SCoRe Lab

Project Introduction:

Project Name: React Email ([#13](#))

Time: 175Hours (Medium) (12 weeks)

Project Mentor: Milindu Sanoj Kumarage

Problem: Modern-day web browsers are pretty standardized and come with a certain guarantee that if a web page works in one browser, it will work in another web browser too except for a few minor cases. However, this is not true for email clients yet.

The current state of the problem:
The developer community is thus aligned to certain tricks and hacks to make sure the emails sent are properly rendered by these different email clients.
There is no such package that solves this problem of creating responsive emails easily which can also handle cross browsers and clients compatibility issues.

Proposed Solution to this problem:
The end goal will be to implement a React-based component and utility methods-based lightweight library to provide a common interface for email building that users can install and use to build clean and responsive emails easily.

Project Goals and Implementation:

- **Selecting the module bundler and project initialization:**
  Initialize the project using TypeScript, based on Rollup as the bundler along with storybook and jest.
  Need to look for Rollup based Node Module building boilerplates.

  Implementation:
  https://github.com/vijayst/react-package-boilerplate
  We can take references from this code and set up the project along with the storybook and jest.

- **Creating initial components:**
  After setting up the project, initially, we need a few components. Some important ones are - Email, Grid, Row, Cell, Typography, Image, Button, Link, Quote, Divider, Section, etc.

  Few ideas for the components:

  1. **Email**: It will be the top-level component, and the users need to wrap the overall content of the email and all other components with this one. It will include all the necessary DOCTYPE, html, head, meta, and body tags.

  2. **Typography:** It will be used to render text

  3. **Cell**: We can use Cell to create new rows and columns

  4. **Image:** Component to render responsive images

  5. **Button:** Component for adding buttons to the email

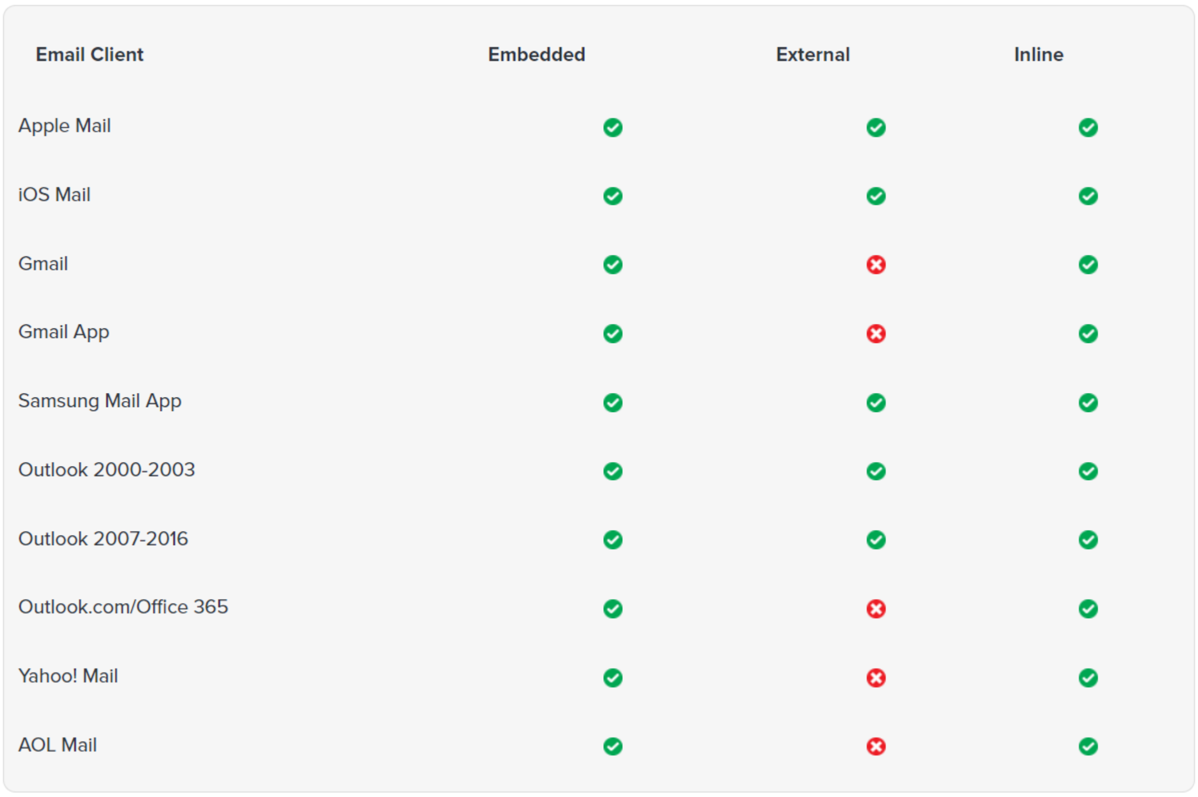
  6. **Link:** For adding hyperlinks to the emails

  7. **Divider:** A divider can be used to add a line between two sections

Implementation:

There are a few things that we should keep in mind while building the components.

- Using `table`s for layout rather than `div`s - reference
- Using Inline CSS styles instead of a class-based approach due to the different email clients' compatibility issues.



