React 디자인패턴

Status	Not started
Assign	React.js

1. Presentational and Container Component Pattern

➡️ 데이터 처리와 데이터 출력을 분리하는 패턴입니다.

✏️ Container Components : how things work

- 컨테이너 컴포넌트에서는 주로 데이터 fetch가 이루어 지게 됩니다.
  - Redux를 이용해 상태 관리를 하게 된다면 `dispatch` 를 예로 들 수 있습니다.
- 연관이 있는 서브 컴포넌트를 렌더링합니다.
- DOM Markup 이나 스타일(css)가 없습니다.
- Presentational 또는 Container Components 에 callback 함수나 데이터를 전달해 줄 수 있습니다.
- stateful한 경향을 가지고 있는 컴포넌트입니다.

✏️ Presentational Components (FC) : how things look

➡️ 화면에 보여지는 것만을 담당하는 Components 입니다.

- DOM markup 과 style(css)를 가지고 있습니다.
- props를 통해서 데이터나 callbacks 받아 올 수 있습니다.
- 뷰에 필요한 state를 가지고 있을 수 있습니다.
- state, lifecycle, Performance optimization이 필요한 경우가 아니면 Functional component로 작성합니다.
- stateless한 경향을 가지고 있는 컴포넌트입니다.

✏️ 예시

- **Container Component**➡️ 데이터를 Presenter Component에 전달해줍니다.

```
// CommentListContainer.js
import React from "react";
import CommentList from "./CommentList";

class CommentListContainer extends React.Component {
  constructor() {
    super();
    this.state = { comments: [] };
  }

  componentDidMount() {
    fetch("/my-comments.json")
      .then(res => res.json())
      .then(comments => this.setState({ comments }));
  }

  render() {
    return <CommentList comments={this.state.comments} />;
  }
}
```

- **Presenter Component**➡️ Container Component에서 받은

```
// CommentListPresenter.js
import React from "react";

const Commentlist = comments => (
  <ul>
    {comments.map(({ body, author }) => (
      <li>
        {body}-{author}
```

```
      </li>
    ))}
  </ul>
```

✏️ 장점

1. **관심사의 분리를 더 잘할 수 있다.**

이 패턴 방식으로 Component 를 작성하면 작업을 하는 앱 과 UI 를 보다 이해하기 쉽게 만들 수 있습니다.

1. **재 사용성을 높일 수 있다.**

완전히 다른 곳으로부터 온 여러 상태(state) 라 할지라도, 이를 표현하기 위해 같은 Presentational Component 사용할 수 있는데 이때, 각 상태를 나타내는 Container Component 를 만들어 이를 또 재사용 할 수 있습니다.

1. **Markup 작업이 편하다.**

이 패턴을 제대로 적용하려면 Layout Component 를 별도로 추출하여 관리해야 하는데, 이렇게 Layout Component 를 추출하게 되면 똑같은 Layout 마크업을 여러 Container Component 에 작성하는 작업을 피할 수 있습니다.

