Developing and Collaborating on React JS Apps

React JS

CS317 Mobile App Development

Lyn Turbak

Department of Computer Science
Wellesley College

Plan for Today and Friday

CodeSandbox is suitable for simple React JS examples, but is limited for more complex ones. Today you will learn how to develop React JS apps on your own laptop and collaborate with others on these apps. In particular:

Today you will learn how to:

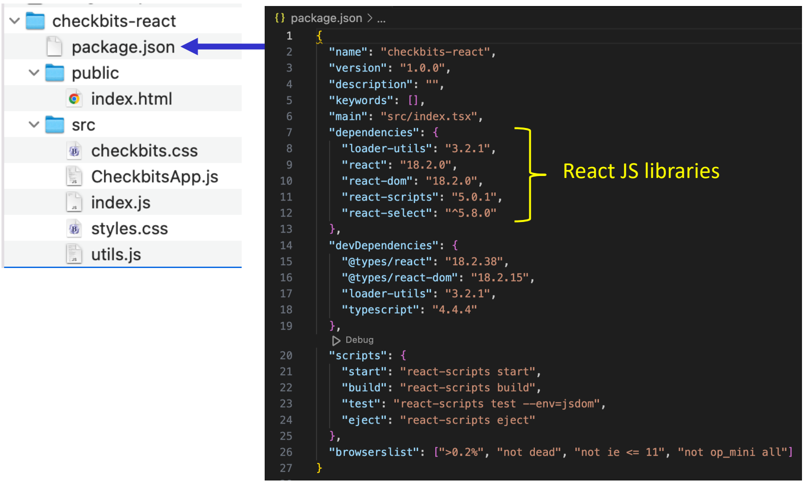
- Install and run a React JS app from a .zip file (such as one downloaded from Code Sandbox).
- Use `npm create-react-app` (CRA) to create develop a React JS app from scratch on your laptop.

On **Friday**, you will learn how to:

- Use git to make a repo and commits to repos
- Install and run a read-only React JS app from GitHub using git
- Use git together with GitHub to collaborate on program development.
- Collaborate with partners on a React JS app in VS Code

Installing a React JS app from a .zip file (Part 1)

1. Download the checkbit-react.zip app linked from today's course schedule entry.
2. Unzip it and study its structure:



```
checkbits-react
  package.json
  public
    index.html
  src
    checkbits.css
    CheckbitsApp.js
    index.js
    styles.css
    utils.js
```

{} package.json > ...

```
{
  "name": "checkbits-react",
  "version": "1.0.0",
  "description": "",
  "keywords": [],
  "main": "src/index.tsx",
  "dependencies": {
    "loader-utils": "3.2.1",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-scripts": "5.0.1",
    "react-select": "^5.8.0"
  },
  "devDependencies": {
    "@types/react": "18.2.38",
    "@types/react-dom": "18.2.15",
    "loader-utils": "3.2.1",
    "typescript": "4.4.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  },
  "browserslist": [">0.2%", "not dead", "not ie <= 11", "not op_mini all"]
}
```

Installing a React JS app from a .zip file (Part 2)

You cannot run the checkbits-react app yet because it is missing hundreds of library directories require by React JS. Follow the steps below to create a `node_modules` subdirectory.

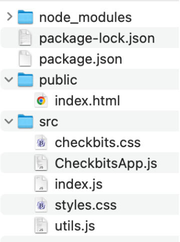
1. cd to the unzipped `checkbits-react` directory in your `Downloads` directory.
2. In a Terminal window (Mac) or Powershell (Windows) execute

   ```
   npm install
   ```

   This may take a minute or two to install all the necessary files, which are determined from `package.json`.

Installing a React JS app from a .zip file (Part 3)

Study the structure of the `checkbits-react` directory after the `npm install`.



