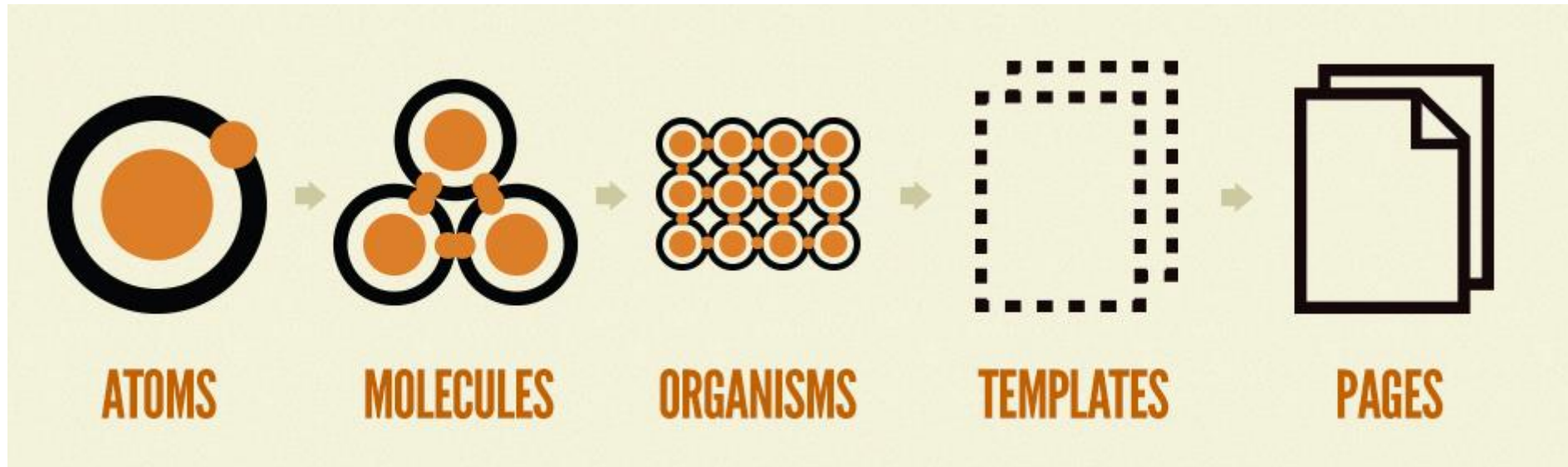


Tools

아토믹 디자인(Atomic Design)

- 아토믹 디자인(Atomic Design)은 Brad Frost가 제안한 디자인 방법론으로,
UI를 더 작고 재사용 가능한 컴포넌트로 나누어 설계하는 접근 방식
- 화학의 원자 구조에서 영감을 받아, 디자인 시스템을 더 체계적이고 일관되게 구축할 수 있도록 도와줌
- 아토믹 디자인은 5가지 단계로 구성



아토믹 디자인(Atomic Design)

1. 원자 (Atoms)

- 설명: 원자는 UI의 가장 기본적인 요소로, 더 이상 분해할 수 없는 최소 단위
- 예시: 버튼, 입력 필드, 라벨 등
- 역할: 원자는 독립적이고 재사용 가능하며, 단일 책임 원칙을 준수하여 복잡성을 줄일 수 있음

2. 분자 (Molecules)

- 설명: 분자는 원자들이 결합된 단위로, 단일 기능을 수행하는 보다 복잡한 구성 요소
- 예시: 검색 창(입력 필드 + 버튼), 카드 컴포넌트(이미지 + 텍스트 + 버튼) 등
- 역할: 분자를 통해 원자를 조합하여 단일 책임을 가진 더 큰 기능적 단위를 만들 수 있음

3. 유기체 (Organisms)

- 설명: 유기체는 여러 분자들이 결합하여 보다 복잡한 인터페이스를 구성하는 요소
- 예시: 네비게이션 바, Header, Footer 등
- 사용 이유: 유기체는 애플리케이션의 주요 섹션을 구성하며, 다양한 분자들이 함께 작동하여 특정 기능을 수행

아토믹 디자인(Atomic Design)

4. 템플릿 (Templates)


- 설명: 템플릿은 유기체들의 레이아웃을 정의하는 구조
- 예시: 페이지 레이아웃, 대시보드 레이아웃 등
- 사용 이유: 템플릿을 통해 전체 레이아웃을 정의하고, 유기체와 분자가 어떻게 배치될지를 결정
이는 디자인의 일관성을 유지하고, 구조를 명확히 할 수 있음

