SORT

Sort uses the React framework to build a PWA (progressive web app). If you are not familiar with React please read up on the subject before continuing. Knowing the basics of HTML, CSS, and Javascript is also required to be able to understand what has been done in this project.

Motivation

We decided to make a progressive web app since it allows for good usability in areas where internet conection is spoty. It is for this reason that we chose to use React to build our website. Bootstrap is being used to build a reactive design that will allow the website to look good on any device size. We want SORT to work on any device from a small cell phone to a huge computer screen.

Getting started

Our main website component is called *index.js* within this JS we render our single object which is situated in *App.js*. It is within *App.js* that most of the code for the webiste resides.

Our `App.js` is in charge of keeping track of what questions where answered and rendering the appropriate questions. A question in this project is an object that represents a question that the user must awnser, or information about how to dispose of the waste as well as rendering the form to fill in to create the label. `App.js` simply take care of rendering the website in the correct language, dealing with the history of previously awnsered questions, and redering new questions/information once I user has anwsered them.

`Question.js` get's all the information that it needs from `App.js` and renders the question or *leaf (information on how to dispose of the waste).*

How to modify stuff:

For modifications please set up your enviroment by following the instructions on https://reactjs.org/docs/create-a-new-react-app.html. Once this is done you should be able to run `npm start` and have a server launch on localhost:3000 displaying you the current version of the website.

Decision tree and other webiste text

The decision tree is located in `./data/treeEN.json` or `./data/treeFR.json` depending on what language you wish to modify. Do know more on how to edit

this please read the `README.md` in the `./data/` folder. Click here to go there now

Adding an image to the more information pop up

This is a multistep process. Please follow it carefully.

1. Export the image that you wish to add as a **png**. If the image is displayed on the website is to large please resize the image.
2. Get a base64 version of the image. Sounds scarry, it isn't. It is a simple way used in websites to store images as text. We would recommend that you use a website to convert your image to base64 such as https://www.base64-image.de/
3. Paste the base64 representation in the `./image/images.json` with an previously ususued key.
4. Add the key value to the `treeEN.json` and `treeFR.json` for the corresponding question under the `"moreInfoPicture" : "THE NUMBER OF THE IMAGE IN images.json",` field

Adding an image to the information on how to dispose of waste (leaf)

The procedure is the same as for the information popup. The only step that differes is step 4.

1. Follow steps 1 to 3 of **Adding an image to the more information popup**
2. Add the key value to the `treeEN.json` and `treeFR.json` for the corresponding question (leaf) under the `"picture" : "THE NUMBER OF THE IMAGE IN images.json",` field

Deleting an image

The procedure described is the same for both the pop up and the images displayes in the leaves (area where detailed instructions on the disposal of waste are given).

1. Find the number of the image that you wish to remove in the `treeXX.json` file.
2. Remove it and replace it with `""`. For example `"picture": ""`
3. If no other questions or information (leaves) are referencing the image number you can go into the `./images/images.json` and remove the entry corresponding tp the image.

Modified required fields

This is super easy but requires actually modifying the code. Go into the `./labels` directory open the label that you want to modify. In the render function you will see the different elements of the form and you can add or remove the required parameter for each field of the form.

Publishing it to the server

Getting the website published is supper easy if you are comfortable using the linux terminal.

0. First thing to do is to check that your website is OK! To do so run `npm start` in the terminal to launch a server to allow you to view the content of your website. If you get no errors and the modifications that you have made look OK then proceed to step 1.
1. Start by running the `npm build` command to prepare the deployable version of the code. This should create a `build` directory where all the content for the website is ready to be deployed.
2. Use scp to copy the files in the build direcory onto the server. The IP of the server is 10.95.32.14 and the username is `webuser`. And the password.... Well you should know it, or at least if you don't and you think that you should please contact dylan.portmann@epfl.ch or sebastian.bruckner@epfl.ch. They should know how to give you access. I wasn't going to put all the credentials into the documentation in case somebody that shouldn't be poking around gets his hands on this.
3. Place the files that you just copied into the `/var/www/html/` folder.

That's it! You have deployed or update the website. Congratulation!

Folder structure

Data used for the website is located in `./data` . To have more information about the files in this direcotry please click here

Images are stored in the `./images` folder. To have more information about the files in this directory please click here

Labels are sotred in the `./labels` folder. To have more information about the files in this directory please click here