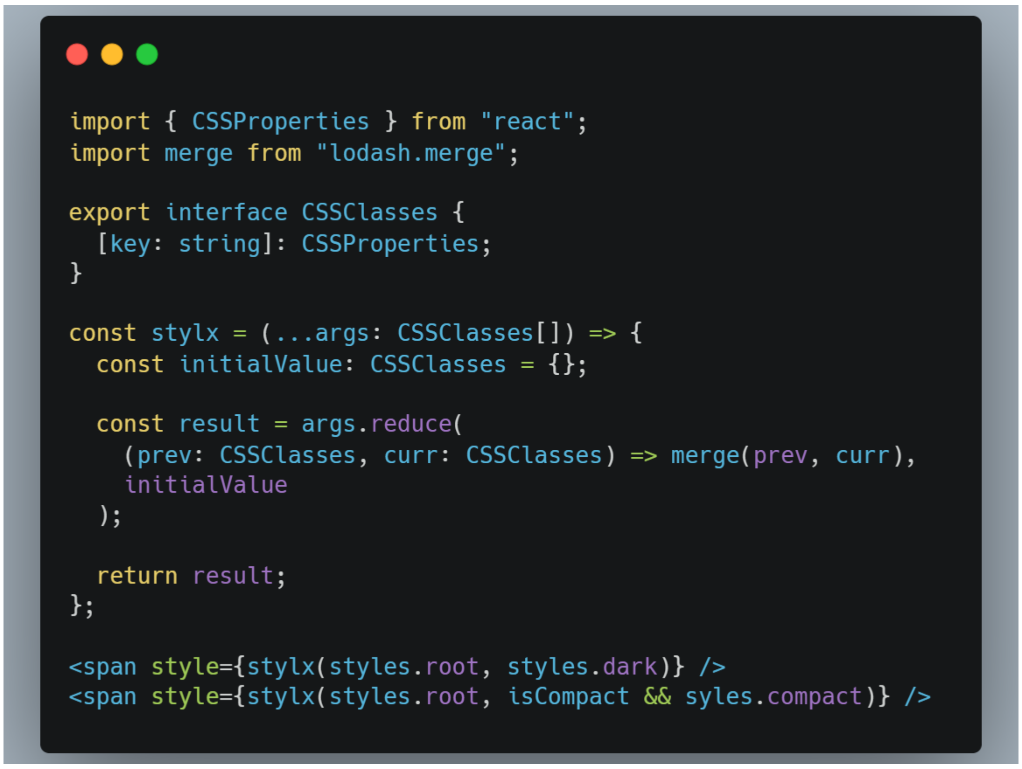
Email Client	Embedded	External	Inline
Apple Mail	✓	✓	✓
iOS Mail	✓	✓	✓
Gmail	✓	✗	✓
Gmail App	✓	✗	✓
Samsung Mail App	✓	✓	✓
Outlook 2000-2003	✓	✓	✓
Outlook 2007-2016	✓	✓	✓
Outlook.com/Office 365	✓	✗	✓
Yahoo! Mail	✓	✗	✓
AOL Mail	✓	✗	✓

- Also, these are a few rules and best practices that we can follow: https://templates.mailchimp.com/getting-started/html-email-basics

- **Styling the components:**

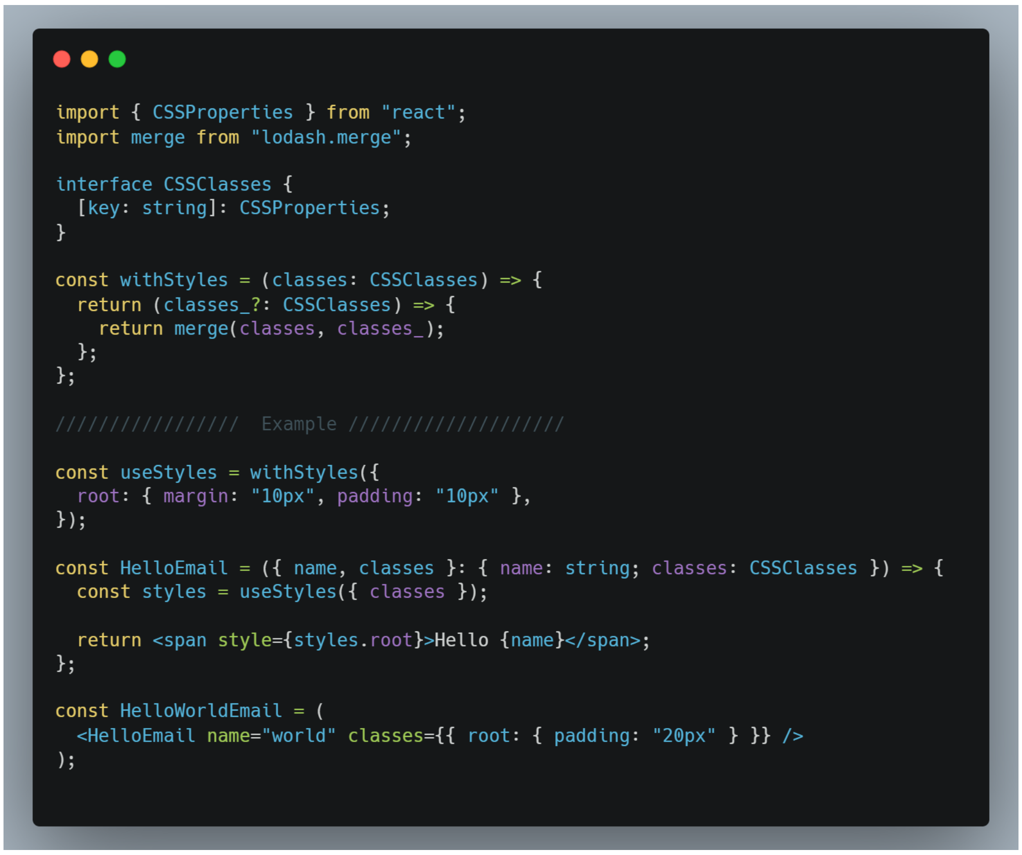
Styling is also important. We can implement a utility method to combine styles, a hook for exposing styles to components with style overrides. But there might be some limitations as we are going to use tables instead of divs and other commonly used tags, also we need to keep in mind the browser and client compatibility issues also.

