div className="ui segment">

            <form onSubmit={this.**onFormSubmit**} className="ui form">

                <div className="field">

                    <label>Image Search</label>

                    <input

                        type="text"

                        value={this.state.**term**}

                        onChange={(e)=>this.setState({ **term**: e.target.value})}

                    />

                </div>

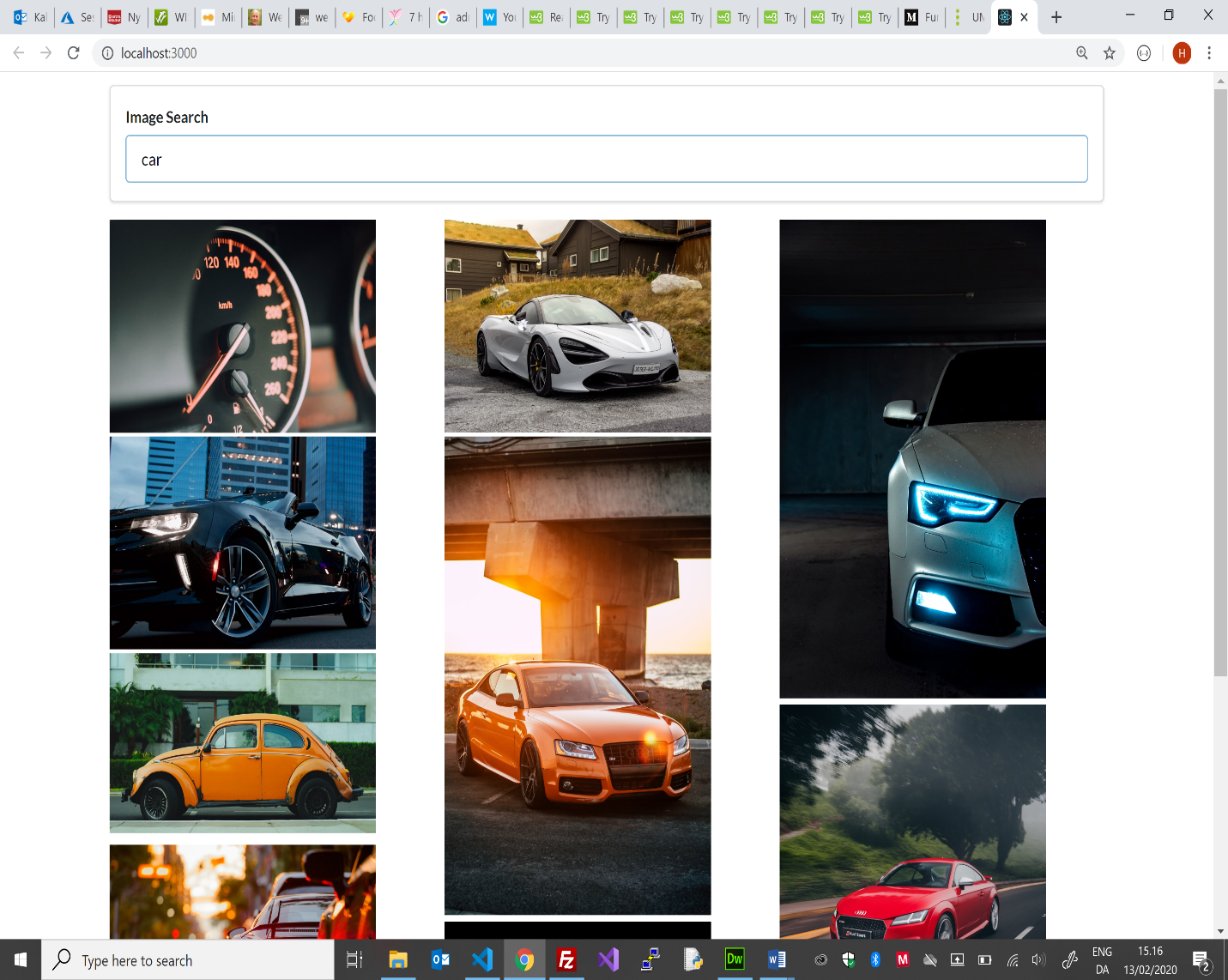
            </form>

        </div>

        );

    }

}



**onChange** update the state term

**onSubmit** is called on Return

The Eventhandler method **onFormSubmit** call the function (**onSerchSubmit**) binded to the props **onSubmit** with the state term as argument

preventDefault() – prevent any default actions to be executed on submit – all we need is to invoke the Callback function

React App Pics - Event handling overview Part 2

C/Henrik Høltzer Zealand 12.02.2020

https://api.unsplash.com/search/photo/car

**Unsplash WS**

class **App** extends React.Component {

  state = {**images**: []};

**onSearchSubmit** = async (**term)** =>{

    const response = await unsplash.get('/search/photos', {params: {query: **term**}});

    this.setState({**images**: response.data.results});