5. 페이지 (Pages)

- 설명: 페이지는 실제 콘텐츠를 포함하여 완성된 UI를 나타내는 단계
- 예시: 홈 페이지, 프로필 페이지 등
- 사용 이유: 페이지는 템플릿에 실제 데이터를 결합하여 최종적인 결과물을 생성
이는 디자인 시스템의 최종 목표인 사용자 인터페이스를 완성하는 단계

아토믹 디자인(Atomic Design)

- 원자 (Atoms)



```
// Button.js
import React from 'react';

const Button = ({ children }) => {
  return <button>{children}</button>;
};

export default Button;
```

아토믹 디자인(Atomic Design)

- 분자 (Molecules)

```
// SearchBox.js
import React from 'react';
import Input from './Input'; // 원자
import Button from './Button'; // 원자

const SearchBox = () => {
  return (
    <div>
      <Input type="text" placeholder="Search..." />
      <Button>Search</Button>
    </div>
  );
};

export default SearchBox;
```

아토믹 디자인(Atomic Design)

- 유기체 (Organisms)

```
// Header.js
import React from 'react';
import Logo from './Logo'; // 원자
import Navigation from './Navigation'; // 분자
import SearchBox from './SearchBox'; // 분자

const Header = () => {
  return (
    <header>
      <Logo />
      <Navigation />
      <SearchBox />
    </header>
  );
};

export default Header;
```

아토믹 디자인(Atomic Design)

- 템플릿 (Templates)

```
// MainTemplate.js
import React from 'react';
import Header from './Header'; // 유기체
import Footer from './Footer'; // 유기체

const MainTemplate = ({ children }) => {
  return (
    <div>
      <Header />
      <main>{children}</main>
      <Footer />
    </div>
  );
};

export default MainTemplate;
```


아토믹 디자인(Atomic Design)

- 페이지 (Pages)

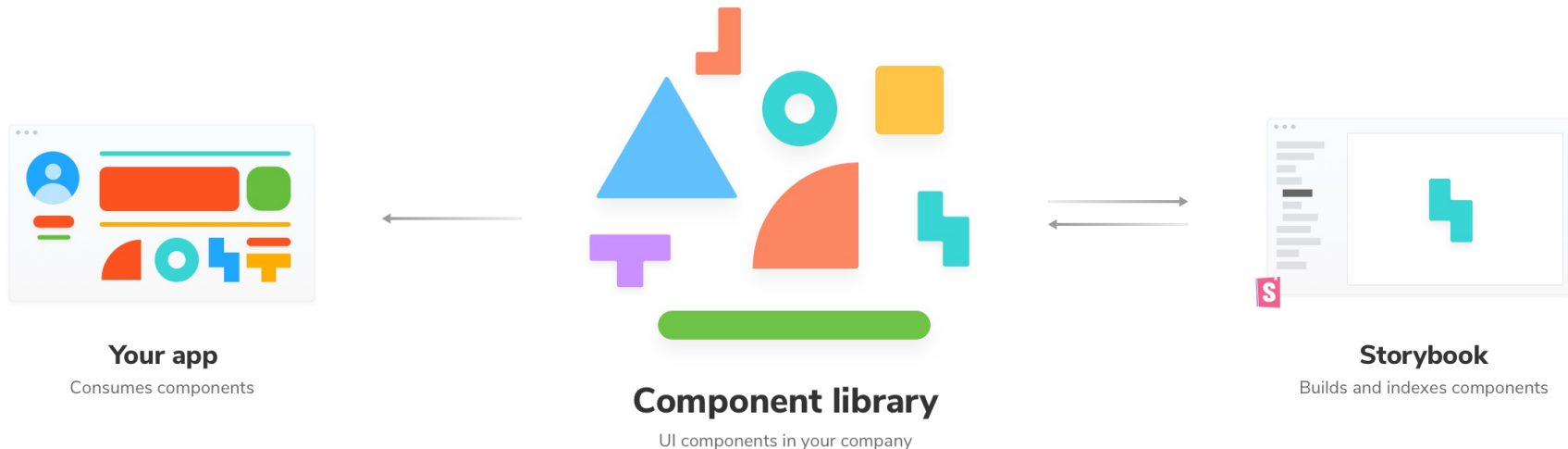
```
// HomePage.js
import React from 'react';
import MainTemplate from './MainTemplate'; // 템플릿
import HeroSection from './HeroSection'; // 유기체
import FeaturedProducts from './FeaturedProducts'; // 유기체

const HomePage = () => {
  return (
    <MainTemplate>
      <HeroSection />
      <FeaturedProducts />
    </MainTemplate>
  );
};

export default HomePage;
```

