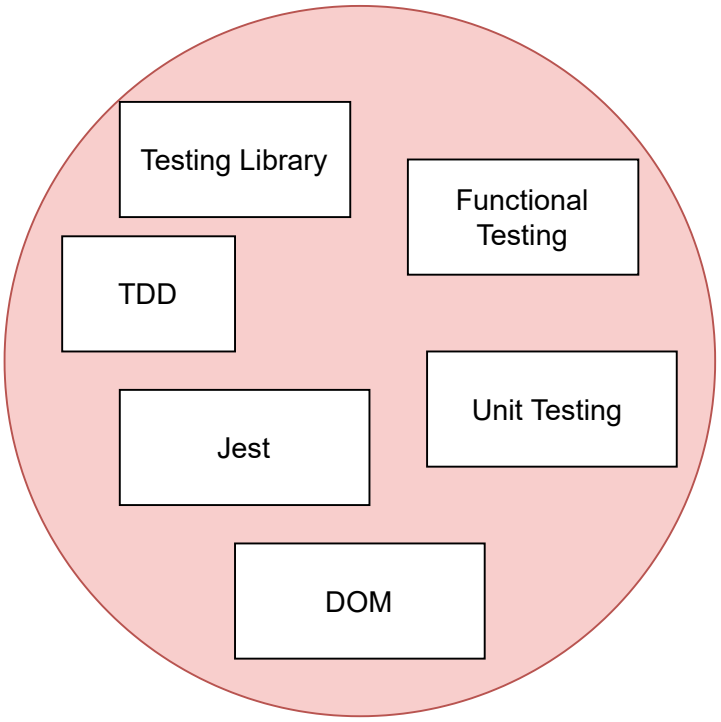
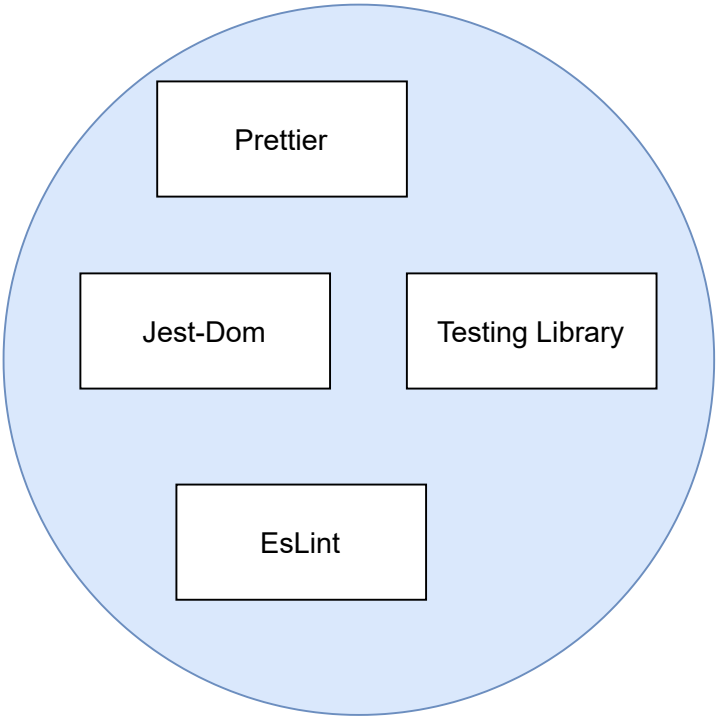
강의 구성