  }

  render(){

   return (

      <div className="ui container" style={{marginTop: '10px'}}>

        <SearchBar **onSubmit**={this.**onSearchSubmit**}/>

        <**ImageList** **images**={this.state.**images**}/>

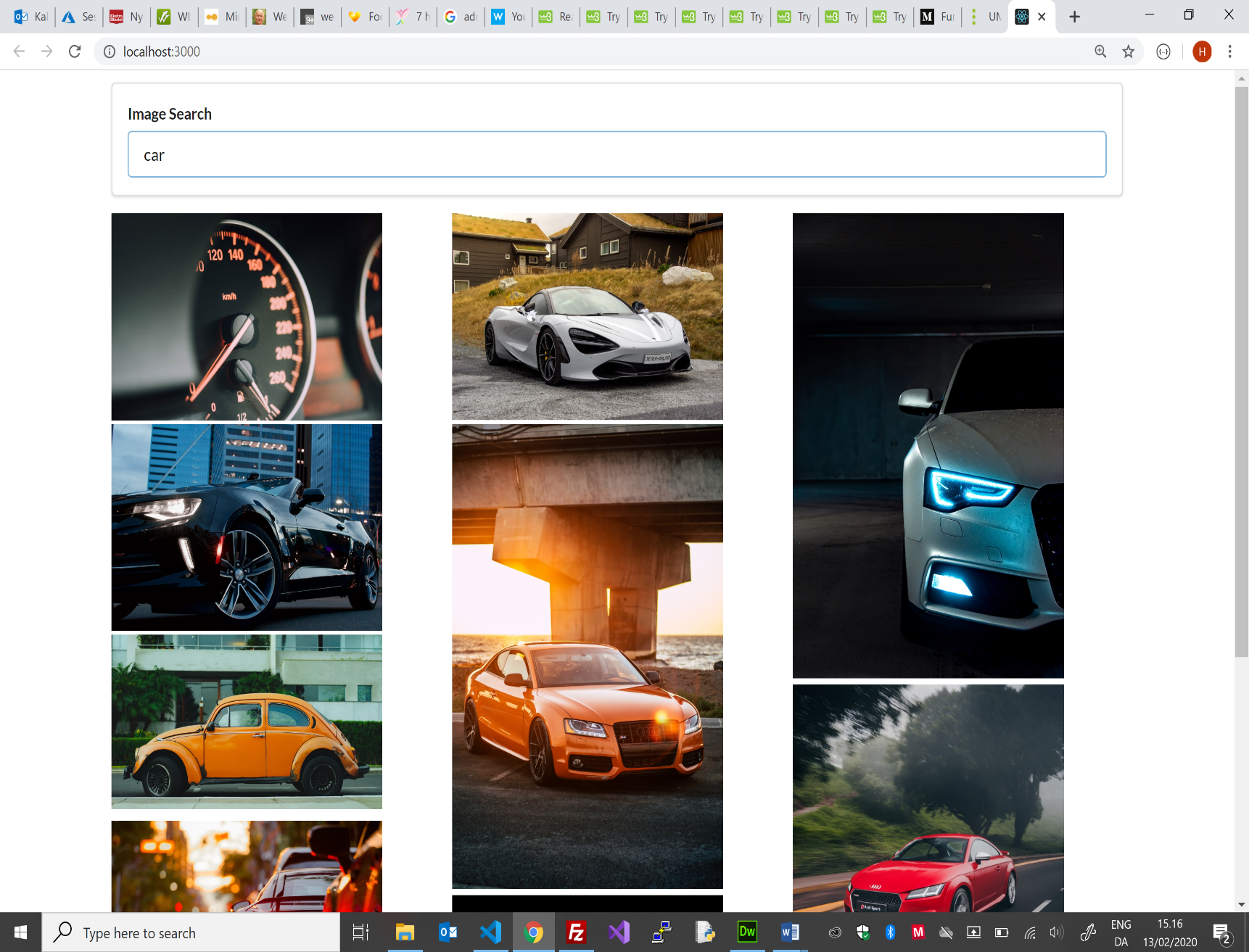
        Found: {this.state.**images**.length} images

      </div>

    );

  }

}



Each child in a list must have a unique “key” props

description: null

urls.regular: <https://images.unsplash.com/photo-1498887960847-2a5e46312788>

height: 167, spans: 17

gridRowEnd : span XX

XX specify the no of rows/tracks the Image scould span over

Math.ceil rounds upwards to nearest int

clientHeight is the hight of the mounted image

current is a ref to the mounted instance of the ImageCard component

Create a ref to ImageCard

Load event is fired when the image has been loaded and the callback function setSpan is invoked

class **ImageCard** extends React.Component {

    state = {spans:0};

    imageRef = React.createRef();

    componentDidMount() {

        this.imageRef.current.addEventListener('load', this.setSpans)

    }

    setSpans = () => {

        const height = this.imageRef.current.clientHeight;

        const spans = Math.ceil(height/10);

        this.setState({spans});

    }

    render(){

        const {description, urls} = this.props.image;

        return (

            <div style={{gridRowEnd: `span ${this.state.spans}`}}>

                <img

                    ref={this.imageRef}

                    alt={description}

                    src={urls.regular}

                />

            </div>

        );

    }

}

const **ImageList** = (props) => {

    const images = props.images.map((image) => {

        return <**ImageCard** key={image.id} image={image} />;

    });

    return <div className="image-list">{images}</div>;

};