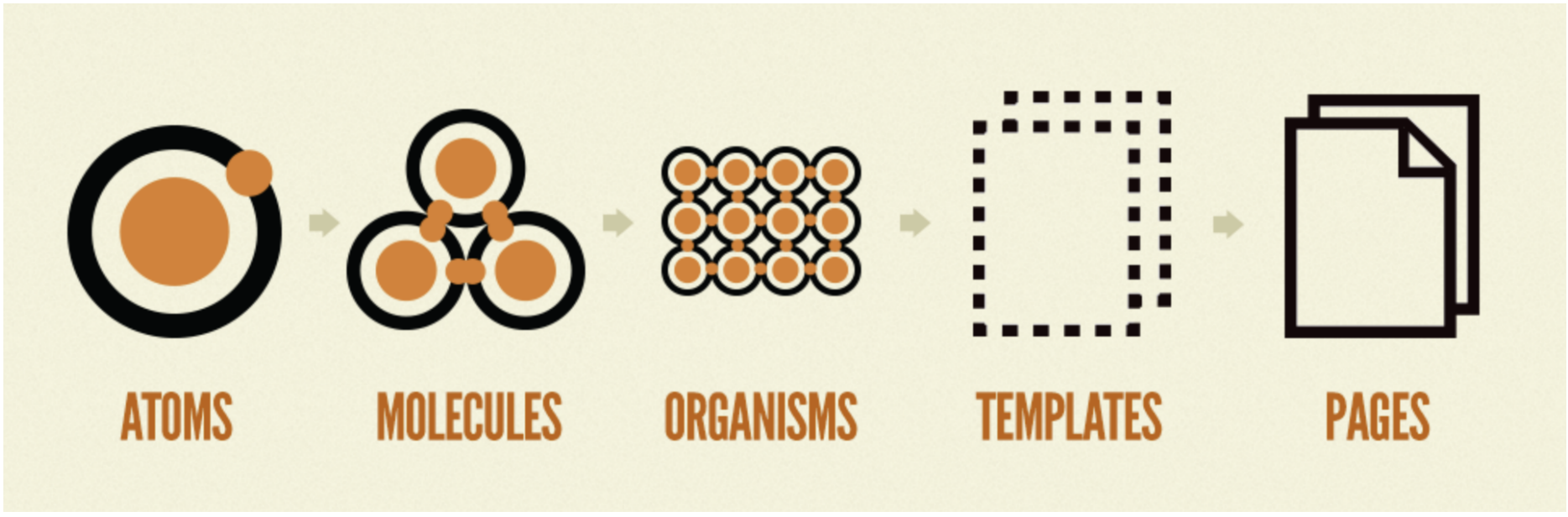
---

2. Atomic Design Pattern

➡️ 디자인 요소들을 나누어 파악하고 이 요소들이 조합되는 과정을 통해서 디자인을 구성하는 방식을 의미합니다.

Atomic Design 에서는 5개의 구분된 단계가 존재합니다.