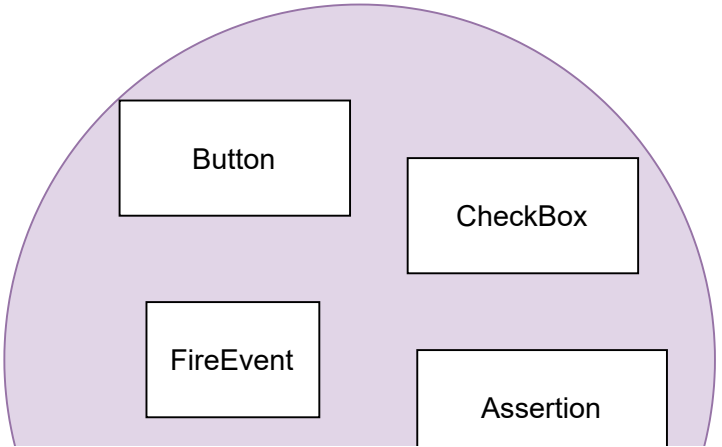
Testing 에 대한 소개

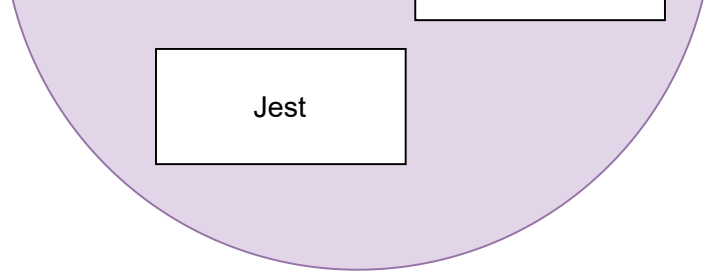


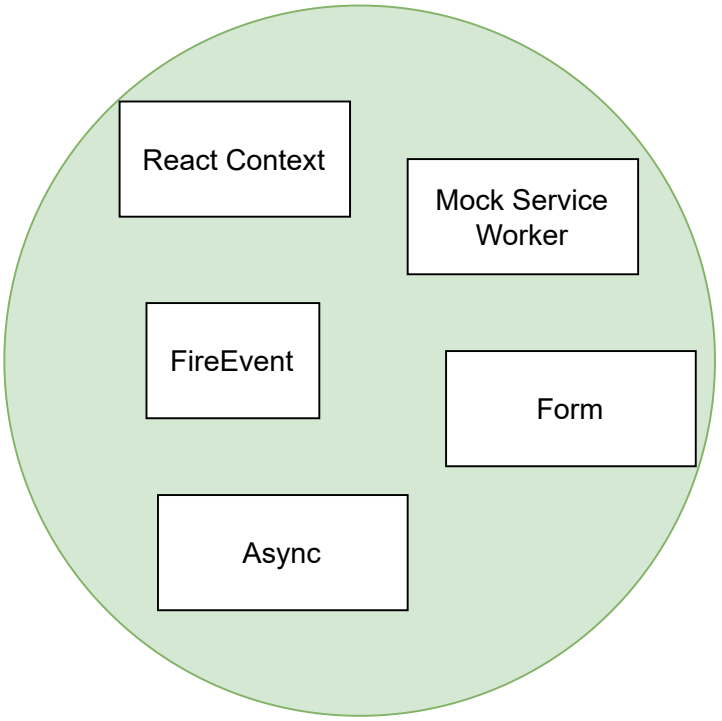
Testing을 위한 설정 및 준비



간단한 앱을 만들면서 테스팅 적용







좀 더 복잡한 앱을
만들면서
테스팅 적용

테스팅의 이유

왜 어플리케이션을 TEST 해야 할까요?

간단하게 더 안정적인 어플리케이션을 위해서는 여러 방법으로 테스트를 해줘야

더 안정적인 어플리케이션이 될 수 있습니다.

테스팅으로 얻는 이점은 무엇일까요?

1. 디버깅 시간을 단축! 만약 데이터가 잘못 나왔다면 그것이 UI의 문제인지 DB의 문제인지등 전부 테스트를 해봐서 찾아야 하는데 테스팅 환경이 구축되어있다면 자동화 된 유닛 테스팅으로 특정 버그를 쉽게 찾아 낼 수있습니다.

2. 더욱 안정적인 어플리케이션! 많은 테스트 코드와 함께 작성된 코드의 어플리케이션이 되기 때문에 훨씬 안정적인 어플리케이션이 됩니다.

3. 이밖에도 재설계 시간의 단축, 추가로 무언가를 더 구현해야 할 때 더 용이하게 할 수 있는 등의 이점들이 있습니다.

React Testing Library

Create React App 로 리액트 앱을 생성하면 기본적으로 테스팅 할 때 React Testing Library를 사용하는 것을 볼 수 있습니다. 그럼 이 React Testing Library가 무엇일까요?