- “stylx” method will be used to combine styles in this way



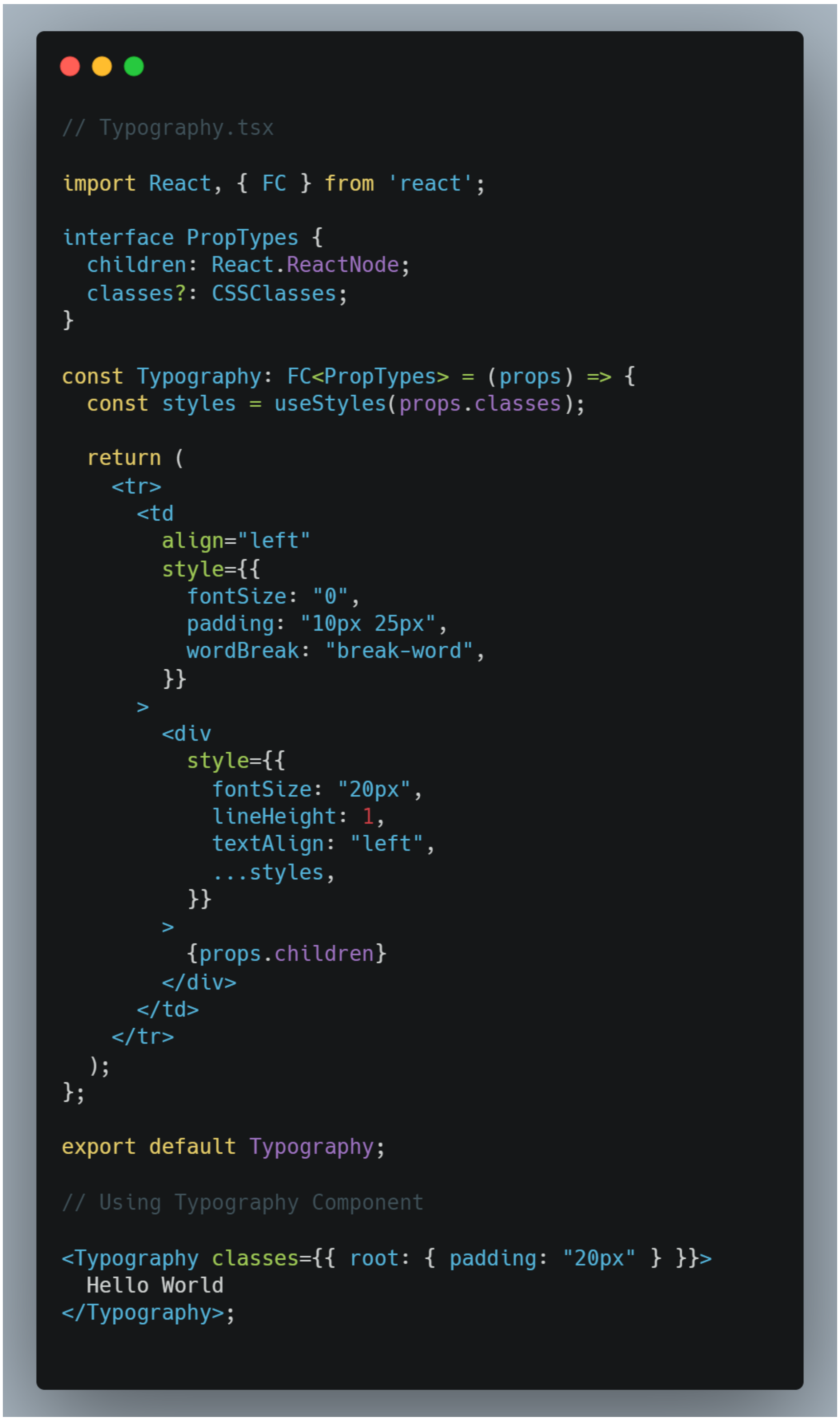
```
import { CSSProperties } from "react";
import merge from "lodash.merge";

export interface CSSClasses {
  [key: string]: CSSProperties;
}

const stylx = (...args: CSSClasses[]) => {
  const initialValue: CSSClasses = {};

  const result = args.reduce(
    (prev: CSSClasses, curr: CSSClasses) => merge(prev, curr),
    initialValue
  );

  return result;
};

<span style={stylx(styles.root, styles.dark)} />
<span style={stylx(styles.root, isCompact && syles.compact)} />
```