- **Atoms**
- **Molecules**
- **Organisms**
- **Templates**
- **Pages**



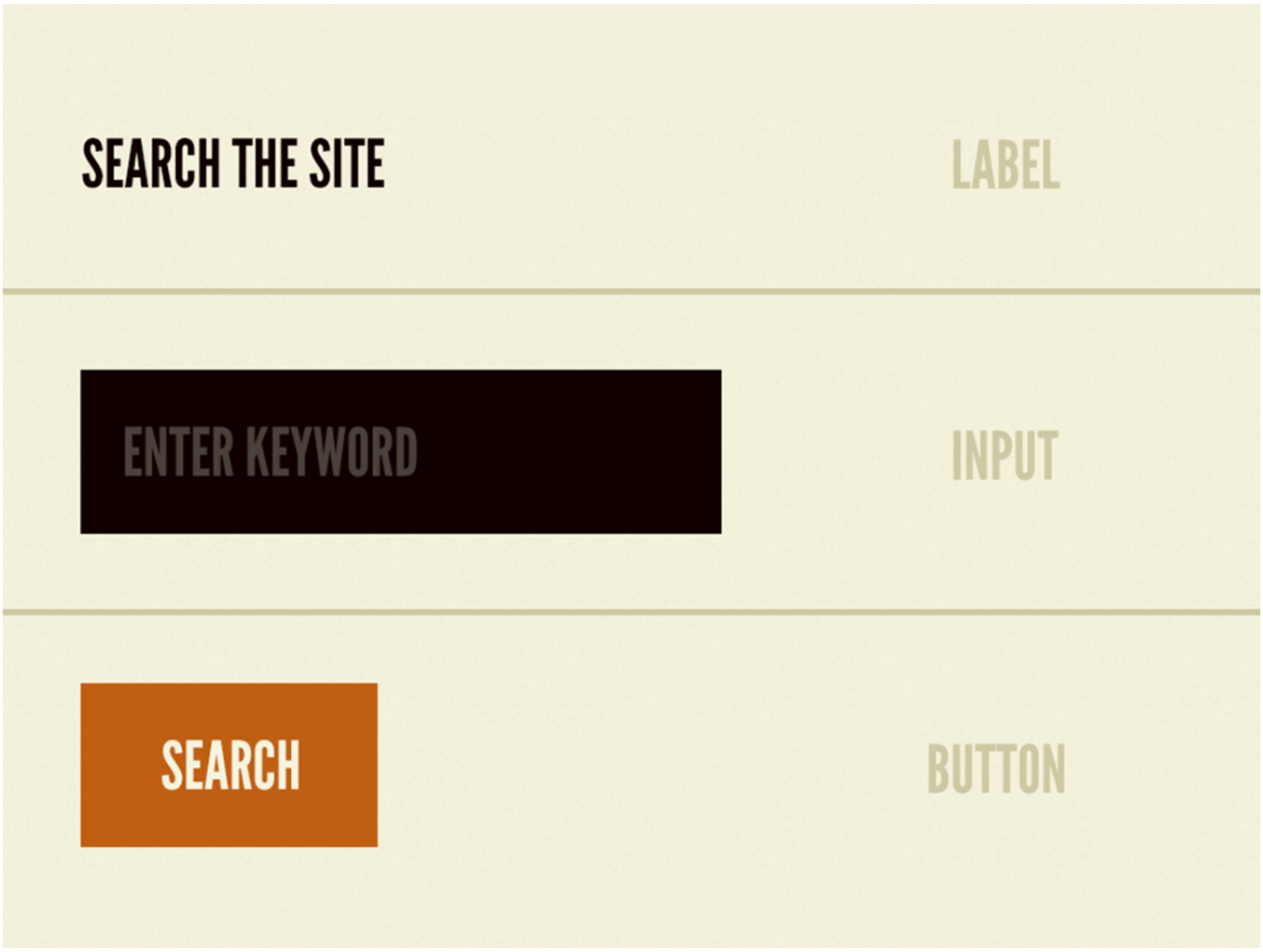
각각 개별적으로 살펴보겠습니다. 🧐

⚛️ Atoms

➡️ 하나의 구성 요소. 본인 자체의 스타일만 가지고 있으며 다른 곳에 영향을 미치는 스타일은 적용되지 않아야 합니다. 원자는 form labels, inputs, buttons와 같은 basic hmtl elements를 포함합니다.

