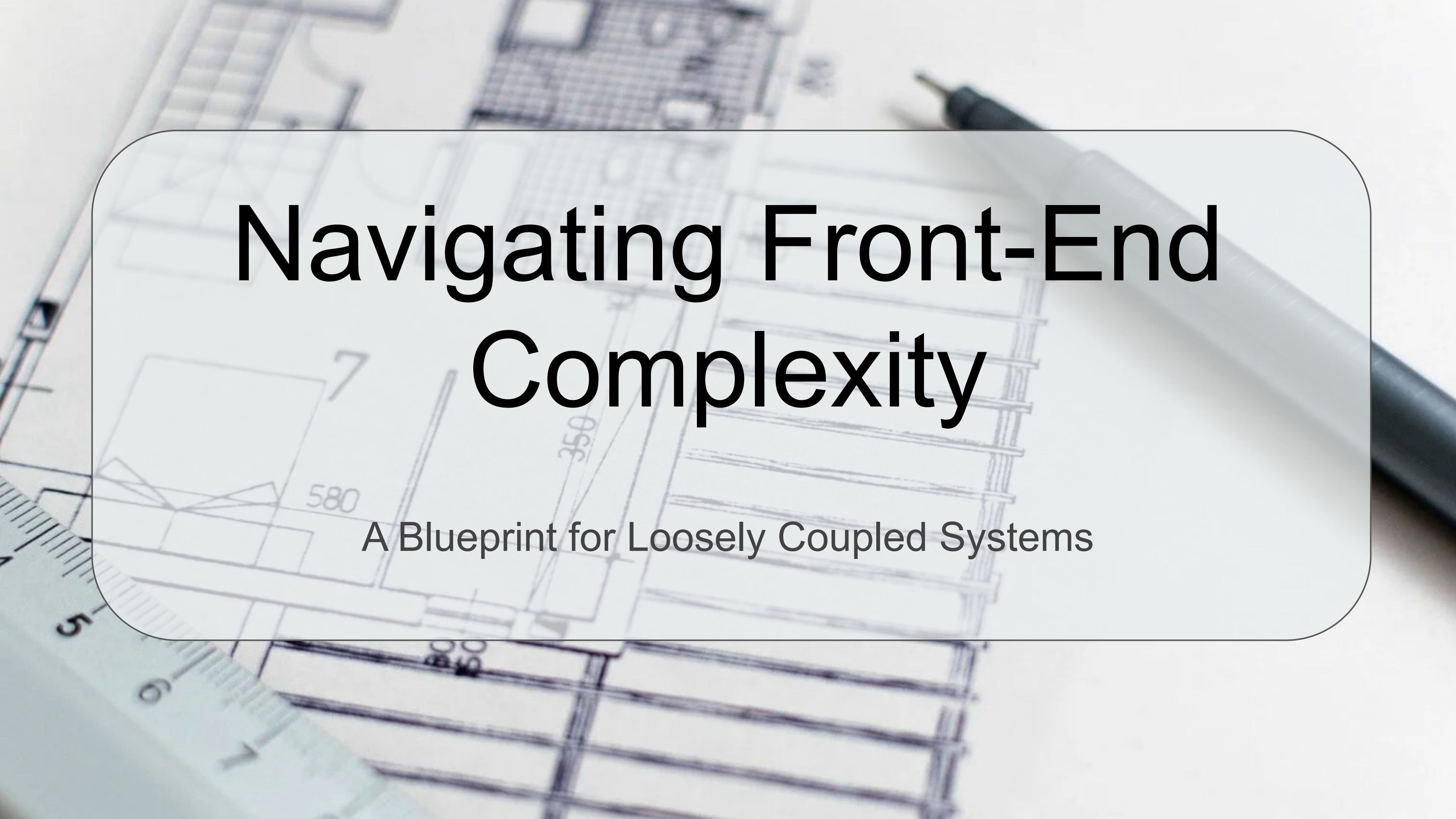


Navigating Front-End Complexity

A Blueprint for Loosely Coupled Systems

@Chad_R_Stewart Chad Stewart

Chad Stewart

A Software Engineer from Kingston, Jamaica. Been coding for over 10 years. Have worked at Enterprise orgs, startups and on open source projects.

Also **the Founder** of

TechIsHiring

@Chad_R_Stewart Chad Stewart

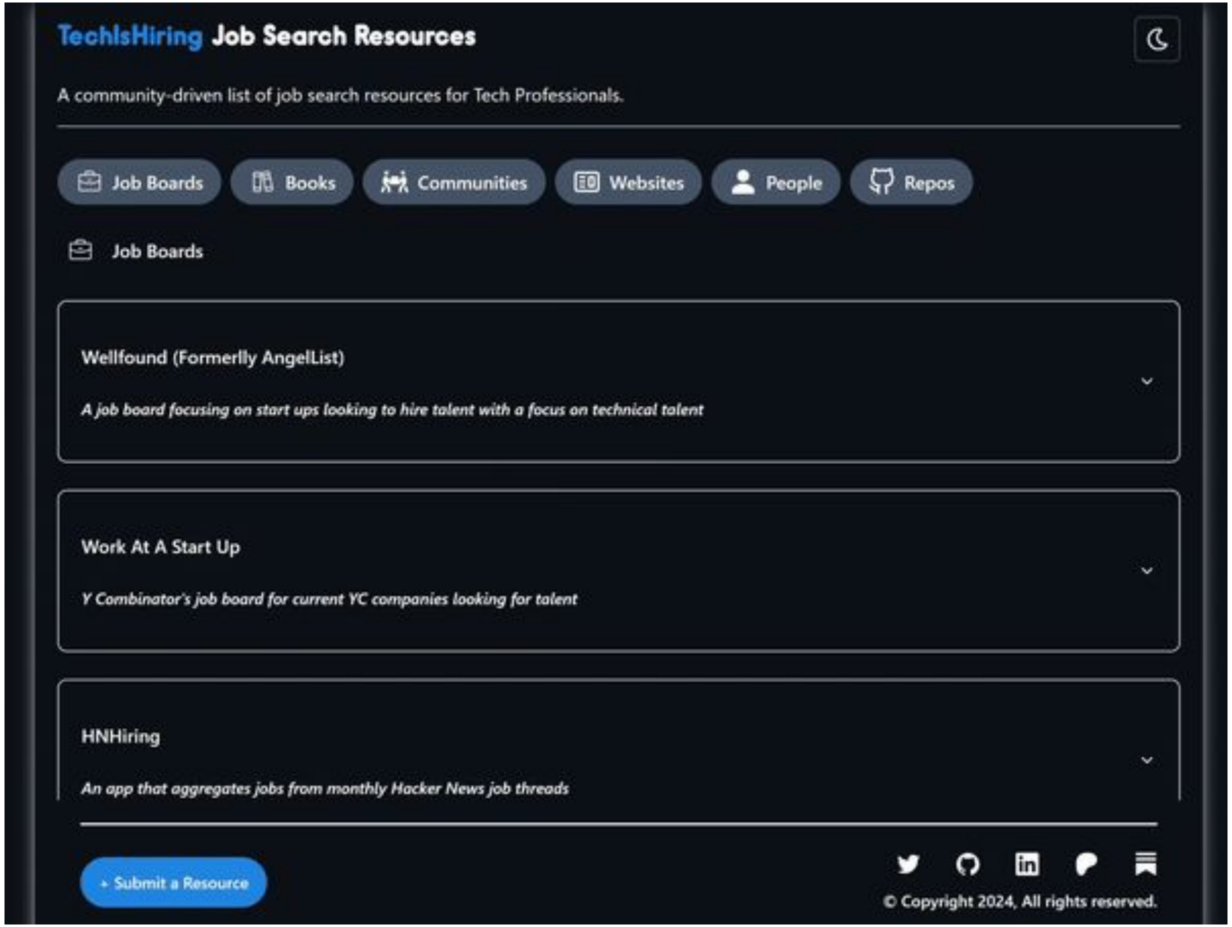
Front-End Complexity Growth is Sneaky

So what options do we have to deal with Front-End Complexity?

Encourage Loosely-Coupled & Modular Code

Makes it easier to reason about your system and gives you options when modifying & extending your codebase

How are we going to explain how to do to this?


TechIsHiring Job Search Resources

A community-driven list of job search resources for Tech Professionals.

Job Boards | Books | Communities | Websites | People | Repos

Job Boards

Wellfound (Formerlly AngelList)

A job board focusing on start ups looking to hire talent with a focus on technical talent

Work At A Start Up

Y Combinator's job board for current YC companies looking for talent

HNHiring

An app that aggregates jobs from monthly Hacker News job threads

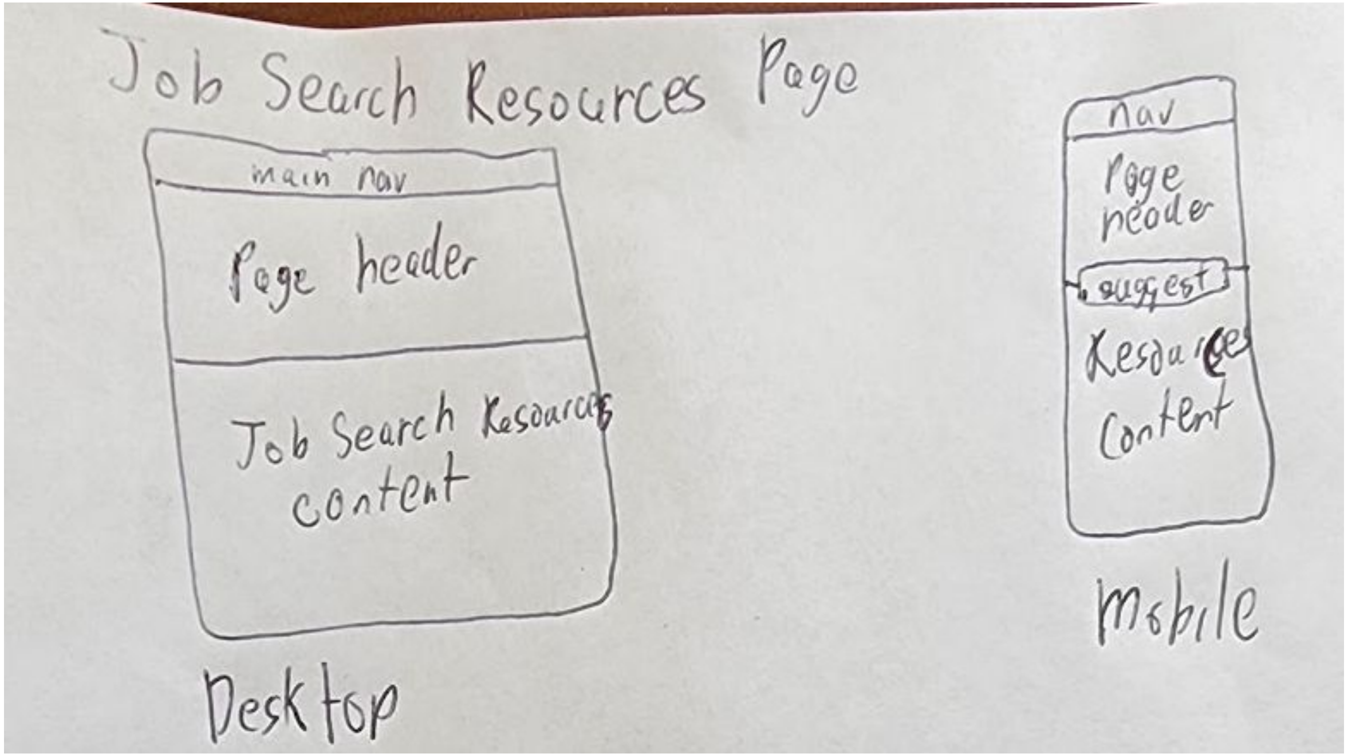
+ Submit a Resource

© Copyright 2024, All rights reserved.