- “withStyles” hook will be used to add some default styling. And then it will expose another hook “useStyles” which we can use to expose and also to overwrite the styling



```
import { CSSProperties } from "react";
import merge from "lodash.merge";

interface CSSClasses {
  [key: string]: CSSProperties;
}

const withStyles = (classes: CSSClasses) => {
  return (classes_?: CSSClasses) => {
    return merge(classes, classes_);
  };
};

////////////////  Example //////////////////

const useStyles = withStyles({
  root: { margin: "10px", padding: "10px" },
});

const HelloEmail = ({ name, classes }: { name: string; classes: CSSClasses }) => {
  const styles = useStyles({ classes });

  return <span style={styles.root}>Hello {name}</span>;
};

const HelloWorldEmail = (
  <HelloEmail name="world" classes={{ root: { padding: "20px" } }} />
);
```

Example “Typography” component:



```
// Typography.tsx

import React, { FC } from 'react';

interface PropTypes {
  children: React.ReactNode;
  classes?: CSSClasses;
}

const Typography: FC<PropTypes> = (props) => {
  const styles = useStyles(props.classes);

  return (
    <tr>
      <td
        align="left"
        style={{
          fontSize: "0",
          padding: "10px 25px",
          wordBreak: "break-word",
        }}
      >
        <div
          style={{
            fontSize: "20px",
            lineHeight: 1,
            textAlign: "left",
            ...styles,
          }}
        >
          {props.children}
        </div>
      </td>
    </tr>
  );
};

export default Typography;

// Using Typography Component

<Typography classes={{ root: { padding: "20px" } }}>
  Hello World
</Typography>;
```

- **Styled and Unstyled Components**
    - We need to expose both Styled and Unstyled components for the package.

We need to provide a set of components without any styles. These can be used to implement a custom design system that is not based on our design patterns.

When to use Unstyled components:

- need to implement a design system that significantly differs from our design pattern
- need precise control over the rendered HTML of the components
- or need to implement custom components but don't want to deal with accessibility features on your own

When to use Styled components:

- need components that look good out of the box
- need to customize react-email design with your brand colors
- or need to implement a design system based on our design pattern

- **Adding Storybook:**
    - Need to set up Storybook and implement user stories for each component.
    - Add sample templates using the components in the Storybook.
    - Hosting the Storybook using GitHub Pages.

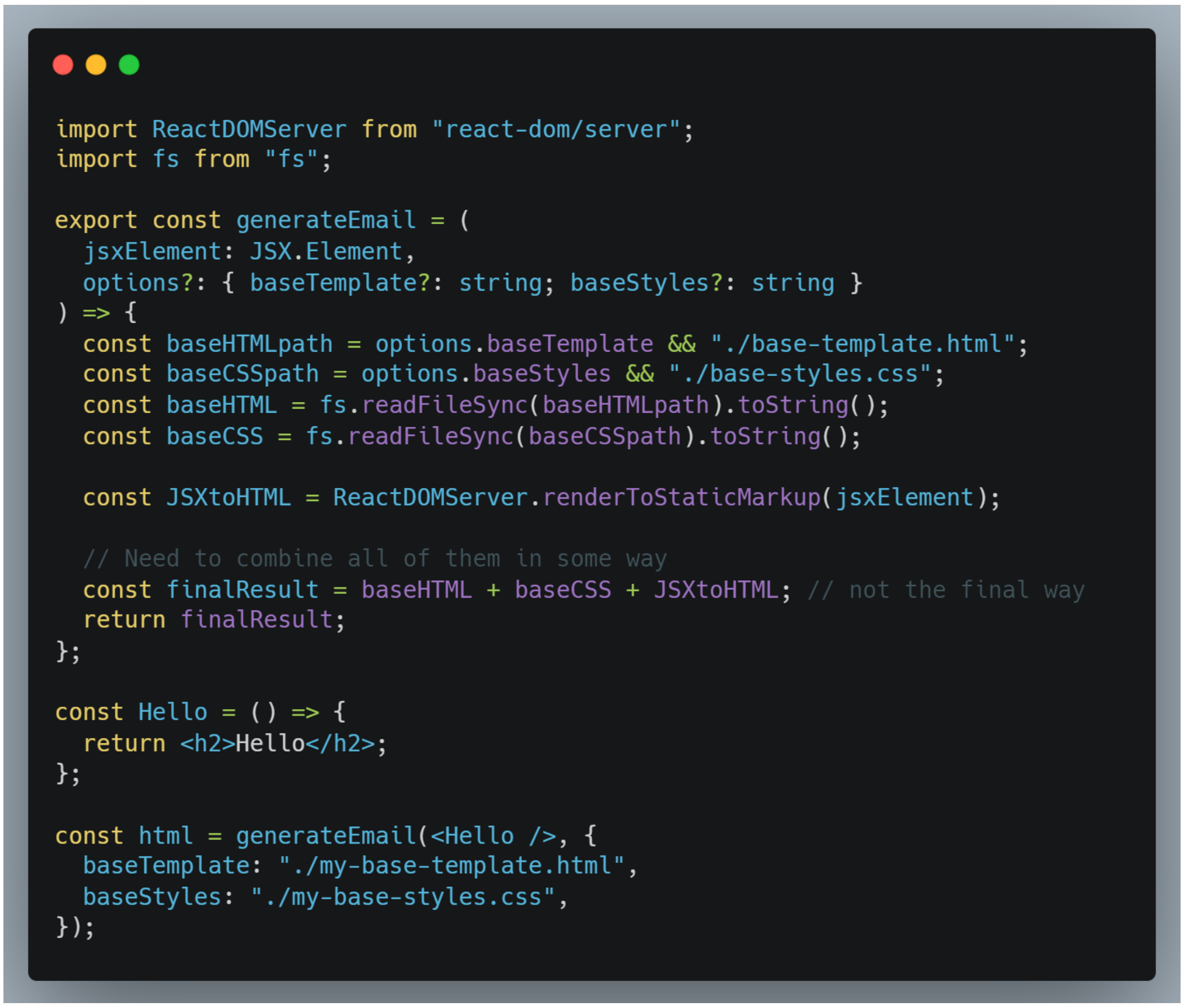
- **Testing with Jest:**
    - We are going to use “Jest” for our testing purpose to write test cases for component and utility methods.
    - Our main focus will be on “Snapshot Testing” which is widely used for the UI component libraries.