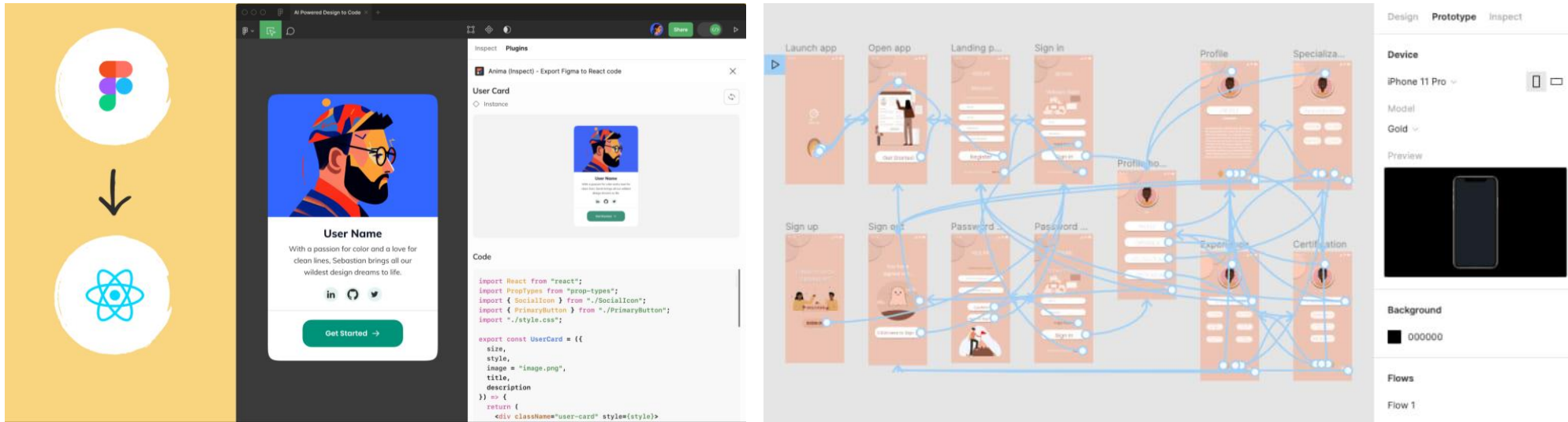
스토리북 (Storybook)

- 스토리북(Storybook)은 React 컴포넌트 개발을 위한 독립 실행형 도구로, 컴포넌트를 독립적으로 개발하고 문서화하며 테스트할 수 있는 환경을 제공
- 이는 컴포넌트 기반 개발을 촉진하고, UI 개발의 생산성을 높이는 데 매우 유용함
- 스토리북 장점
 - 컴포넌트 중심 개발: 컴포넌트를 독립적으로 개발하고 테스트할 수 있어 개발 효율성을 높일 수 있음
 - 자동화된 문서화: 컴포넌트의 사용 방법과 다양한 상태를 문서화하여 공유 가능
 - 디자인 시스템: 일관된 UI 컴포넌트를 유지하고 관리하는 데 유용
 - 강력한 커뮤니티와 애드온: 다양한 애드온을 통해 기능을 확장하고, 커뮤니티에서 제공하는 여러 도구를 활용 가능