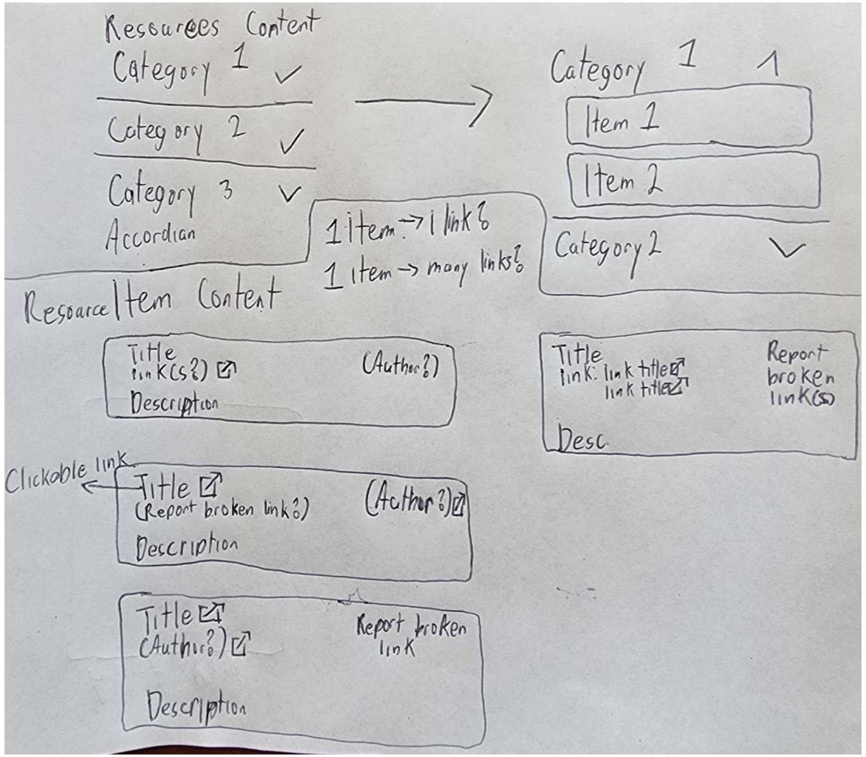
@Chad_R_Stewart Chad Stewart

Starting out, I built the Experience first

Building the Experience first



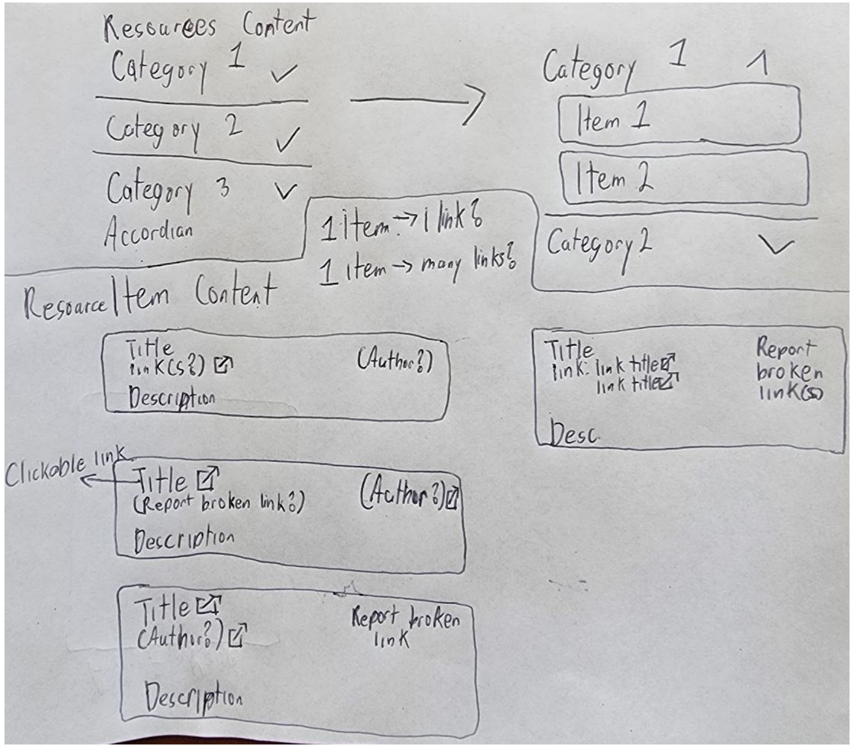
Building the Experience first



@Chad_R_Stewart Chad Stewart

Problem: How should I approach building the Experience?

Problem: Approach building the Experience



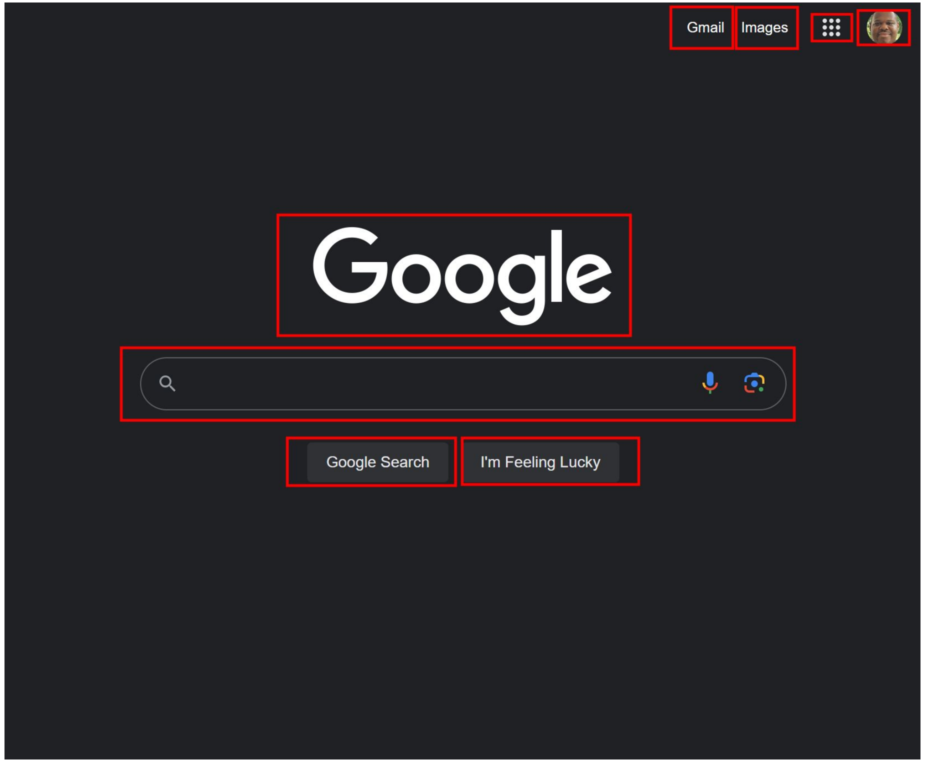
Problem: Approach to building the Experience

Concerns:

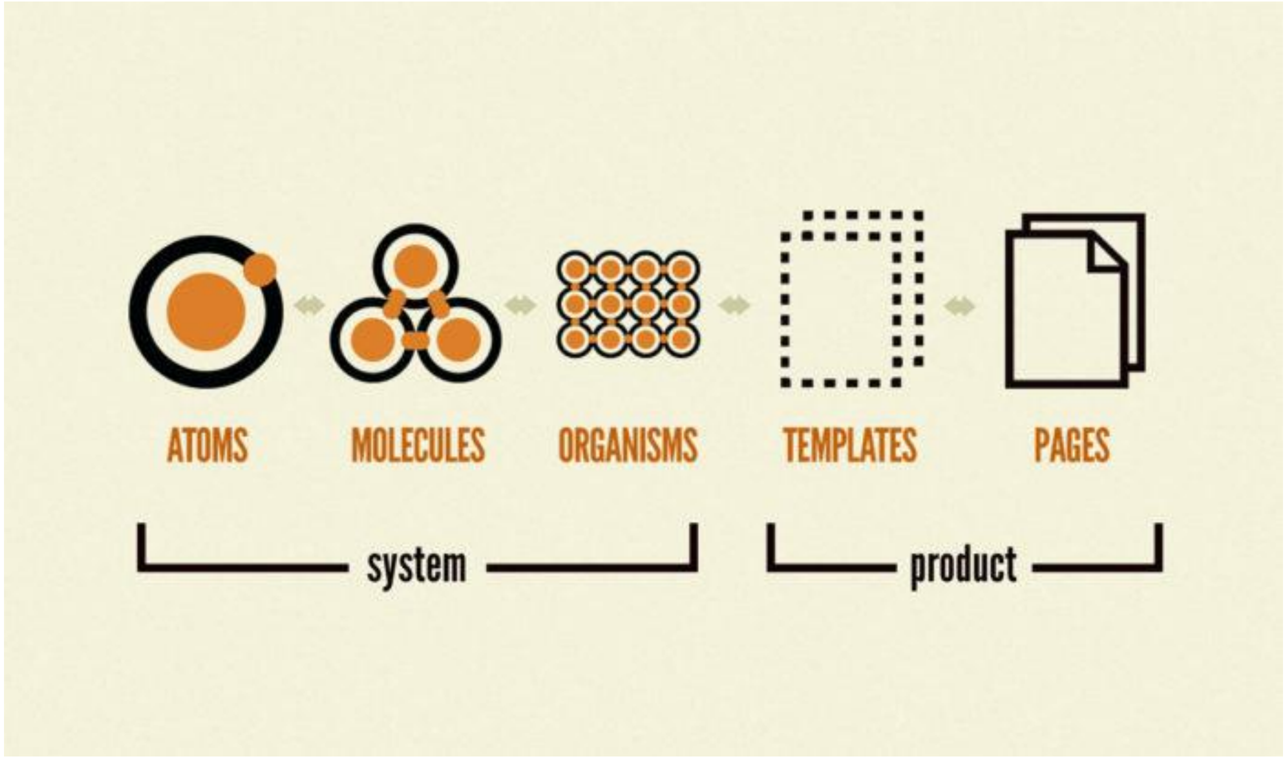
- Make it easy to experiment with parts of the Experience
- Make it easy to add to the Experience