- **Converting React components to HTML:**

  After creating react components, we need to convert the react code to plain HTML code which users can send through email. We need to create the “generateEmail” utility function which will do this conversion.

  Now to convert the React component to plain HTML, we can use the “renderToStaticMarkup” method from ReactDOMServer in this way.

However, we also need to add the option to configure the base HTML and base CSS styles. Both of them should be added to the separate files and then should be imported. Users can also customize the base code as per their need.
If the users explicitly pass the “baseTemplate” and “baseStyles”, then the function will use those presets and generate the full email content. However, it is completely optional to pass own presets and in that case the function will use the default presets and will generate the email.

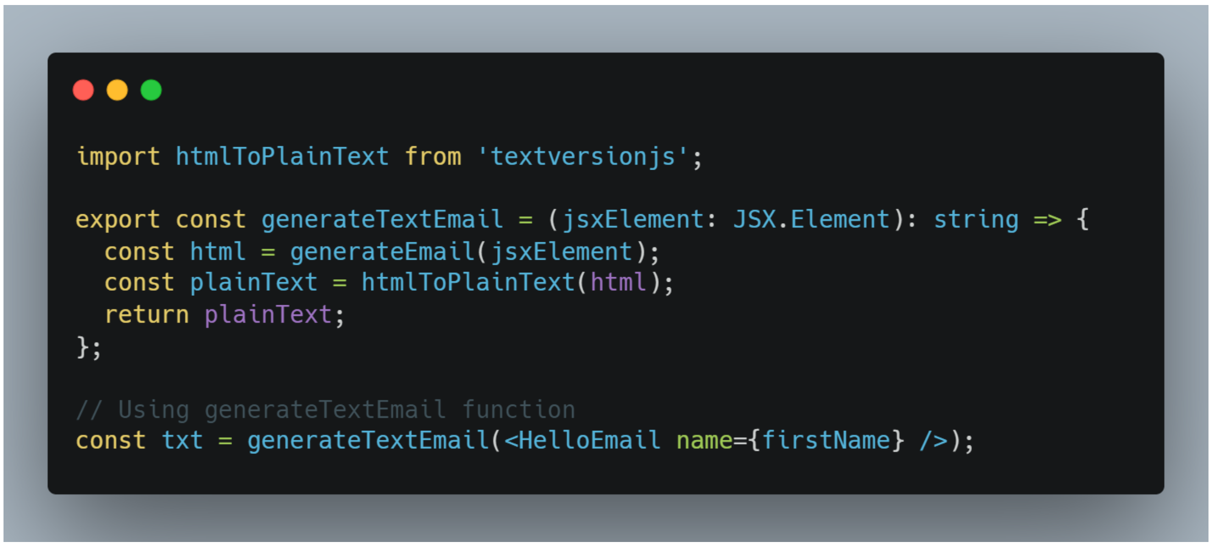


```
import ReactDOMServer from "react-dom/server";
import fs from "fs";

export const generateEmail = (
  jsxElement: JSX.Element,
  options?: { baseTemplate?: string; baseStyles?: string }
) => {
  const baseHTMLpath = options.baseTemplate && "./base-template.html";
  const baseCSSpath = options.baseStyles && "./base-styles.css";
  const baseHTML = fs.readFileSync(baseHTMLpath).toString();
  const baseCSS = fs.readFileSync(baseCSSpath).toString();

  const JSXtoHTML = ReactDOMServer.renderToStaticMarkup(jsxElement);

  // Need to combine all of them in some way
  const finalResult = baseHTML + baseCSS + JSXtoHTML; // not the final way
  return finalResult;
};

const Hello = () => {
  return <h2>Hello</h2>;
};

const html = generateEmail(<Hello />, {
  baseTemplate: "./my-base-template.html",
  baseStyles: "./my-base-styles.css",
});
```

- **Implement text email generation utility based on Email templates:**
  We also need to generate the plaintext version of the email template from the HTML or React.