The `npm_modules` directory takes a **huge** amount of space, which is why it's not included in .zip files and git repositories:

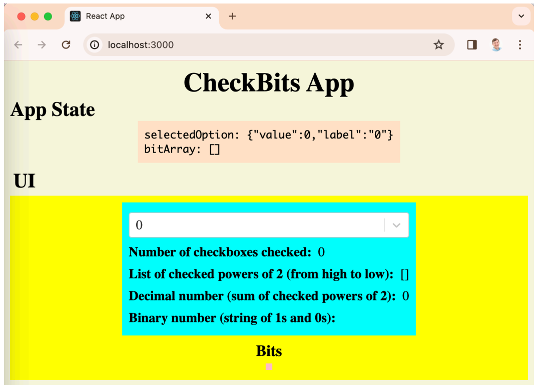
```
[fturbak@LynAirbook:checkbits-react]$ pwd
/Users/fturbak/Downloads/checkbits-react
[fturbak@LynAirbook:checkbits-react]$ du -h -d 1
337M     ./node_modules
4.0K     ./public
 32K     ./src
338M     .
```

Installing a React JS app from a .zip file (Part 4)

From the `checkbits-react` directory, run the app by executing

`npm start`

It may take a while before the app launches in a browser window.



CheckBits App

App State

```
selectedOption: {"value":0,"label":"0"}
bitArray: []
```

UI

0

Number of checkboxes checked: 0

List of checked powers of 2 (from high to low): []

Decimal number (sum of checked powers of 2): 0

Binary number (string of 1s and 0s):

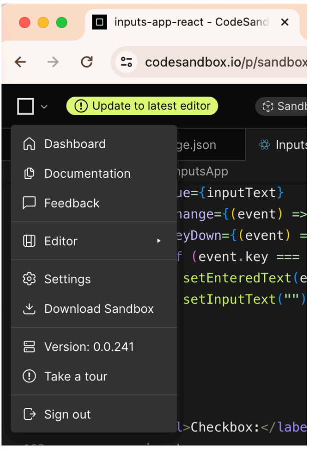
Bits

If you edit CheckbitsApp.js (in VSCode) and save the file, you'll see the changes immediately reflected in the browser! This has the interactivity of CodesSandbox, but on your laptop.

Installing a React JS app from a .zip file (Part 5)

You can download a .zip file from any CodeSandbox React JS sandbox and then install and run the app on your laptop. Practice this using the inputs-app-react sandbox linked from today's lecture.

Click here to download a .zip file for the sandbox



```
ue={inputText}
nange={(event) =>
eyDown={(event) =
f (event.key ===
 setEnteredText(e
 setInputText("")
l>Checkbox:</labe
```

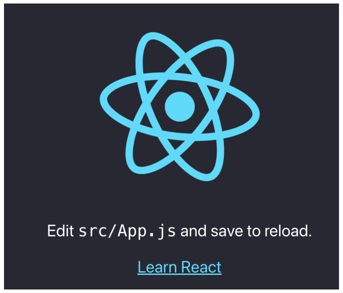
`create-react-app (CRA):` Create a React JS app from scratch

You should have already installed the Node Version Manager (`nvm`), and used it to install `node.js`

node.js comes with the Node Package Manager (`npm`) & Node Package eXecuter (`npx`). Use the latter in a shell as follows to create a React project named `counter-app`.

```
npx create-react-app counter-app
```

- Executing this command can take a long time (can be several minutes) as it fetches a huge number of modules modules for the node_modules subdirectory.
- The resulting project folder (`counter-app`) is large (> 300MB), but the parts that will be put in a git repo are much smaller (because they will exclude the large node_modules subdirectory).
- `cd` into `counter-app` and run the project via `npm start`. This will show the default app (a rotating React JS icon) in a new browser window for `localhost:3000`.



Edit `src/App.js` and save to reload.

Learn React

Changing the default app

- If you edit `src/App.js` (e.g, change the text displayed for the rotating icon), the displayed app will automatically update!
- Here’s a much bigger change:
  - Using VS Code, in `counter-app/src`, create a new file CounterApp.js, and copy the contents of CounterApp.js from Lyn’s CodeSandbox Counter App (`https://2mmwm4.csb.app/`) into this new file.
  - In `counter-app/src`, create a new file styles.css, and copy the contents of styles.js from Lyn’s CodeSandbox Counter App (`https://2mmwm4.csb.app/`) into this new file.
  - Edit `counter-app/src/index.js` to change

    ```
    import App from './App';
    ```

    to

    ```
    import App from './CounterApp’
    ```

  - The displayed App should change to show the Counter App!

git for version control & collaboration

- git is a popular version control system
  - Can use it by yourself to manage different versions of projects
  - Can use it to collaborate on projects with others
- Many of you have used git in courses like CS240 and CS304, but it’s OK if you’ve never used it before.
- We’ll focus on a few standard `git` commands. For more details, see git links in CS317 schedule.
- The `git` shell command is pre-installed on Macs, but not Windows. See windows installation details on the next page
- Ask me for help if you have trouble installing or using git!