Solution: Component-Driven Design & Atomic Design

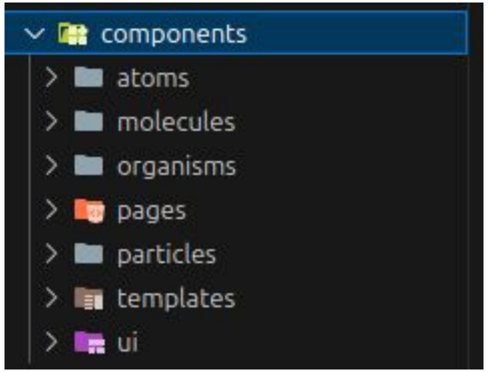
Solution: Component-Driven Design



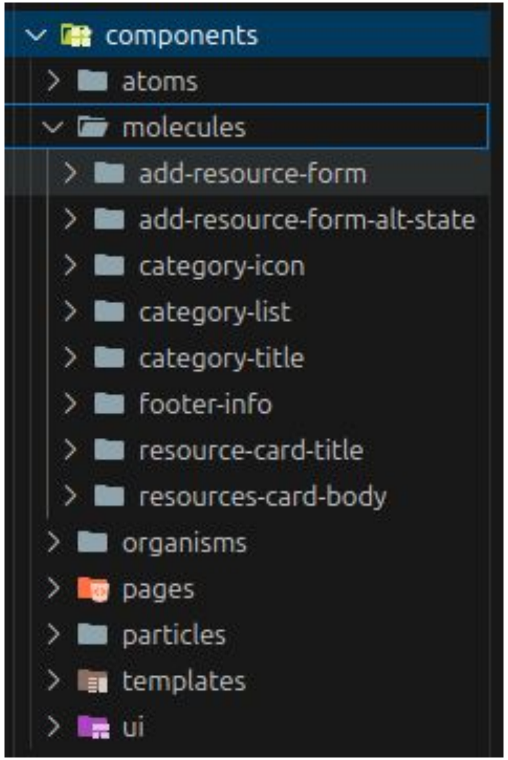
Solution: Atomic Design



Execution: Component-Driven Design & Atomic Design



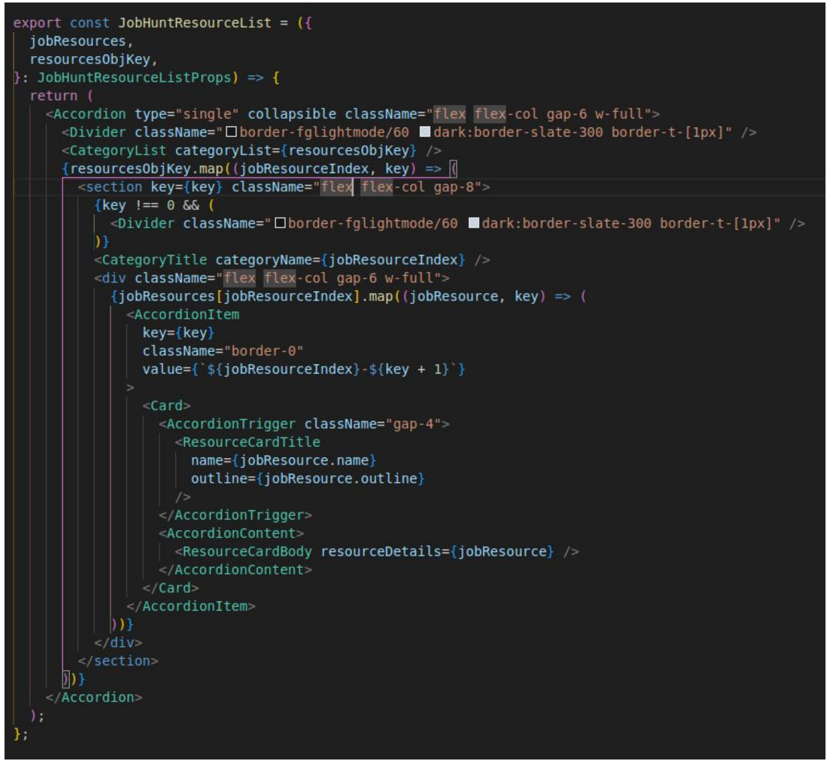
```
components
  atoms
  molecules
  organisms
  pages
  particles
  templates
  ui
```



```
components
  atoms
  molecules
    add-resource-form
    add-resource-form-alt-state
    category-icon
    category-list
    category-title
    footer-info
    resource-card-title
    resources-card-body
  organisms
  pages
  particles
  templates
  ui
```

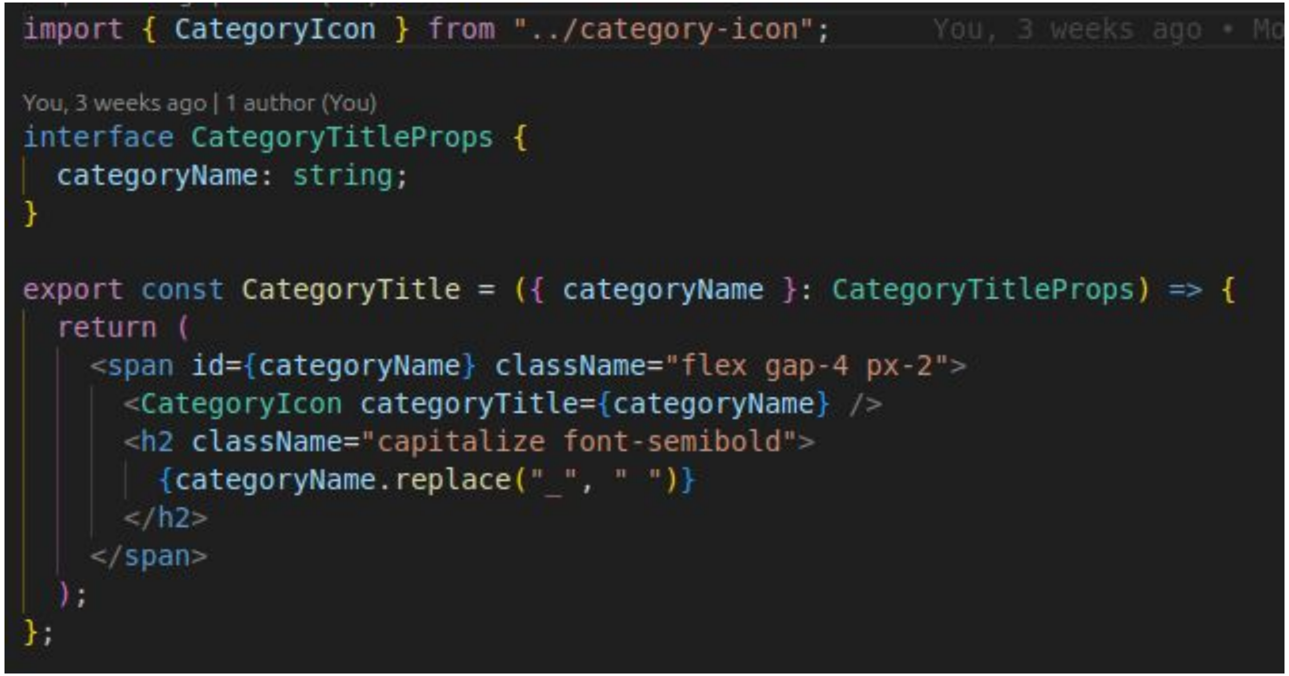
@Chad_R_Stewart Chad Stewart

Execution: Component-Driven Design & Atomic Design



```
export const JobHuntResourceList = ({
  jobResources,
  resourcesObjKey,
}: JobHuntResourceListProps) => {
  return (
    <Accordion type="single" collapsible className="flex flex-col gap-6 w-full">
      <Divider className="□border-fglightmode/60 ■dark:border-slate-300 border-t-[1px]" />
      <CategoryList categoryList={resourcesObjKey} />
      {resourcesObjKey.map((jobResourceIndex, key) => (
        <section key={key} className="flex flex-col gap-8">
          {key !== 0 && (
            <Divider className="□border-fglightmode/60 ■dark:border-slate-300 border-t-[1px]" />
          )}
          <CategoryTitle categoryName={jobResourceIndex} />
          <div className="flex flex-col gap-6 w-full">
            {jobResources[jobResourceIndex].map((jobResource, key) => (
              <AccordionItem
                key={key}
                className="border-0"
                value={`${jobResourceIndex}-${key + 1}`}
              >
                <Card>
                  <AccordionTrigger className="gap-4">
                    <ResourceCardTitle
                      name={jobResource.name}
                      outline={jobResource.outline}
                    />
                  </AccordionTrigger>
                  <AccordionContent>
                    <ResourceCardBody resourceDetails={jobResource} />
                  </AccordionContent>
                </Card>
              </AccordionItem>
            ))}
          </div>
        </section>
      ))}
    </Accordion>
  );
};
```

Execution: Component-Driven Design & Atomic Design



```
import { CategoryIcon } from "../category-icon";      You, 3 weeks ago • Mo

You, 3 weeks ago | 1 author (You)
interface CategoryTitleProps {
  categoryName: string;
}

export const CategoryTitle = ({ categoryName }: CategoryTitleProps) => {
  return (
    <span id={categoryName} className="flex gap-4 px-2">
      <CategoryIcon categoryTitle={categoryName} />
      <h2 className="capitalize font-semibold">
        {categoryName.replace("_", " ")}
      </h2>
    </span>
  );
};
```

@Chad_R_Stewart Chad Stewart

Summary: Approach to building the Experience

- Atomic Design gives you an easy-to-use framework to think about decomposing UI into loosely-coupled components
- Components tend to be more easy to reason about even when rendering multiple components
- There are a lot of resources that teaches Atomic Design when onboarding new team members

Problem: How do I style Components?