예시)

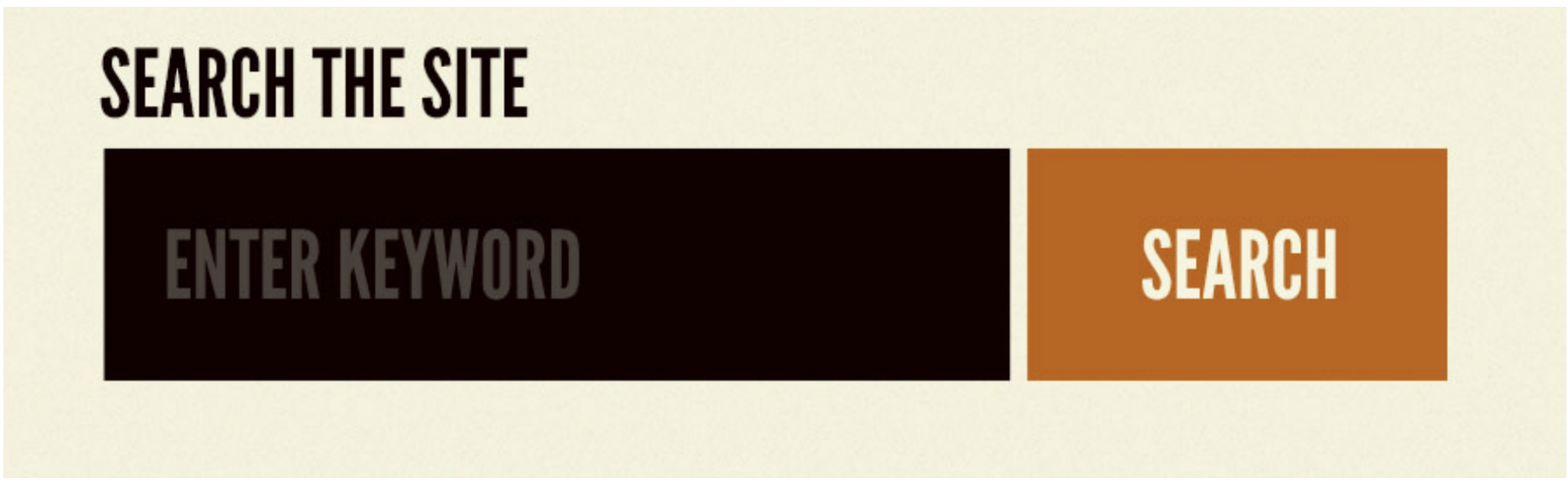
```
// src/components/atoms/button/index.js

const Button = ({ type = 'button', children = '' }) => (
  <button type={type}>{children}</button>   // 가장 기본이 되는 html 태그입니다.
)
```