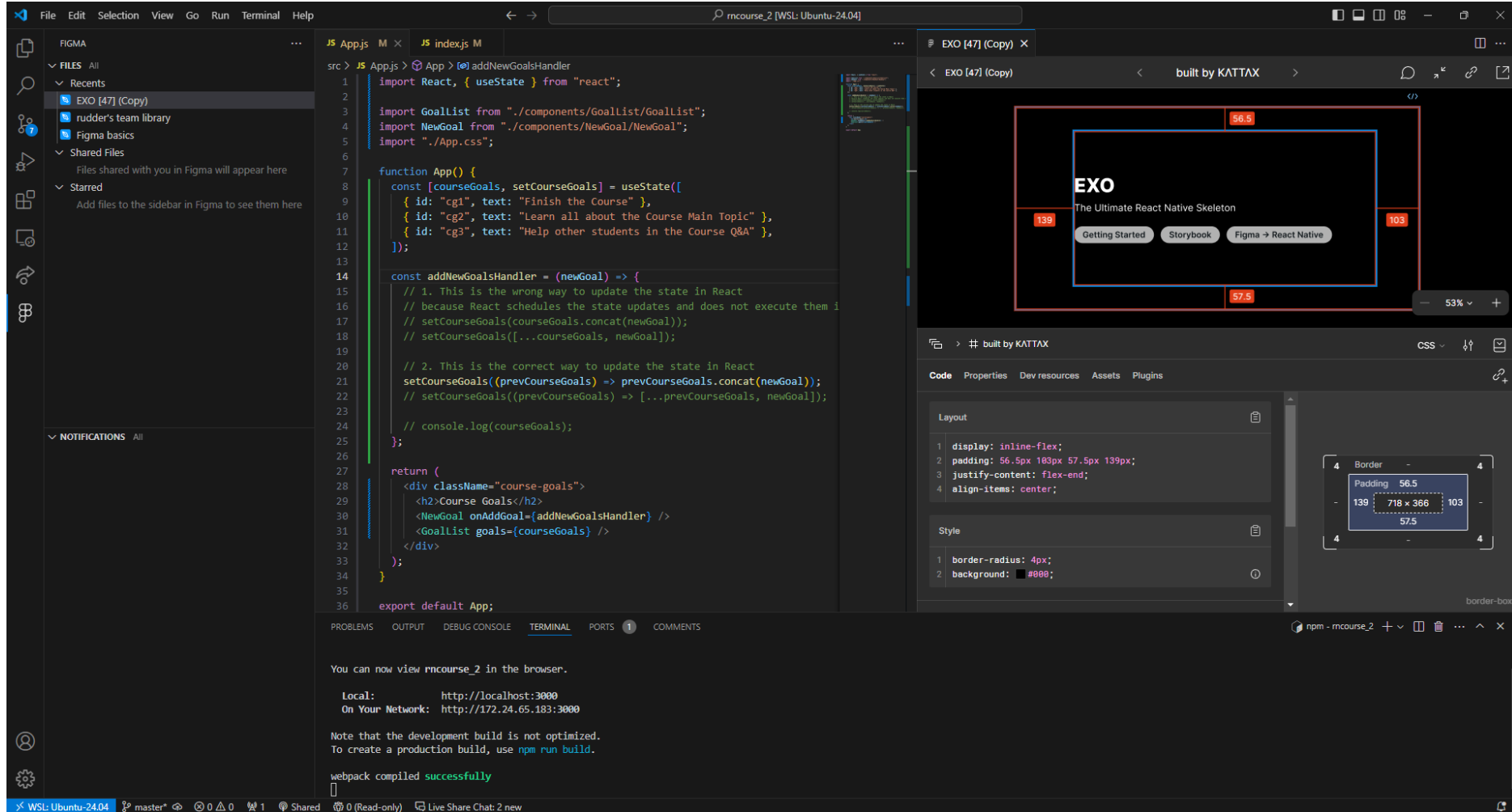
피그마 (Figma)

- Figma는 UI/UX 디자인을 위한 협업 도구로, 디자이너와 개발자가 함께 작업할 수 있는 플랫폼
- Figma는 벡터 그래픽 편집 및 프로토타이핑 기능을 제공하며, 실시간 협업과 클라우드 기반 저장소를 제공
- Figma 주요 장점:
 - 협업 강화: 실시간 협업 기능을 통해 디자이너와 개발자가 동시에 작업할 수 있어 커뮤니케이션이 원활
 - 디자인 시스템 구축: 재사용 가능한 컴포넌트와 스타일을 통해 일관된 디자인 시스템을 구축하고 유지 가능
 - 클라우드 접근성: 클라우드 기반으로 언제 어디서나 접근 가능하며, 자동 저장 및 버전 관리로 안전하게 작업 가능
 - 프로토타이핑: 인터랙티브한 프로토타입을 만들어 사용자 흐름을 시뮬레이션하고 테스트 가능
 - 디자인에서 코드로의 전환: 다양한 플러그인과 도구를 통해 디자인을 React 코드로 쉽게 변환 가능



Figma VS Code Extension

- VS Code extension 제공해 기존 프로젝트 로딩 및 스타일 복사 등의 편의성 제공



Figma Plugins

- Anima(모든 Figma 구성 요소를 React 코드로 변환) 및 React Native 등의 플러그인 제공