Problem: Styling Components

Concerns:

- Keeping CSS rules isolated to their components

Solution: Atomic / Functional CSS

Solution: Atomic / Functional CSS

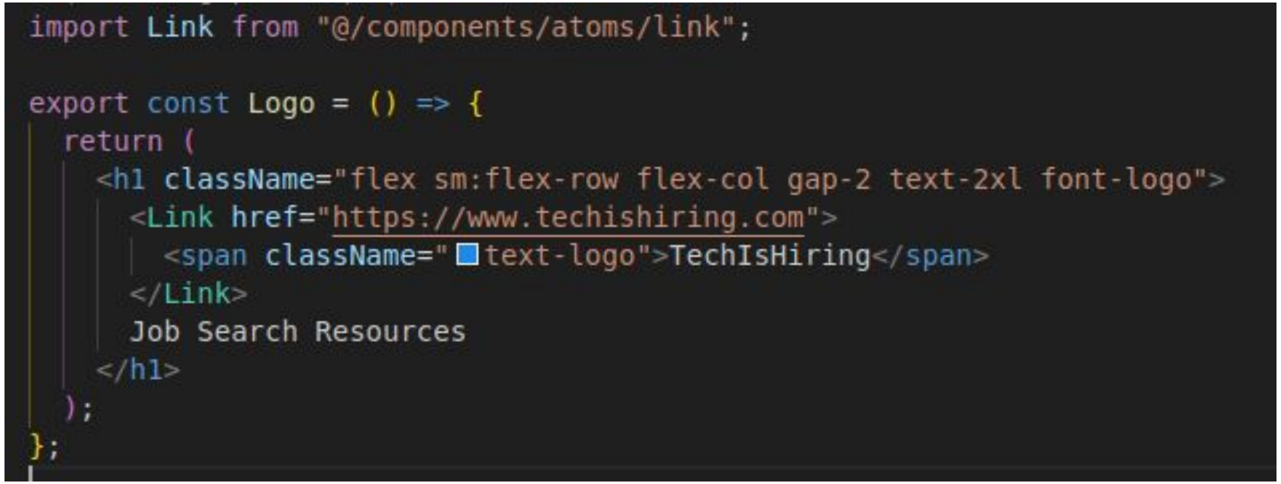
Various CSS Paradigms encouraging smaller predefined CSS classes, usually action-oriented, that are applied to markup to add styling

Solution: Atomic / Functional CSS

Advantages:

- Isolates CSS to specific components
- Easier to reason about CSS classes
- CSS errors tend to be isolated to the component with the erroneous rule
- Specific tooling or dependencies aren’t required

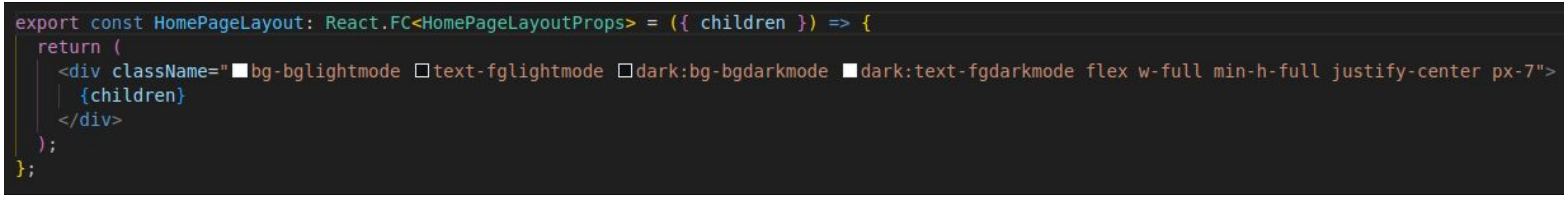
Execution: Atomic / Functional CSS - e.g. w/ Tailwind CSS



```
import Link from "@/components/atoms/link";

export const Logo = () => {
  return (
    <h1 className="flex sm:flex-row flex-col gap-2 text-2xl font-logo">
      <Link href="https://www.techishiring.com">
        <span className="text-logo">TechIsHiring</span>
      </Link>
      Job Search Resources
    </h1>
  );
};
```

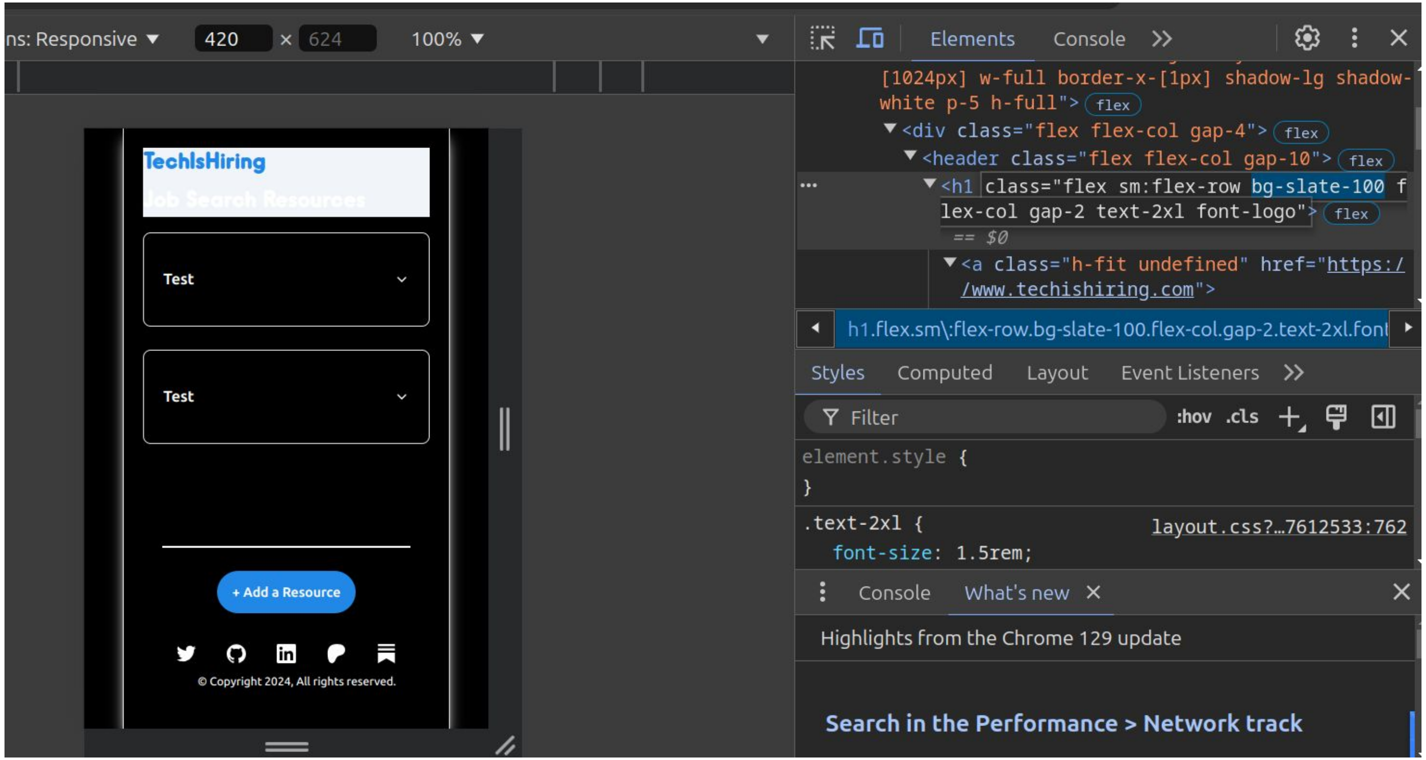
@Chad_R_Stewart Chad Stewart

Execution: Atomic / Functional CSS - e.g. w/ Tailwind CSS



```
export const HomePageLayout: React.FC<HomePageLayoutProps> = ({ children }) => {
  return (
    <div className="■bg-bglightmode □text-fglightmode □dark:bg-bgdarkmode ■dark:text-fgdarkmode flex w-full min-h-full justify-center px-7">
      {children}
    </div>
  );
};
```

Execution: Atomic / Functional CSS - e.g. w/ Tailwind CSS



```
[1024px] w-full border-x-[1px] shadow-lg shadow-
white p-5 h-full"> flex
  <div class="flex flex-col gap-4"> flex
    <header class="flex flex-col gap-10"> flex
      <h1 class="flex sm:flex-row bg-slate-100 f
      lex-col gap-2 text-2xl font-logo"> flex
       == $0
        <a class="h-fit undefined" href="https:/
        /www.techishiring.com">
```

```
h1.flex.sm\:flex-row.bg-slate-100.flex-col.gap-2.text-2xl.font
```

```
element.style {
}
.text-2xl {                                  layout.css?…7612533:762
  font-size: 1.5rem;
```

Highlights from the Chrome 129 update

Search in the Performance > Network track

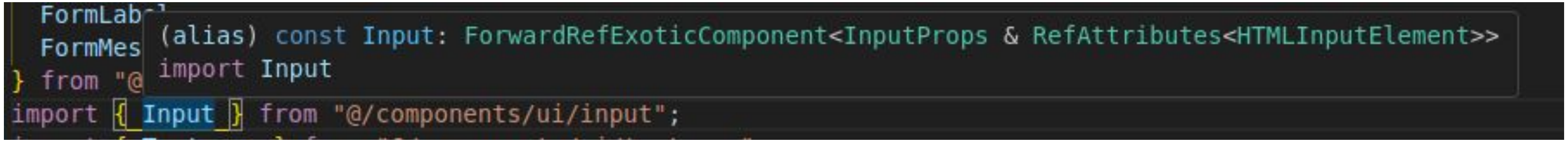
@Chad_R_Stewart Chad Stewart

Summary: Atomic / Functional CSS

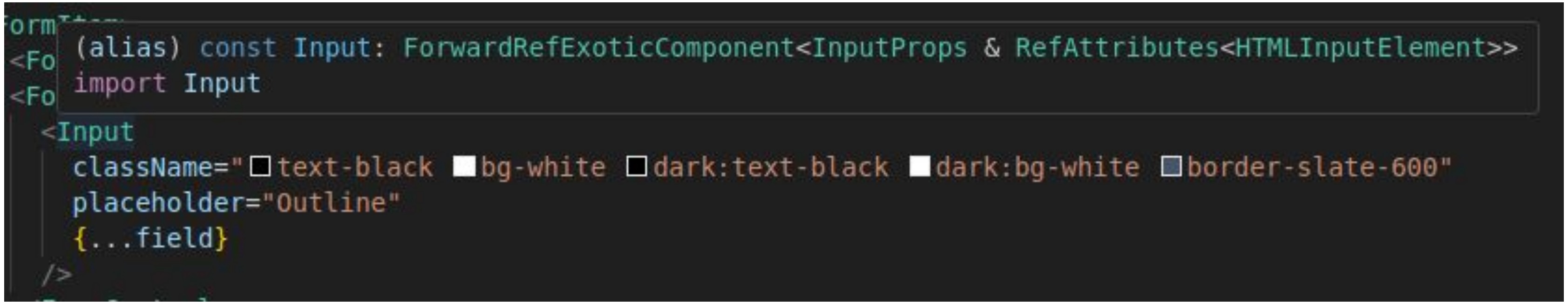
- Atomic CSS makes reasoning about styling easier because of how the classes are named, written and applied
- Atomic CSS is easier to debug