⚛️ Molecules

➡️ Atoms가 모여서 만들어지는 하나의 구성 요소

Atom 단위인 input label, input, buttoms를 합쳐 새로운 의미있는 단위를 만들 수 있습니다. 실제로 무언가 동작을 할 수 있게 됩니다.



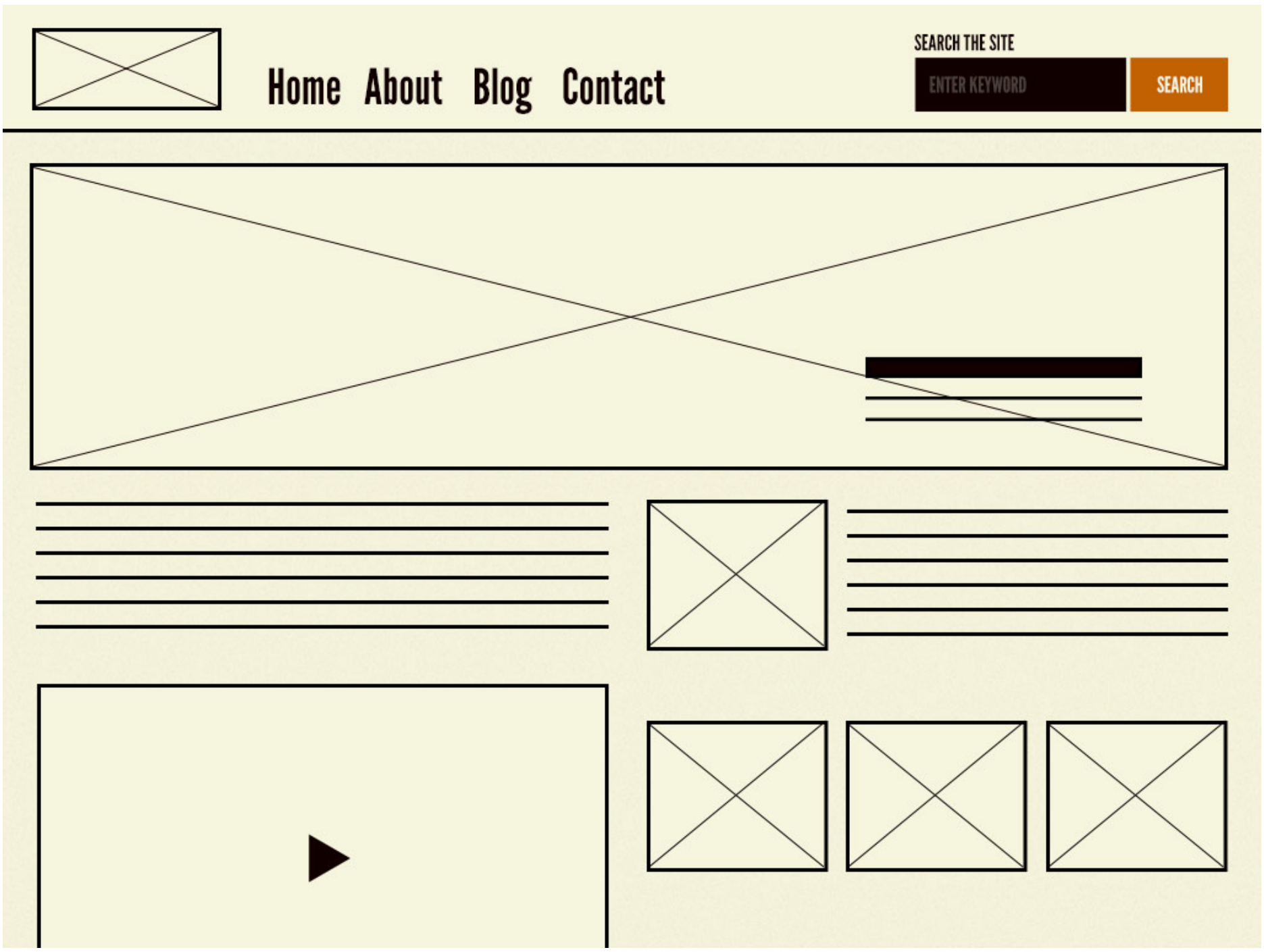
⚛️ Organisms: 분자들의 모음

➡️ **Organisms(유기체)**는 분자들의 모음으로 비교적 복잡한 구조를 가지게 됩니다. **Organisms(유기체)**는 서로 동일하거나 다른 **molecules(분자)**로 구성될 수 있습니다.

유기체는 로고, 메인 내비게이션, 검색, 소셜 미디어 채널리스트와 같은 다양한 컴포넌트(molecules)로 구성될 수 있습니다.

⚛️ Templates: 유기체들을 모아 템플릿으로 생성, 스타일링에 집중한 단위

➡️ Templates의 중요한 특성은 페이지의 최종 내용보다는 페이지의 기본 내용 구조에 초점을 맞춘다는 것입니다.



⚛️ Pages: 실제 페이지를 구성

➡️ 페이지는 실제 대표적인 콘텐츠가 배치된 UI의 모습을 보여주는 템플릿의 특정 인스턴스입니다.

Pages 단위에서 어플리케이션 상태 관리(리덕스, 모벡스 등등)가 이루어져야 합니다.

하지만 분자, 유기체, 템플릿 단위에서 컴포넌틑를 동작시키기 위한 상태를 관리하는건 괜찮습니다. (input과 onChange를 useState로 관리하는등의 상태관리)

3. 디자인 패턴 적용해보기 ⌨️

🎨 Presentational and Container Component Pattern

➡️ Class 101 사이트의 Class 상세 페이지에 대한 구조입니다. => 참고 링크

페이지 내에서 후기, 클래스 소개, 컬리큘럼, 키트 소개, 크리에이터, 커뮤니티, 환불 정책, 추천 항목 별로 컴포넌트를 나눴습니다.

각 컴포넌트 별로 디렉토리를 생성하고 각 디렉토리 내부에는 로직을 담당하는 *Container*와 화면에 데이터를 보여주는 *Presenter*, 그리고 스타일 파일로 나눠주었습니다.

프로젝트에서 전체 스타일링을 담당하는 것은 `globalStyle` 디렉토리에서 따로 관리할 수 있도록 했습니다.

상태관리를 담당하는 store 디렉토리를 생성하고 그 안에 `reducers`, `sagas` 디렉토리를 생성하여 구조를 잡았습니다.