Installing git on Windows

git is not pre-installed in Windows, so you’ll need to install it.

To to this, download and run the installer from https://gitforwindows.org

During the installation, you should accept most of the default setup options **except** for the following:

- For **Choosing the default editor used by Git**, in the dropdown menu, choose **Use Visual Studio Code as Git’s default editor.**
- For **Adjusting the name of the initial branch in new repositories**, select **Override the default branch name for new repositories** (and keep **main** as the override).
- For **Adjusting your PATH environment**, choose **Git from the command line and also from 3rd party software**

Creating a git repo

- npx create-react-app *appDirectoryName* makes *appDirectoryName* a git repo. How can we tell? It contains a `.git` directory (normally hidden).
- We’ll see later that the `git clone` command also creates a git repo.
- To create your own git repo from scratch:
  - *Option 1:*

    ```
    cd existingDirectoryToMakeARepo
    git init
    ```

  - *Option 2:*

    ```
    git init nameForNewRepoDirectory
    ```

git commit model & git log

In the git model of version control, you declare checkpoints/snapshots in your development process called **commits**, where each commit bundles together some recent changes on top of a previous commit.

The git log command shows the commit history of your project. Each commit entry includes a long alphanumeric fingerprint identifying the commit, the author of the commit, the timestamp of the commit, and a message explaining that commit. For example, `counter-app` has one commit when it starts:

```
[fturbak@LynAirbook:counter-app (main)]$ git log
commit 106bb51b46eacdafe075f075557a398289d7d465 (HEAD -> main)
Author: Franklyn Turbak fturbak@wellesley.edu
Date:   Tue Mar 5 10:46:28 2024 -0500

    Initialize project using Create React App
```

git status: staged, changed, and untracked files

git tracks which files have changed since the last commit. Because the programmer may wish to include only some of these in the next commit, git allows the programmer to specify which changed/new files are planned to be committed by adding them them into a **staging area.**

The git status command indicates which files are

1. staged,
2. changed but not staged, or
3. untracked (i.e., never staged for a previous commit, which can include new files).

E.g., after making the modifications in slide 9 on your new counter-app git status will report something like the following:

```
[fturbak@LynAirbook:counter-app (main)]$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/App.js
    modified:   src/index.js

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/CounterApp.js
    src/styles.css
```

git add stages a file

The git add command adds a changed or untracked file to the staging area for a commit

In our running example, if you want `src/CounterApp.js`, `src/styles.css`, and `src/index.js` to be part of the next commit, you would execute:

```
git add src/CounterApp.js src/styles.css src/index.js
```

After executing the above command, the first two files have been moved from the untracked area to the staged (to-be-commited) area, and the last file has been moved from the changed-but-not-staged area to the staged area.

```
[fturbak@LynAirbook:counter-app (main)]$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
     new file:   src/CounterApp.js
     modified:   src/index.js
     new file:   src/styles.css

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
     modified:   src/App.js
```

git commit makes a commit with the staged files

git commit bundles the staged files into a new commit that has its own alphanumeric fingerprint along with the author of the commit and the current timestamp. If the **–m** optional argument is provided followed by a string, that string becomes the commit message. Otherwise, and editor is launched in which the commit message should be typed.

```
[fturbak@LynAirbook:counter-app (main)]$ git commit -m "Converted default react app
into counter app."
[main a59b41b] Converted default react app into counter app.
 3 files changed, 96 insertions(+), 1 deletion(-)
 create mode 100644 src/CounterApp.js
 create mode 100644 src/styles.css

[fturbak@LynAirbook:counter-app (main)]$ git log
commit a59b41b8df445cd4dde3408295a64fa4e3a71a24 (HEAD -> main)
Author: Franklyn Turbak fturbak@wellesley.edu
Date:   Fri Mar 8 10:31:15 2024 -0500

   Converted default react app into counter app.

commit 8716c814733a7a83014f7198c26635bd30c2bf9b
Author: Franklyn Turbak fturbak@wellesley.edu
Date:   Tue Mar 5 12:22:55 2024 -0500
   Initialize project using Create React App
```