Supplemental: Component Libraries

Component Libraries - Write Interfaces to Libraries



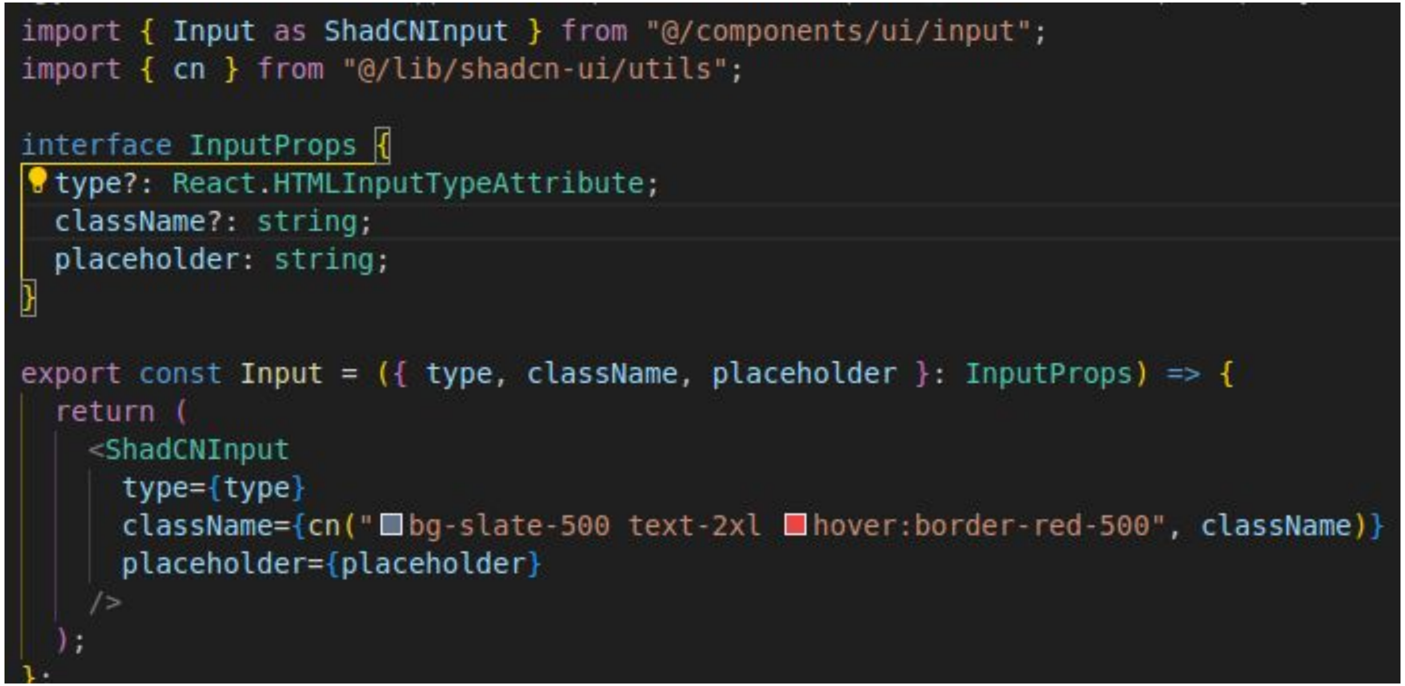
```
  FormLab
  FormMes  (alias) const Input: ForwardRefExoticComponent<InputProps & RefAttributes<HTMLInputElement>>
} from "@  import Input
import { Input } from "@/components/ui/input";
```



```
orm
<Fo  (alias) const Input: ForwardRefExoticComponent<InputProps & RefAttributes<HTMLInputElement>>
<Fo  import Input
  <Input
    className="■text-black □bg-white ■dark:text-black □dark:bg-white ▨border-slate-600"
    placeholder="Outline"
    {...field}
  />
```

@Chad_R_Stewart Chad Stewart

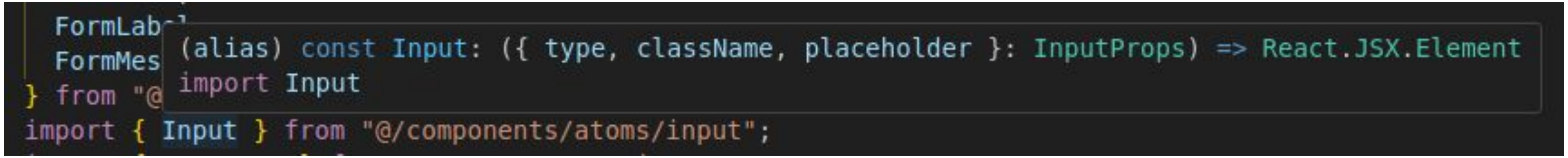
Component Libraries - Write Interfaces to Libraries



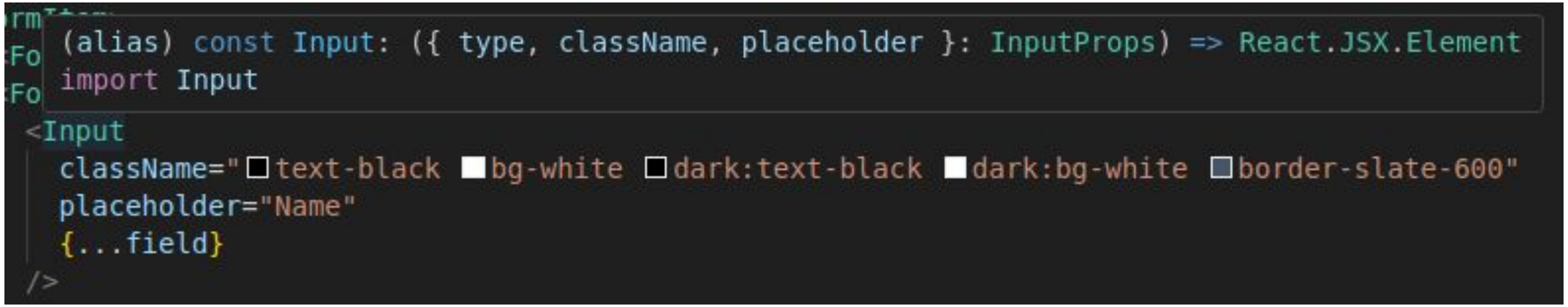
```
import { Input as ShadCNInput } from "@/components/ui/input";
import { cn } from "@/lib/shadcn-ui/utils";

interface InputProps {
  type?: React.HTMLInputTypeAttribute;
  className?: string;
  placeholder: string;
}

export const Input = ({ type, className, placeholder }: InputProps) => {
  return (
    <ShadCNInput
      type={type}
      className={cn("bg-slate-500 text-2xl hover:border-red-500", className)}
      placeholder={placeholder}
    />
  );
};
```

@Chad_R_Stewart Chad Stewart

Component Libraries - Write Interfaces to Libraries



```
  FormLab
  FormMes   (alias) const Input: ({ type, className, placeholder }: InputProps) => React.JSX.Element
} from "@   import Input
import { Input } from "@/components/atoms/input";
```



```
rm  (alias) const Input: ({ type, className, placeholder }: InputProps) => React.JSX.Element
Fo  import Input
Fo
 <Input
   className="■text-black □bg-white ■dark:text-black □dark:bg-white ■border-slate-600"
   placeholder="Name"
   {...field}
 />
```

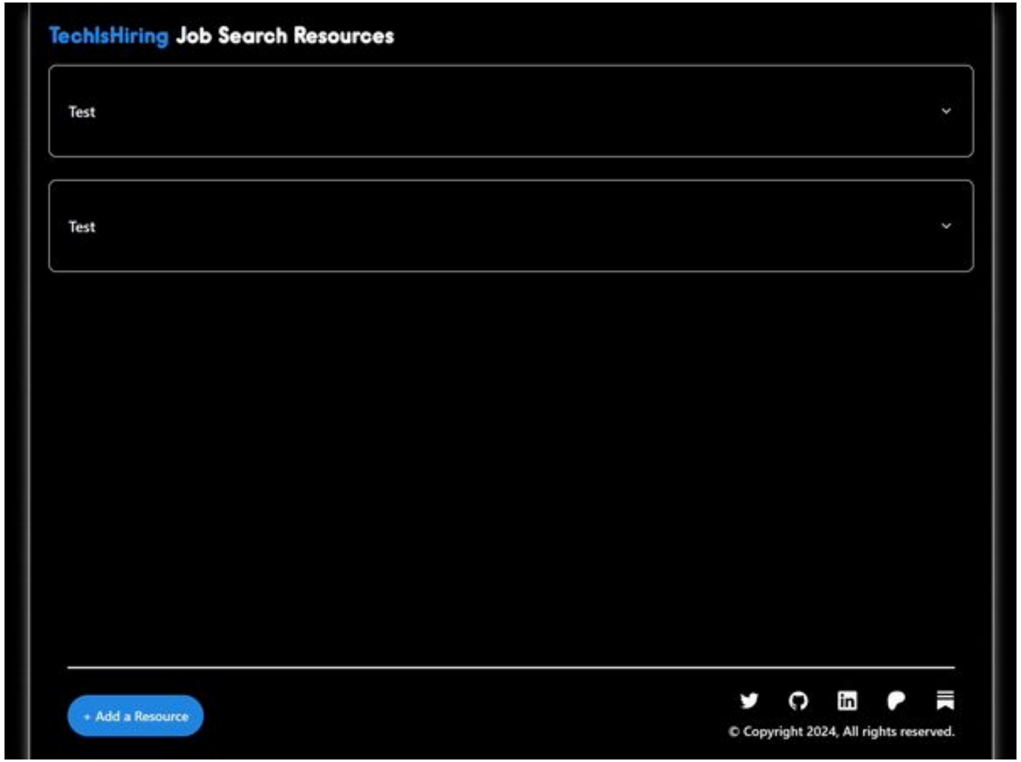
@Chad_R_Stewart Chad Stewart

Component Libraries - Write Interfaces to Libraries

Benefits

- Helps decouple Component Libraries from the rest of your UI
- Makes maintenance easier so if changes need to be made, your wrapper gets updated instead multiple instances of the component in the rest of your UI

The Experience (so far)


TechIsHiring Job Search Resources

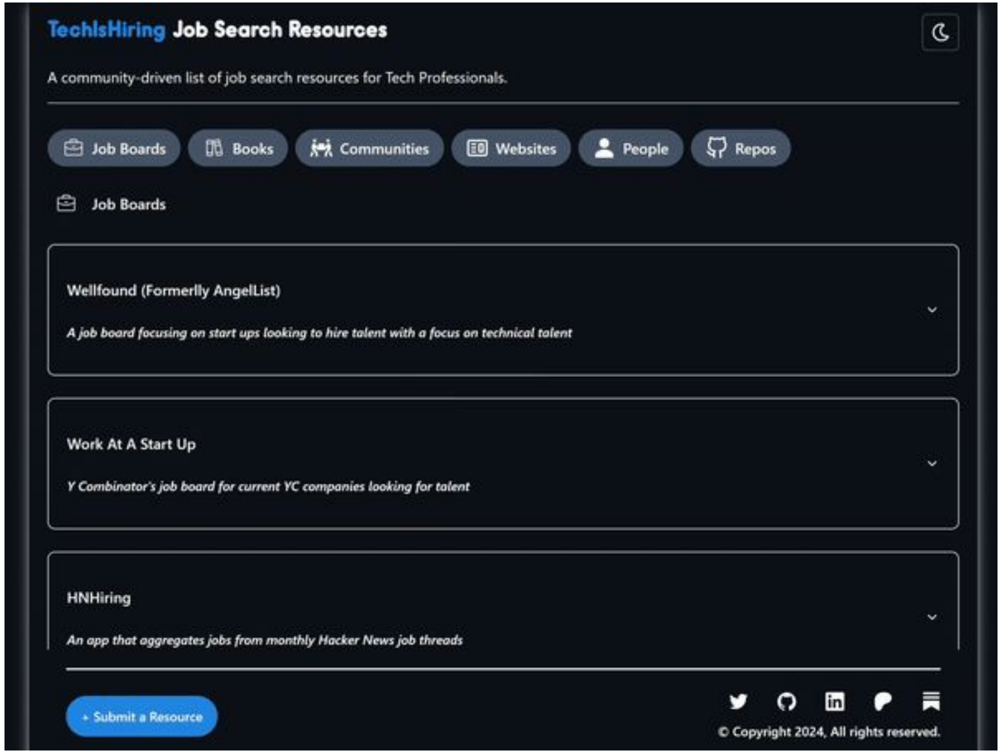
Test

Test

© Copyright 2024, All rights reserved.