> Presentational and Container Component Pattern을 기반으로 프로젝트를 설계함으로써 화면에 보여지는 것과 비즈니스 로직을 구분하여 작업을 수행할 수 있다는 점이 큰 장점이라고 느꼈습니다.

```
.
└── src
    ├── @types  // 타입을 관리하는 곳
    │   ├── dataTypes.ts
    │   └── handlerTypes.ts
    ├── App.tsx
    ├── Modal
    │ ├── ModalContainer.tsx
    │ ├── ModalPresenter.tsx
    │ └── Modal.style.ts
    │
    ├── components
    │   ├── ClassInformation
    │   │   ├── ClassInformation.style.ts
    │   │   ├── ClassInformationContainer.tsx
    │   │   └── ClassInformationPresenter.tsx
```

```
    │   ├── Common
    │   │   ├── Footer
    │   │   │   ├── Footer.style.ts
    │   │   │   └── Footer.tsx
    │   │   └── Header
    │   │       ├── Header.style.ts
    │   │       └── Header.tsx
    │   ├── Community
    │   │   ├── Community.style.ts
    │   │   ├── CommunityContainer.tsx
    │   │   └── ContainerPresenter.tsx
    │   ├── Creator
    │   │   ├── Creator.style.ts
    │   │   ├── CreatorContainer.tsx
    │   │   └── CreatorPresenter.tsx
    │   ├── Curriculum
    │   │   ├── Curriculum.style.ts
    │   │   ├── CurriculumContainer.tsx
    │   │   └── CurriculumPresenter.tsx
    │   ├── Kit
    │   │   ├── Kit.style.ts
    │   │   ├── KitContainer.tsx
    │   │   └── KitPresenter.tsx
    │   ├── Recommend
    │   │   ├── Recommend.style.ts
    │   │   ├── RecommendCotainer.tsx
    │   │   └── RecommnedPresenter.tsx
    │   ├── RefundPolicy
    │   │   ├── RefundPolicy.style.ts
    │   │   ├── RefundPolicyContainer.tsx
    │   │   └── RefundPolicyPresenter.tsx
    │   ├── Review
    │   │   ├── Review.style.ts
    │   │   ├── ReviewContainer.tsx
    │   │   └── ReviewPresenter.tsx
    │   └── SideNavigationBar
    │       ├── SideNavigationBar.style.ts
    │       ├── SideNavigationBarContainer.tsx
    │       └── SideNavigationBarPresenter.tsx
    ├── api
    │   └── classInformation.ts
    ├── assets  // 이미지 파일 등 관리
    ├── data
    │   └── classInformation.json
    ├── globalStyle
    │   └── globalStyle.ts
    ├── index.tsx
    ├── react-app-env.d.ts
    └── store
        ├── reducers
        │   ├── ClassInformation.ts
        │   └── index.ts
        └── sagas
            ├── ClassInformation.ts
            └── index.ts
```

🎨 Atomic Design Pattern

➡️ 동일하게 Class 101 사이트에서의 클래스 상세 페이지에 대한 구조 설계를 진행했습니다.