GitHub for sharing and collaboration

- GitHub is a company that hosts public and private git repositories. We'll use it in this class for sharing and collaborating on projects.
- As part of HW0, you should have created a github account along with (1) a way to access it with ssh and a two-factor authentication mechanism.

Creating a github repo

Suppose you want to share your changes to inputs-app with others.
You can do this by creating a public repo on github, and "pushing" the version on your laptop to the github repo.

Follow these steps to create the repo:

1. In your browser, visit https://github.com/yourGitUserName
2. Navigate to **Repositories** and click **New**
3. Choose the repository name. In order to avoid confusion, name the app yourTempestUserName-counter-app. E.g. gdome-counter-app
4. If you want to share it with the world, make the repo **public**; if you want to limit sharing, make it **private**.
   - Make this counter app **public**
   - Make all homework repos **private**!
5. Click **Create Repository** to create the repo.
6. In the **Add collaborators to this repository** area, click **Invite Collaborators** and add the github username of your Project 3 partner (or whomever your working with in class today.) Your partner will need to accept the generated email invitation.

Adding a remote

In order to share your local counter-app with others, you'll need to "push" it to the github repo you just created.

But in order to perform the push, you'll need to tell your local git the location of your github repo. You do this via `git remote add`

```
[fturbak@LynAirbook:counter-app (main)]$ git remote add mygithub
git@github.com:fturbak/fturbak-counter-app.git
```

Here `mygithub` is the name of the remote. You can choose any name you like. You can have multiple named remotes associated with a local git repository.

You can list the remotes of your local git repo via `git remote -v`

```
fturbak@LynAirbook:counter-app (main)]$ git remote –v
mygithub  git@github.com:fturbak/fturbak-counter-app.git (fetch)
mygithub  git@github.com:fturbak/fturbak-counter-app.git (push)
```

Pushing your local repo to github

Once the gihub repo is created and you added it as one of the named remote repos, you can upload your local repo to the github repo by "pushing" it via git push remote branch, where remote is the name of the remote repository and branch is the repo branch you want to push.

You can list your branches via git branch –a. The main branch is usually called main or master; you can change master to main via git branch -M main

```
[fturbak@LynAirbook:counter-app (main)]$ git push mygithub main
Enumerating objects: 28, done.
Counting objects: 100% (28/28), done.
Delta compression using up to 8 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (28/28), 181.57 KiB | 5.19 MiB/s, done.
Total 28 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To github.com:fturbak/fturbak-counter-app.git
* [new branch]      main -> main
```

After this push, in your github repo, you will be able to see the details of your counter app code in the a web browser connected to your github account.

Adding collaborators to a github repo

By default, no other github users can push to your repos. But when working in a small team, you may want your team members to be able to directly push to a shared repo so that you can more easily work together to edit the same code.

In github parlance, a **collaborator** on a repo owned by you is another github user with direct push and pull access to your repo. (Public repos can have any number of collaborators, but private repos can have at most 3 collaborators under the default free plan.)

You should have already added a collaborator when you created the counter app repo, but if you forgot to do it then, you can do it now. Follow these steps to add a collaborator to a github repo you own:

1. On your repo page, click **Settings**
2. In the left hand navigation menu, click **Manage Access**. (You may be asked to confirm your github password in the browser.)
3. Click the green Add People button, and search for your collaborator's github username. Click **Invite Collaborator** for this user. **Warning!** This does not actually add the user as a collaborator; it just sends them an invitation email. They must accept the invitation in the email to actually become a collaborator.
4. Once the collaborator accepts the invitation, they can (1) clone your repo and (2) add your repo as a remote.