Making the app functional

Making the app functional


TechIsHiring Job Search Resources

A community-driven list of job search resources for Tech Professionals.

Job Boards

Wellfound (Formerlly AngelList)

A job board focusing on start ups looking to hire talent with a focus on technical talent

Work At A Start Up

Y Combinator's job board for current YC companies looking for talent

HNHiring

An app that aggregates jobs from monthly Hacker News job threads

© Copyright 2024, All rights reserved.

@Chad_R_Stewart Chad Stewart

My approach to the app’s functionality

Making the app functional - Confs.Tech’s approach

Features:

- Pull data from a JSON file in a GitHub Repo
- Add data by making a Pull Request to the JSON data in the GitHub Repo

Problem: How do I approach adding Functionality?

Problem: Adding Functionality

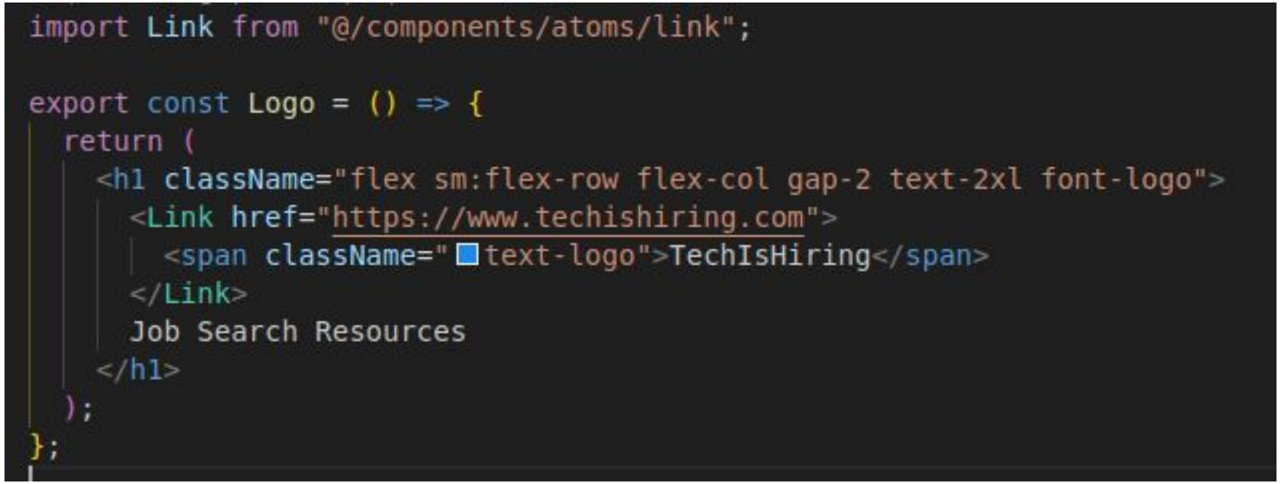
Concerns:

- Code should continue to be easy to reason about
- Changes to UI components should not overly disrupt functionality

Solution: Keep Presentation & Business Logic separate

Solution: Keep Presentation & Business Logic separate

Presentation Logic: Any code whose primary function is to display something on screen, i.e. HTML, CSS, Native Code, etc.

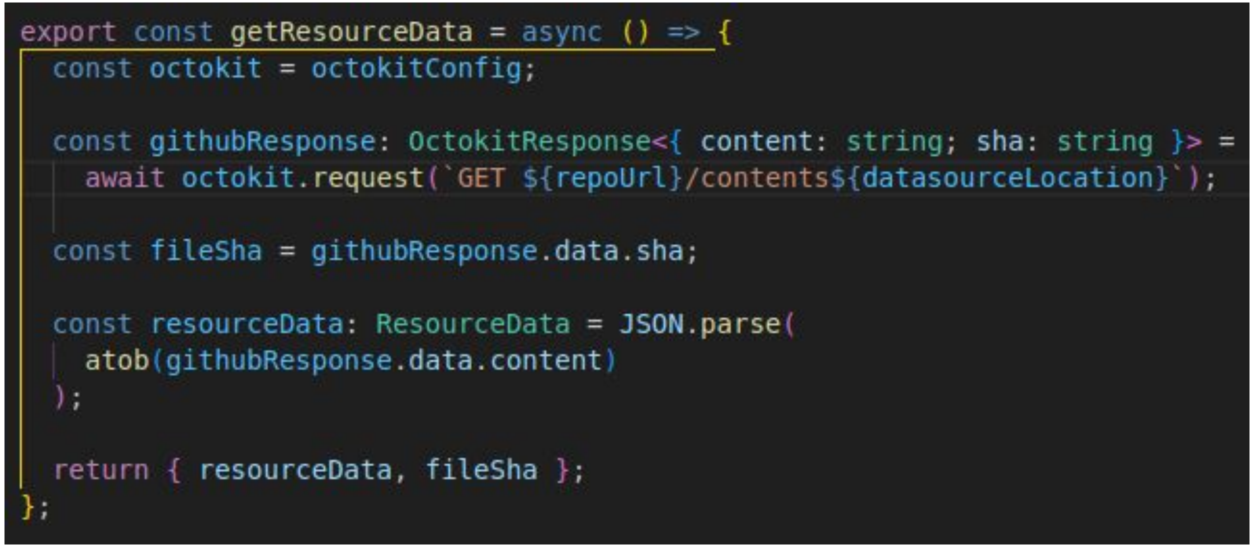


```
import Link from "@/components/atoms/link";

export const Logo = () => {
  return (
    <h1 className="flex sm:flex-row flex-col gap-2 text-2xl font-logo">
      <Link href="https://www.techishiring.com">
        <span className="text-logo">TechIsHiring</span>
      </Link>
      Job Search Resources
    </h1>
  );
};
```

@Chad_R_Stewart Chad Stewart

Solution: Keep Presentation & Business Logic separate

Business Logic: Any code that helps facilitate a Business function or rule, i.e. API Calls, Database Queries



```
export const getResourceData = async () => {
  const octokit = octokitConfig;

  const githubResponse: OctokitResponse<{ content: string; sha: string }> =
    await octokit.request(`GET ${repoUrl}/contents${datasourceLocation}`);

  const fileSha = githubResponse.data.sha;

  const resourceData: ResourceData = JSON.parse(
    atob(githubResponse.data.content)
  );

  return { resourceData, fileSha };
};
```

@Chad_R_Stewart Chad Stewart

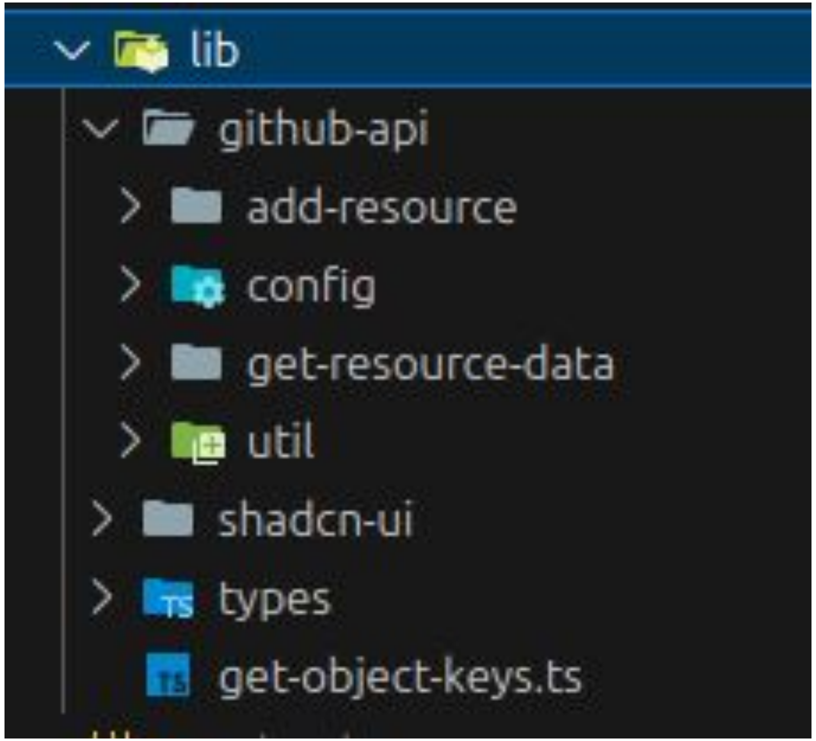
Solution: Keep Presentation & Business Logic separate

Presentation Logic: The codified version of the experience itself

Business Logic: The code powering the actions of interacting with the experience

… And both type of code's concerns are different

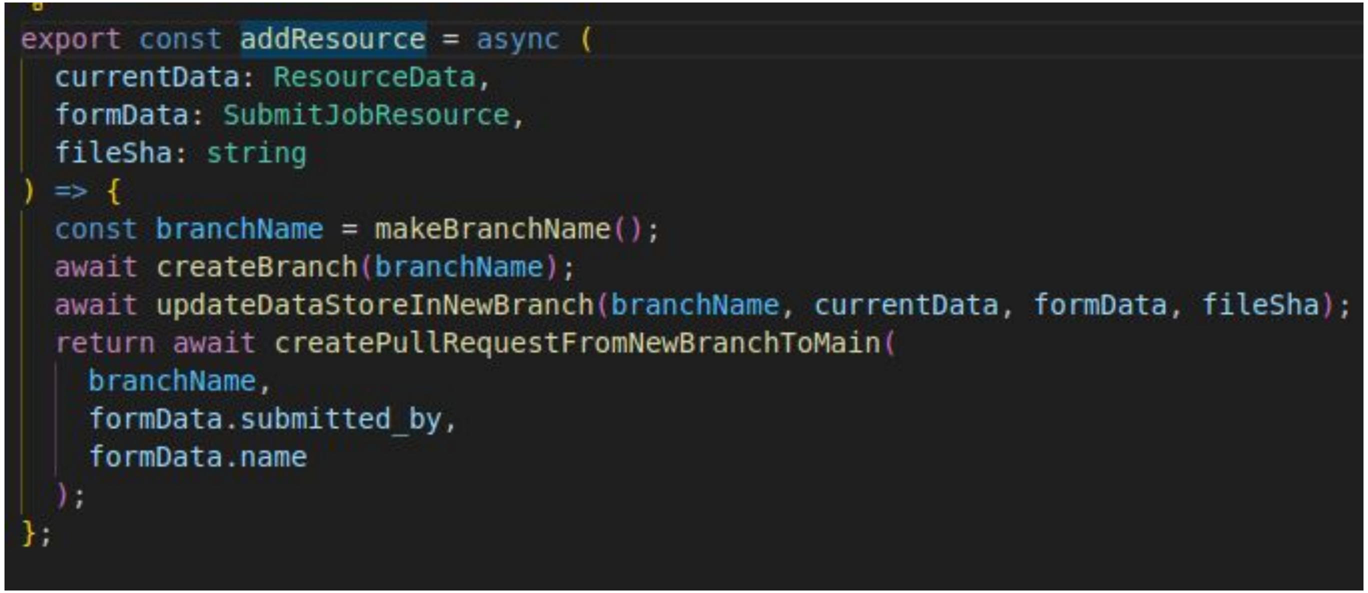
Execution: Keep Presentation & Business Logic separate



```
lib
  github-api
    add-resource
    config
    get-resource-data
    util
  shadcn-ui
  types
  get-object-keys.ts
```