  Implementation:
  We can use this package to generate text from the Template.
  https://www.npmjs.com/package/textversionjs

Demo:


```
import htmlToPlainText from 'textversionjs';

export const generateTextEmail = (jsxElement: JSX.Element): string => {
  const html = generateEmail(jsxElement);
  const plainText = htmlToPlainText(html);
  return plainText;
};

// Using generateTextEmail function
const txt = generateTextEmail(<HelloEmail name={firstName} />);
```

Sample output:
Hello World
![Image]
(https://images.unsplash.com/photo-1453728013993-6d66e9c9123a)
Hello World 1

- **Adding Email specific options**
  We also need to provide some email specific options to optimize the emails. Some of them are Preheader, CIDs, text quotes, etc.

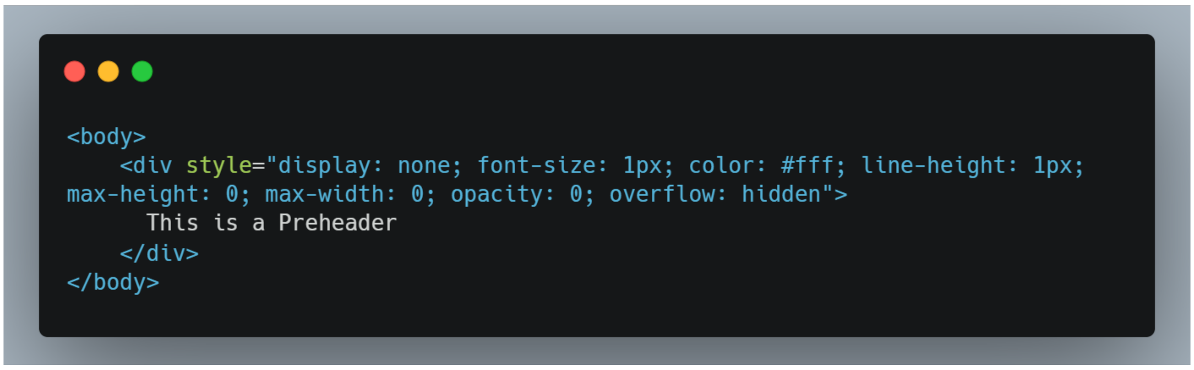
  - **Preheader**
    Email preheader text is a small line of text that appears after the subject line in an email inbox. Email preheaders give a short summary of the contents of an email, and may appear differently on mobile and web email clients.

    Adding Preheader:
    - Add text as the first text within the body tag of your email
    - Put the text in a div style
    - Use the div style to hide the text from the actual email

    To email clients, the preheader text will show up as the first text in the email. When someone opens the email, that text will be hidden (because of the hidden div style).

We can add Preheader as the props for our Email component or we can create a separate component.

Example to add Preheader:



```html
<body>
    <div style="display: none; font-size: 1px; color: #fff; line-height: 1px;
max-height: 0; max-width: 0; opacity: 0; overflow: hidden">
      This is a Preheader
    </div>
</body>
```

- **Adding Git hooks, linter, code formatter, CI-CD**
    - Set up CI/CD to auto-deploy the Storybook when a change is merged to the "main" branch (Preferably GitHub Actions)

      Few References:
      https://dev.to/kouts/deploy-storybook-to-github-pages-3bij

      https://medium.com/swlh/how-to-deploy-storybook-to-github-pages-4894097d49ab
    - We can also add a few additional things which will help us to maintain and scale up the package
        - Husky for initializing git hooks with lint-staged which will help us to maintain proper commit rules
        - ESLint for linting and Prettier for code formatting
        - Semantic-release for maintaining the version and release of the package
        - Since we are using GitHub to host our code, we can also use the power of GitHub Actions for the CI and CD of the project. We can automatically test, build and publish the package to npm using GitHub actions.

- **Validate the compatibility of the generated HTML content:**
  We are also searching for some free or paid (not too costly) 3rd party APIs that can help us to validate the compatibility of the generated HTML content across different browsers and clients.

  References:
  I did some deep research but I am unable to find such a strong and reliable service. Though there are lots of APIs to validate the email address.
  1. https://stackoverflow.com/questions/37291746/how-to-validate-html-email-templates
  2. https://stackoverflow.com/tags/html-email/info

  We can refer to the following documentation for possible support:
  1. https://mailtrap.docs.apiary.io/#reference/message/apiv1inboxesinboxidmessagesidspamreport/get
  2. https://mailtrap.docs.apiary.io/#reference/message/apiv1inboxesinboxidmessagesidanalyze/get

  Here are a few websites which do the same, but they don't have an API to use.
  1. https://www.htmlemailcheck.com/check
  2. https://www.google.com/webmasters/markup-tester
  3. https://www.pilotmail.io
  4. https://www.litmus.com
  5. https://www.emailonacid.com
  6. https://www.mailgun.com/email-testing-tool/html-email-tester

- **Documentation:**
  Proper documentation of the whole library is also important.
  We can add the whole documentation to the project README.md file or we can use any documentation site generator like Docusaurus to create a separate website for documentation purposes like Material UI