atomic design pattern으로 프로젝트 설계를 하는데 시간이 오래걸리는 것 같습니다. atomic design pattern에서는 최소 단위인 atoms를 어떻게 구성을 해야될 지도 고려해봐야 하고 각 구조 단위를 사용하는 것이 효율적인지를 생각해봐야 합니다.상태관리에 관하여 개인적인 생각으로는 수행하는 각 프로젝트마다 다를 수 있지만 oraganisms에서 상태관리 로직을 구성하는게 좋다고 생각이 듭니다.최상위 구조인 pages에서 상태관리 로직을 구성하게 되면 props를 통해서 하위 구조에 계속 내려줘야 하기 때문에 복잡해질 수 있을 것이라고 생각합니다.이 부분에 대해서는 프로젝트 그리고 각 회사의 팀마다 유동적으로 조절할 수 있을 것입니다.

```
.
└── src
    ├── @types
    │   ├── dataTypes.ts
    │   └── handlerTypes.ts
    ├── App.tsx
    ├── Modal
    │   ├── atoms
    │   ├── molecules
    │   ├── organisms
    │   └── templates
    ├── _components
    │   ├── UI
    │   │   ├── atoms
    │   │   │   ├── Button
```

```
    │   │   │   │   ├── Button.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── Icon
    │   │   │   │   ├── Icon.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── Image
    │   │   │   │   ├── Image.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── Input
    │   │   │   │   ├── Input.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── Text
    │   │   │   │   ├── Test.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   └── index.ts
    │   │   ├── molecules
    │   │   │   ├── Banner
    │   │   │   │   ├── Banner.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── ClassIntroduceContent
    │   │   │   │   ├── ClassIntroduceContent.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── ClassIntroduceTitle
    │   │   │   │   ├── ClassIntroduceTitle.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── ClassSimpleInforrmation
    │   │   │   │   ├── ClassSimpleInformation.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── CommentInput
    │   │   │   │   ├── CommentInput.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── CommunityTitle
    │   │   │   │   ├── CommunityTitle.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── CurriculumContentList
    │   │   │   │   ├── CurriculumContentList.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── CurriculumTitle
    │   │   │   │   ├── CurriculumTitle.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── FooterInformation
    │   │   │   │   ├── FooterInformation.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── KitIntroduceContent
    │   │   │   │   ├── KitIntroduceContent.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── KitIntroduceTitle
    │   │   │   │   ├── KitIntroduceTitle.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── RecommendInformation
    │   │   │   │   ├── RecommendInformation.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── RecommendPrice
    │   │   │   │   ├── RecommendPrice.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── ReviewGrid
    │   │   │   │   ├── ReviewGrid.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── ReviewList
    │   │   │   │   ├── ReviewList.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SearchInput
    │   │   │   │   ├── SearchInput.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SectionNavBar
    │   │   │   │   ├── SectionNavBar.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SideNavBarButton
    │   │   │   │   ├── SideNavBarButton.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SideNavBarOption
    │   │   │   │   ├── SideNavBarOption.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SideNavBarPrice
    │   │   │   │   ├── SideNavBarPrice.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── SideNavbarTitle
    │   │   │   │   ├── SideNavbarTitle.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   ├── UserInformation
    │   │   │   │   ├── UserInformation.style.ts
    │   │   │   │   └── index.tsx
    │   │   │   └── index.ts
    │   │   └── organisms
    │   │       ├── ClassIntroduceSection
    │   │       │   ├── ClassIntroduceSection.style.ts
    │   │       │   └── index.tsx
```

```
    │   │       ├── ClassKitIntroduceSection
    │   │       │   ├── ClassKitIntroductSection.style.ts
    │   │       │   └── index.tsx
    │   │       ├── CommentSection
    │   │       │   ├── CommentSection.style.ts
    │   │       │   └── index.tsx
    │   │       ├── CommunitySection
    │   │       │   ├── CommunitySection.style.ts
    │   │       │   └── index.tsx
    │   │       ├── CurriculumSection
    │   │       │   ├── CurriculumSection.style.ts
    │   │       │   └── index.tsx
    │   │       ├── Footer
    │   │       │   ├── Footer.style.ts
    │   │       │   └── index.tsx
    │   │       ├── Header
    │   │       │   ├── Header.style.ts
    │   │       │   └── index.tsx
    │   │       ├── RecommendSection
    │   │       │   ├── RecommendSection.style.ts
    │   │       │   └── index.tsx
    │   │       ├── SideNavBarSection
    │   │       │   ├── SideNavNarSection.style.ts
    │   │       │   └── index.tsx
    │   │       └── index.ts
    │   ├── pages
    │   │   ├── ClassDetail
    │   │   │   ├── ClassDetail.style.ts
    │   │   │   └── index.tsx
    │   │   └── index.tsx
    │   └── templates
    │       ├── ClassInformation
    │       │   └── index.tsx
    │       ├── Community
    │       │   └── index.tsx
    │       ├── SideNavBar
    │       │   └── index.tsx
    │       └── index.ts
    ├── api
    │   └── classInformation.ts
    ├── assets
    ├── data
    │   └── classInformation.json
    ├── globalStyle
    │   └── globalStyle.ts
    ├── index.tsx
    ├── react-app-env.d.ts
    └── store
        ├── reducers
        │   ├── ClassInformation.ts
        │   └── index.ts
        └── sagas
            ├── ClassInformation.ts
            └── index.ts
```