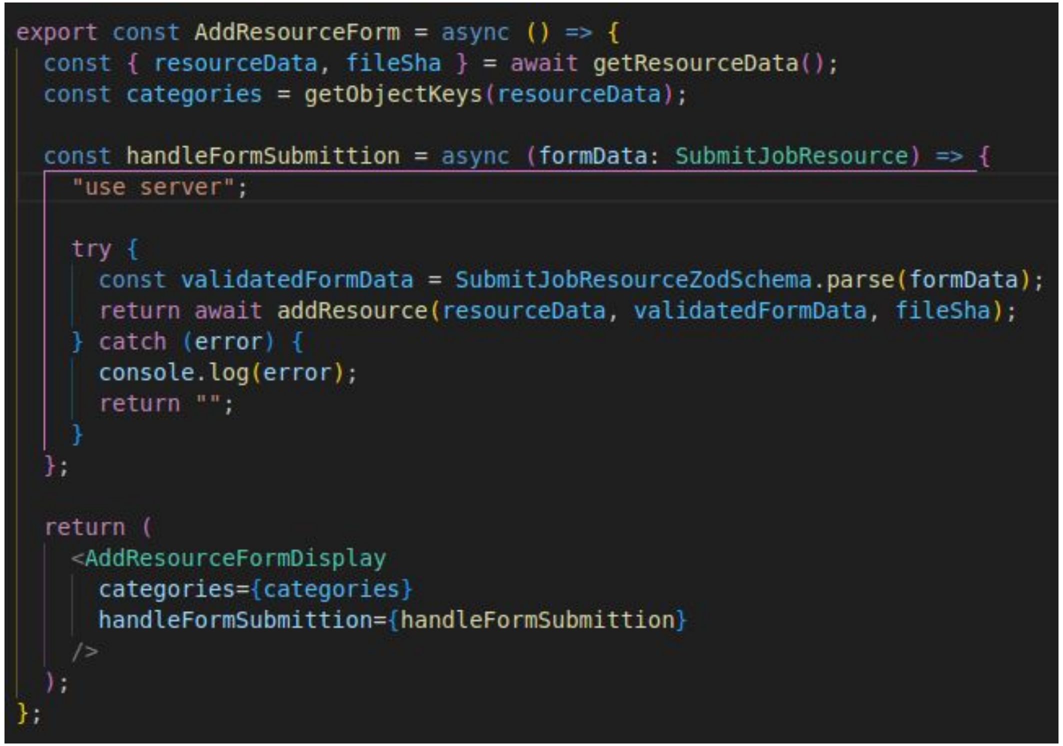
@Chad_R_Stewart Chad Stewart

Execution: Keep Presentation & Business Logic separate



```
export const addResource = async (
  currentData: ResourceData,
  formData: SubmitJobResource,
  fileSha: string
) => {
  const branchName = makeBranchName();
  await createBranch(branchName);
  await updateDataStoreInNewBranch(branchName, currentData, formData, fileSha);
  return await createPullRequestFromNewBranchToMain(
    branchName,
    formData.submitted_by,
    formData.name
  );
};
```

Execution: Keep Presentation & Business Logic separate



```
export const AddResourceForm = async () => {
  const { resourceData, fileSha } = await getResourceData();
  const categories = getObjectKeys(resourceData);

  const handleFormSubmittion = async (formData: SubmitJobResource) => {
    "use server";

    try {
      const validatedFormData = SubmitJobResourceZodSchema.parse(formData);
      return await addResource(resourceData, validatedFormData, fileSha);
    } catch (error) {
      console.log(error);
      return "";
    }
  };

  return (
    <AddResourceFormDisplay
      categories={categories}
      handleFormSubmittion={handleFormSubmittion}
    />
  );
};
```

@Chad_R_Stewart Chad Stewart

Summary: Keep Presentation & Business Logic separate

- Keeping Presentation & Business Logic separate allows them to focus on their specific concerns
- With Business Logic separate, you can leverage other patterns to specifically help apply your Business Logic
- Have Container Components handle running your Business Logic & pass whatever you need into your Presentational Components

Let’s add a feature

Let’s see the finished product!!

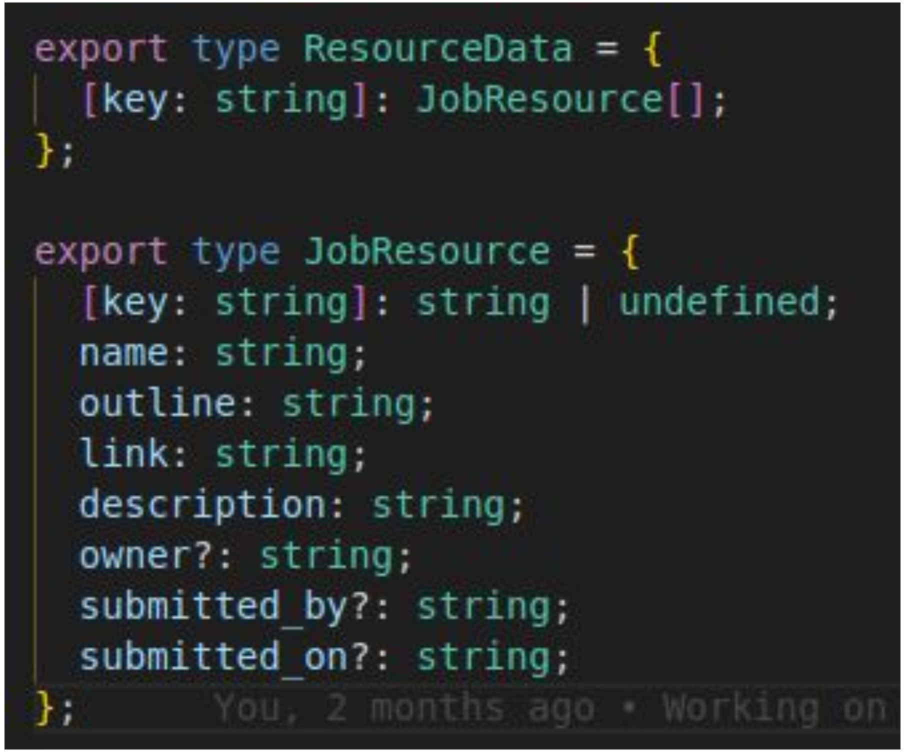
Is all of this really necessary?

Is all of this really necessary?

- Makes your application easier to manage & change
- Makes your application more accessible to other Software Engineers
- Allows other Software Engineers to learn the code base at their own pace & focus on their interests

Miscellaneous

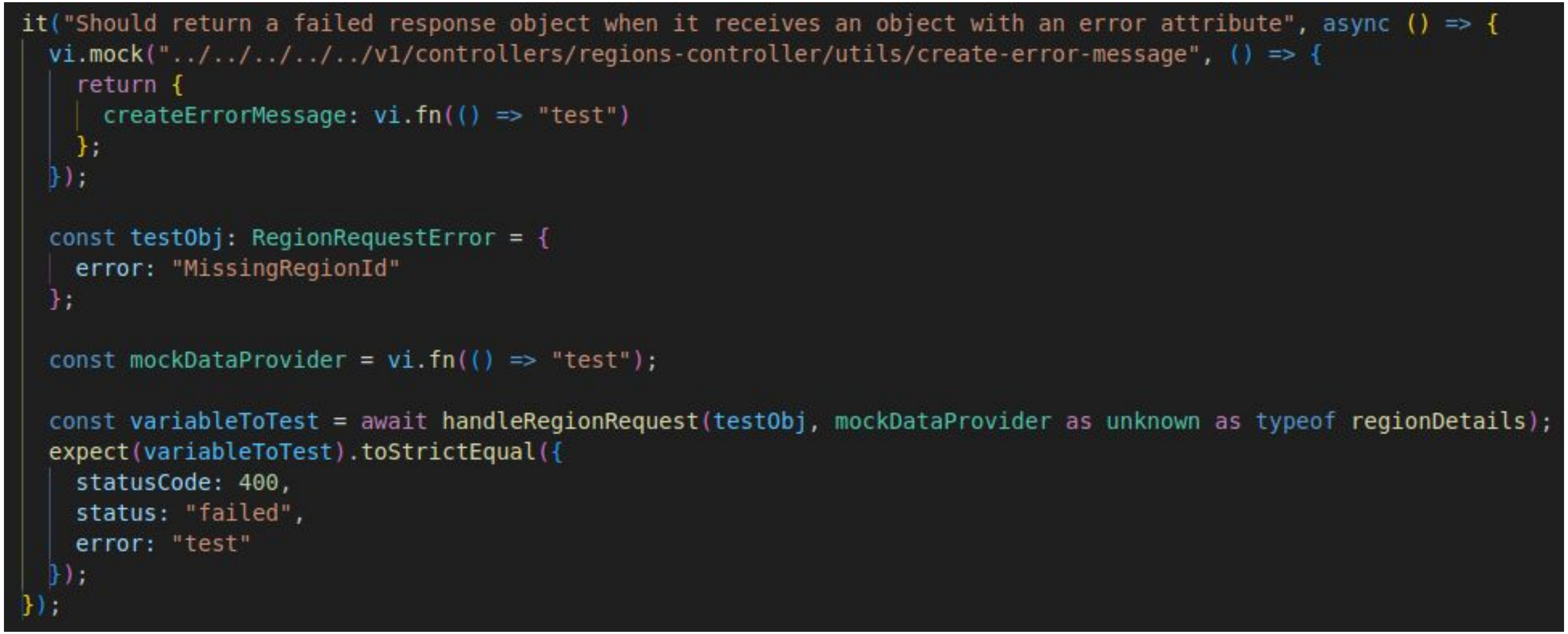
Miscellaneous: TypeScript



```
export type ResourceData = {
  [key: string]: JobResource[];
};

export type JobResource = {
  [key: string]: string | undefined;
  name: string;
  outline: string;
  link: string;
  description: string;
  owner?: string;
  submitted_by?: string;
  submitted_on?: string;
};       You, 2 months ago • Working on
```

@Chad_R_Stewart Chad Stewart

Miscellaneous: Unit Testing



```
it("Should return a failed response object when it receives an object with an error attribute", async () => {
  vi.mock("../../../../../v1/controllers/regions-controller/utils/create-error-message", () => {
    return {
      createErrorMessage: vi.fn(() => "test")
    };
  });

  const testObj: RegionRequestError = {
    error: "MissingRegionId"
  };

  const mockDataProvider = vi.fn(() => "test");

  const variableToTest = await handleRegionRequest(testObj, mockDataProvider as unknown as typeof regionDetails);
  expect(variableToTest).toStrictEqual({
    statusCode: 400,
    status: "failed",
    error: "test"
  });
});
```