Timeline:

Period	Tasks
Pre GSoC Period *Till May 20*	● Research more on building and generating the components and utility functions ● See if there are any other better and optimized techniques to implement the features ● More research on the few missing things like validating the compatibility of the generated HTML content
Community Bonding *May 20 - June 12*	● Discussion on the project started ● Decide best approach for generating components and style overrides ● Discussion with the mentor if anything we are missing before the coding period ● Deciding the final APIs and libraries which we are going to use
Coding Phase 1 *June 13 - July 25*	● Selecting the module bundler and project initialization ● Adding Git hooks, linter, code formatter, CI-CD ● Docs initialization ● Creating initial components ● Adding Storybook and user stories for each component, hosting Storybook with GitHub Pages ● Testing with Jest, setup Jest for snapshot testing and writing tests for the components and functions ● Adding styling methods ● Converting React components to HTML ● Implement text email generation utility based on Email templates ● Updating tests and documentation accordingly ● Bug fixes
Phase 1 Evaluation *July 25-July 29*	● Complete the backlogs (if any) ● Writing a blog post about my work and experience working on the project
Coding Phase 2 *July 25 - Sept 4*	● Adding more components ● Advanced styling of the components, “withStyles” hook to add style overrides,

	expose both styled and unstyled components ● Adding more tests for the components ● Update documentation accordingly ● Adding Sample templates using the components in Storybook ● Validate the compatibility of the generated HTML content ● Fixing bugs ● Release the beta version of the package (v1.0.0) on npm ● Migrating the documentation to a separate site using site generator (if time permits)
Final Week *Sept 5 - Sept 12*	● Writing another blog post about my overall work and experience working on the whole project ● Also will write a final report on my overall work
Final Mentor Evaluation *Sept 12 - Sept 19*	● Mentors submit final GSoC contributor evaluations
Results *Sept 20*	● Initial results of Google Summer of Code 2022 announced

SCoRe Contributions:

Despite not having any prior contributions to SCoRe Lab, I have been closely connected to the Project Mentor and discussed the specifics in-depth and the potential of the project during the ideation period (pre-GSoC period). I highly look forward to making my suggested contributions to SCoRe Lab.

Personal Information:

Name: Niloy Sikdar
Email: niloysikdar30@gmail.com
GitHub Username: niloysikdar
LinkedIn: https://www.linkedin.com/in/niloysikdar
Contact No. : +916297215051
Time Zone: Indian Standard Time (IST) (UTC +05:30)

Educational Details:

College: Jalpaiguri Government Engineering College
Year: 3rd
Major: Computer Science and Engineering

Projects related details:

Have you tried that project you selected from SCoRe project list:
Yes, after several rounds of discussions with the project mentor I finally tried to implement a basic starter code for the project which can be also used as a reference to start. As the project is going to start from scratch, it is also important to decide the efficient and optimized approach for the same.

Here is my sample code: https://github.com/niloysikdar/react-email-scorelab

What problems, if any, were presented?
There were several problems and issues related to the project from selecting the 3rd party libraries to the implementation part. But I tried my best to clearly communicate with the mentor to get things started correctly.

What prevented you from getting the entire system up and running?
Nothing. Setup and Run were done smoothly.

Availability (List down any plans you have during this summer):

I intentionally freed my summer slot for GSoC and didn't take any internships for that period. So, there wouldn't be much disturbance/inactive days while working on my project. I do have my offline classes, but that won't cause any issues as I will be available to dedicate 25-30 hours/week to the project. If the college gets closed for some reason, then the amount of available time will increase.

Post GSoC plans:

Once the GSoC finishes, I will look into the issues and the PRs raised by the other people and will also look into new issues. Adding some more features and making the package more robust and feature-rich will be my primary goal. I want to maintain the package for the long term and want to become a lifelong member of the SCoRe community to help people in the field of open-source.
Also, it was my wish to release my own blog site, but for some reason, it wasn't happening for a long time. I will try to code my own blog site and write some blogs about my overall GSoC experience and about the SCoRe community and the projects.

What programming courses have you taken?

I haven't taken any such programming courses as I am mostly a self-taught programmer. Mostly, I took help from the free resources like YouTube, Documentation, Blog posts, articles, and StackOverflow.

Have you done group projects (programming or otherwise)?

Yes, I have participated in several group projects. Most of them are during hackathons.

A few of them are:

- **GuruCool**
  Tech Stack: React, Redux, Sass, Node.js, Express, MongoDB
  GitHub | Live Link | Demo Video
  - ➔ Progressive Web Application with moderation functionality for Teachers to keep the track of Students' activity
  - ➔ Role-based Sign Up/login system with JWT authentication, having complete Classroom Management features
  - ➔ Gamified approach to learning, with a Point-Based Quiz System with leaderboard and leveling up options
  - ➔ Made it to the top 20 in a national hackathon organizer by Scaler Academy

- **Sukoon**
  Tech Stack: React, CSS, Dialogflow
  GitHub | Live Link | Demo Video
  - ➔ Created a Web App using React to help people suffering from mental health issues and make them feel better
  - ➔ Implemented SAWO Labs API for Authentication with customized feed, activities, and Chatbot
  - ➔ Awarded with Best of HackOn & Best use of SAWO among 370+ teams for our solution to help people suffering from mental health issues in HackOn 2.0, an international hackathon