@Chad_R_Stewart Chad Stewart

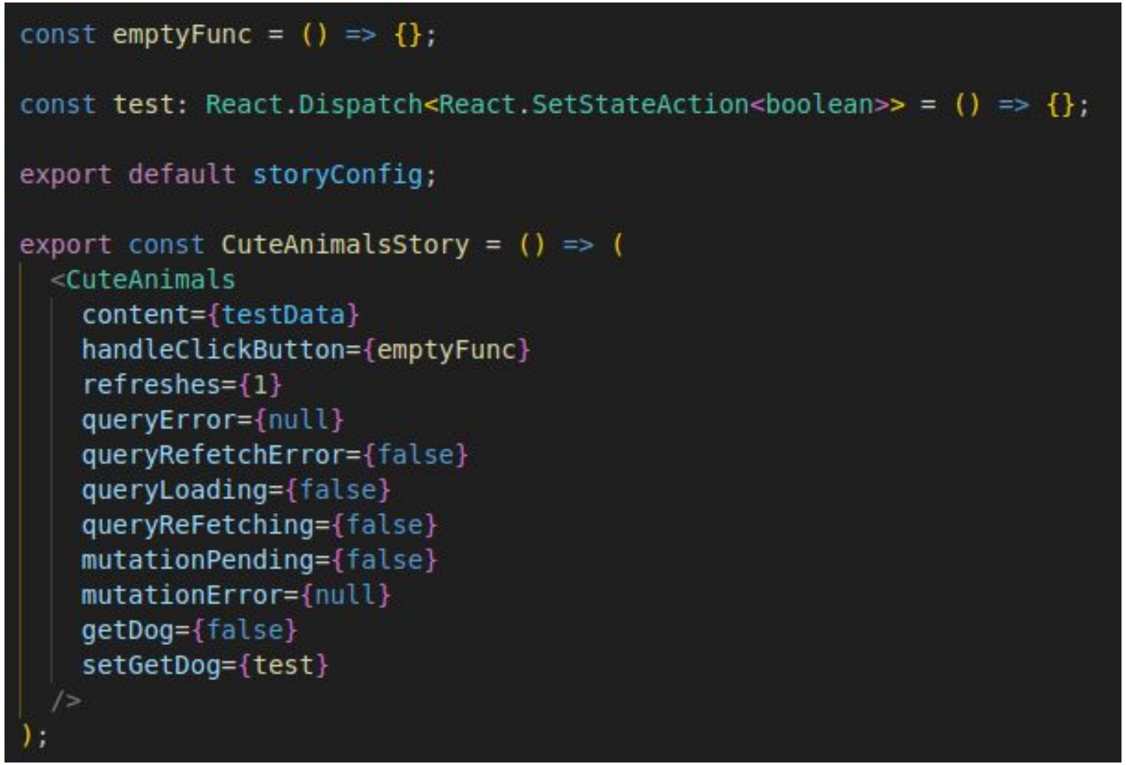
Miscellaneous: Unit Testing

"...those tests [unit / isolated tests] put tremendous pressure on our designs."
- J. B. Rainsberger, JBrains

Miscellaneous: Unit Testing

Unit / Isolated tests put pressure on our software designs through the feedback that they give back while writing tests

Miscellaneous: Storybook

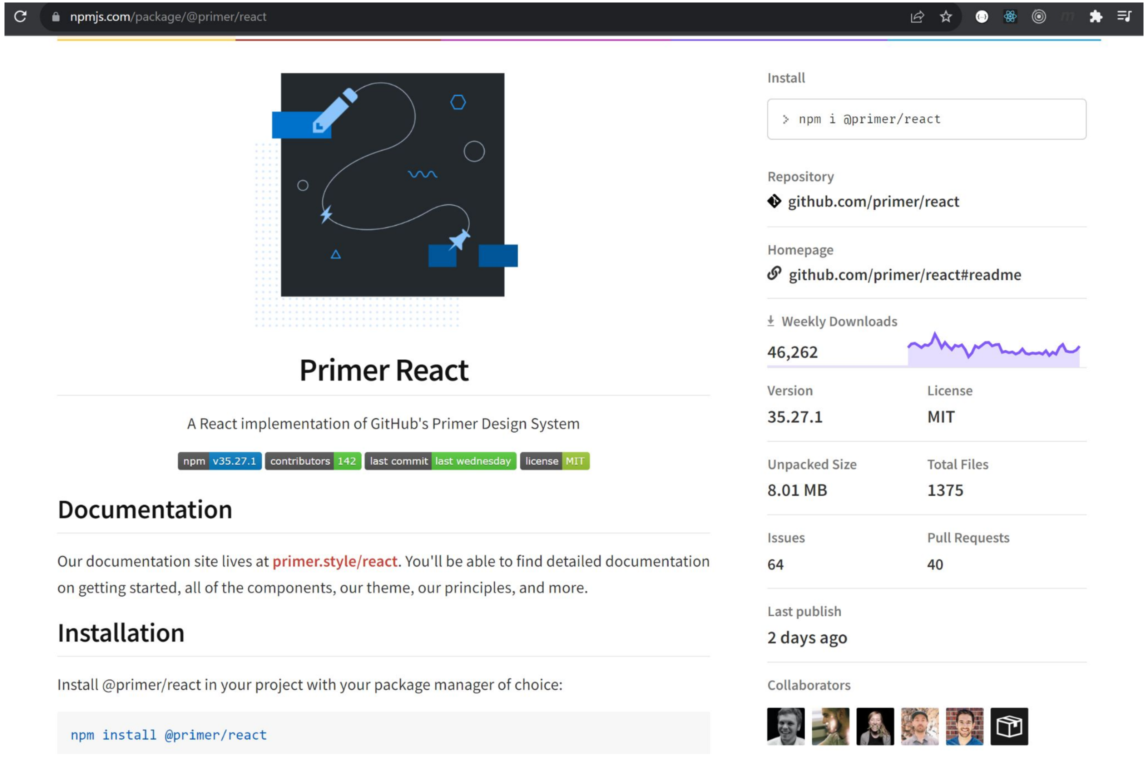


```
const emptyFunc = () => {};

const test: React.Dispatch<React.SetStateAction<boolean>> = () => {};

export default storyConfig;

export const CuteAnimalsStory = () => (
  <CuteAnimals
    content={testData}
    handleClickButton={emptyFunc}
    refreshes={1}
    queryError={null}
    queryRefetchError={false}
    queryLoading={false}
    queryReFetching={false}
    mutationPending={false}
    mutationError={null}
    getDog={false}
    setGetDog={test}
  />
);
```

@Chad_R_Stewart Chad Stewart

Beyond this talk

Beyond this talk - Org-specific Component Library



Primer React

A React implementation of GitHub's Primer Design System

npm v35.27.1 contributors 142 last commit last wednesday license MIT

Documentation

Our documentation site lives at primer.style/react. You'll be able to find detailed documentation on getting started, all of the components, our theme, our principles, and more.

Installation

Install @primer/react in your project with your package manager of choice:

```
npm install @primer/react
```

Install

```
> npm i @primer/react
```

Repository
github.com/primer/react

Homepage
github.com/primer/react#readme

Weekly Downloads
46,262

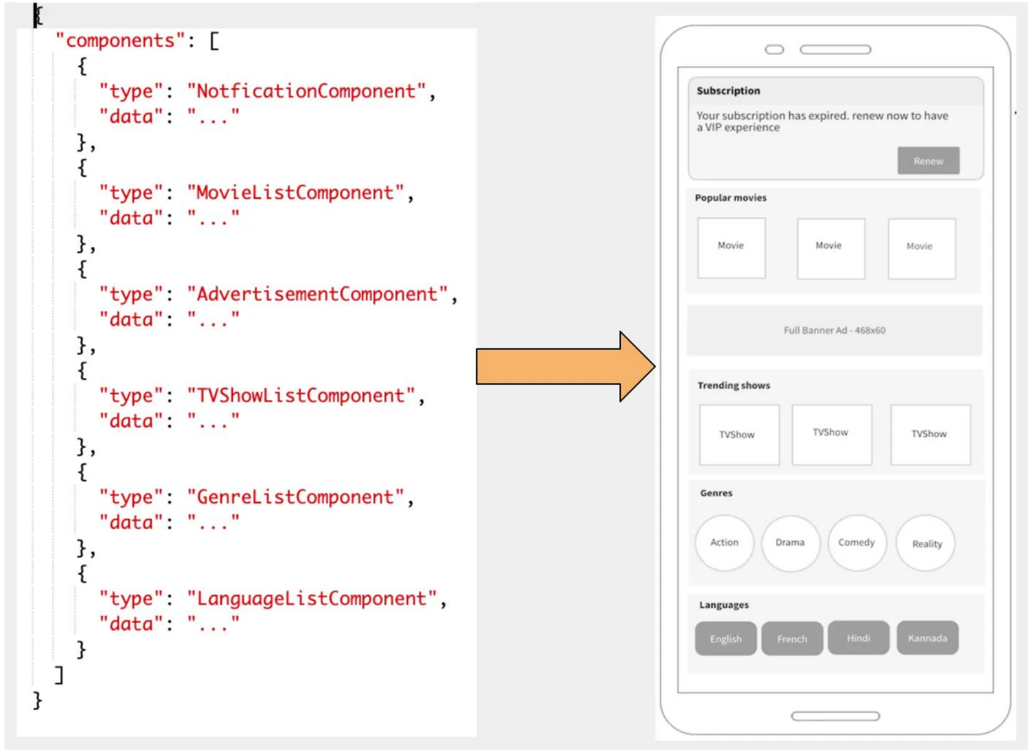
Version	License
35.27.1	MIT
Unpacked Size	Total Files
8.01 MB	1375
Issues	Pull Requests
64	40

Last publish
2 days ago

Collaborators

@Chad_R_Stewart Chad Stewart

Beyond this talk - Server-Driven UI



```
{
  "components": [
    {
      "type": "NotficationComponent",
      "data": "..."
    },
    {
      "type": "MovieListComponent",
      "data": "..."
    },
    {
      "type": "AdvertisementComponent",
      "data": "..."
    },
    {
      "type": "TVShowListComponent",
      "data": "..."
    },
    {
      "type": "GenreListComponent",
      "data": "..."
    },
    {
      "type": "LanguageListComponent",
      "data": "..."
    }
  ]
}
```

Subscription

Your subscription has expired. renew now to have a VIP experience

@Chad_R_Stewart Chad Stewart

Thanks for attending my talk!

@Chad_R_Stewart Chad Stewart

Thanks for attending my talk! - Special Thanks

Rizèl Scarlett @blackgirlbytes

Carmen Huidobro @hola_soy_milk

Jeff Boek @itsboek

Robbie Holmes @RobbieTheGeek

Thanks for attending my talk! - Where to find me

@Chad_R_Stewart Chad Stewart