- **Plaso Connect**
  Tech Stack: Flutter, Firebase, Firestore
  GitHub | Releases | Demo Video
  - ➔ Created a mobile application as a one-stop solution with role-based system where the people requiring blood plasma and oxygen can directly find and contact the donors and healthcare units as per their requirements
  - ➔ Integrated Covid API for fetching realtime data, Firestore CRUD operations, animations and page transitions
  - ➔ **Got showcased in the Google's Dev Library under the Flutter Category**
  - ➔ Winner in Health and Safety Track among 3000+ participants in Equinox'21, a national level Hackathon

- There are lots of other open source projects available on my GitHub

What was your primary contribution to/role in the group?

Mostly, I was the team leader for the projects and hackathons. I managed the whole group consisting of 3-4 people and pushed everyone to perform their best. Also leading the coding part from the front with the team is something that I am proud of.

Do you have work experience in programming? Tell us about it.

Yes, I do have work experience in programming. I have worked for a few startups to create and scale up their creative products. Few of my experiences are:

- **Technical Lead | TidyForm | Apr, 2022 - Present**
  - Leading the Technical team of 4 members and also focusing on the overall product decisions to solve the problem of building brandable forms easily using Tidyform

- **DSA and Full Stack Mentor | The 10x Academy | Apr, 2022 - Present**
    - Taking Practice Sessions, Group Discussions and 1:1 mentorship session for the students in the domain of DSA and Full Stack Development

- **Software Developer Intern | SAWO Labs | Jul, 2021 – Sep, 2021**
    - Increased the pub points from 80/130 to 130/130 of the sawo flutter package with additional improvements
    - Revamped the whole UI of the website using React and Tailwind CSS with a Lighthouse score of 96
    - Fixed bugs, added new reusable components to the Client Dashboard and WebSDK using React and Redux
    - Built SAWO sample apps from scratch using React, Vue.js, Node.js and React Native

- **Flutter Developer | Fivefalcon Pvt. Ltd. | Mar, 2021 – Apr, 2021**
    - Developed the interactive and animated UI from scratch of a booking app named Extacy, using Flutter
    - Added Phone, Email and Google authentication using Firebase and Backend features like adding clubs, bars, and parties from Admin side and showing them on the Client side using Firestore CRUD operations

- **Mobile Application Developer**
    - Designed, Coded and Published 3 Android Applications on Google Play Store in Google Play Store using different technologies like Flutter, Apache Cordova, etc.

Few sentences about the student and why he/she thinks as the best person to do this project.

I always like to think out of the box, that's why I always try to focus on building something exciting to solve real-world problems using different technologies. The email-related problem which we are currently facing is something serious and we need to find some optimized solution for this.

- The project is going to use React and TypeScript for the development, and I have worked on some production level React code with some experience with TypeScript as well.

- I have already created one react package using TypeScript before- react-awesome-audio. So I have a very clear understanding of the whole workflow of creating react packages.
- I already got a clear understanding of the whole project and its requirements after a long discussion with the project mentor. I made detailed and clear research on each and every point provided by the mentor to add proper inputs for creating the project.
- I have already created a basic starter code example with the feedback and inputs provided by the mentor.
- Creating exciting and scalable things which can solve problems for a lot of users and create a large impact is something that I have always liked and I have been doing it on a regular basis.
- Lastly, I am really excited to work on this awesome project this summer under the guidance of mentors and the community.

Do you have previous open source experience? Briefly describe what you have done.

I have been a part of and closely attached to different open source events and communities. I started my programming career at one open-source event named JGEC Winter of Code '21 and I became the top contributor among 600+ participants with 41 successful pull requests in one month by contributing to different projects. I have also mentored lots of students and beginners to start their open source journey by conducting different sessions and participating as a maintainer and mentor (LetsGrowMore Summer of Code, GirlScript Education Outreach, Hacktoberfest 2021). Also, recently I was also one of the lead organizers of JGEC Winter of Code '22 which helps students to plunge into Open Source contribution and the realm of Software Development.

Few of my contributions to large organizations:

- **Google, iree #6568 , Pull Request**
    - Run a check_submodule_init script during CMake configure
    - Checking if git submodules have been initialized and printing warning messages
    - Added a python script in scripts/git/check_submodule_init.py which will run inside the CMake: CMakeLists.txt

- **Google, iree #6575 , Raised Issue and suggestions**
    - Optimizing the check of git submodule status
    - We were running a check_submodule_init.py script during CMake configure
    - It was taking ~2 seconds to run, we could use caching the result and only checking once the git commit changes

- **Fossasia, open-event-next #309, Pull Request**
    - Created dynamic events pages using Next.js and TypeScript

Tell one interesting fact about yourself

I LOVE to teach and empower students and beginners to create solutions using tech. That's why I took the initiative and became the first-ever Google Developer Student Club Lead of my College (selected by Google) to empower and spread a collaborative atmosphere while learning about new technologies. Apart from programming, I am a foodie and keen traveler with a taste for Bollywood music.