React Testing Library란 무엇인가요?

React Testing Library는 React 구성 요소 작업을 위한 API를 추가하여 DOM Testing Library 위에 구축됩니다.

DOM Testing Library란 Dom 노드(Node)를 테스트하기 위한 매우 가벼운 솔루션입니다.

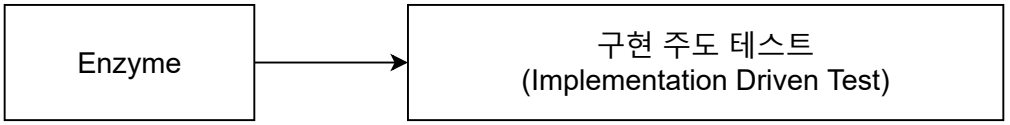
Create React App으로 생성된 프로젝트는 즉시 React Testing Library를 지원합니다. 그렇지 않은 경우 다음과 같이 npm을 통해 추가할 수 있습니다.

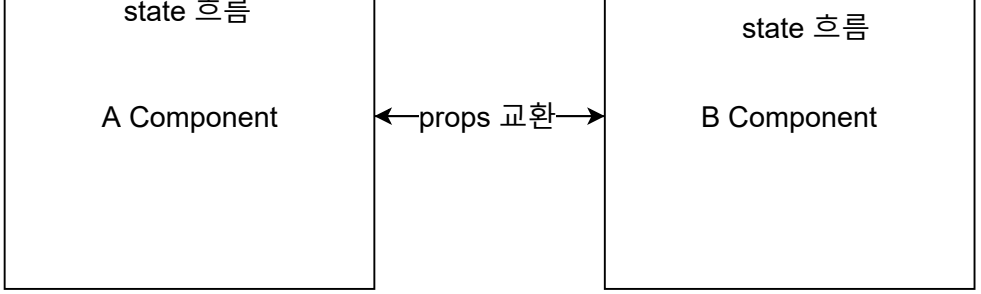
ddfsdf

```
npm install --save-dev @testing-library/react
```

리액트 컴포넌트를 테스트하는 가벼운 솔루션!

React Testing Library는 에어비앤비에서 만든 Enzyme을 대처하는 솔루션 입니다.
Enzyme이 구성 요소의 구현 세부 정보를 테스트하는 대신
React Testing Library 개발자를 React 애플리케이션의 사용자 입






```
<p>
Edit <code>src/App.js</code> and save to
reload....
</p>
```

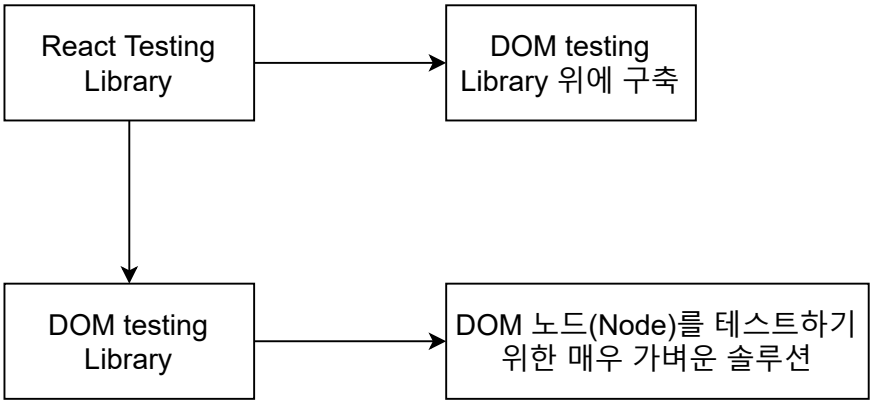
구현 주도 테스트에서는 위의 UI를 테스트할 때 주로 <p> 태그가 쓰였고 Edit 등의 문자가 들어갔다는것을 테스트합니다. 그래서 만약 <p>태그를 <h2> 태그로 바뀌면 에러가 날 것입니다.

하지만 행동 주도 테스트에서는 사용자의 입장에서 테스트 하기에 <p>태그가 쓰이던 <h3> 태그가 쓰여서 글을 표현하는지가 중요한지 보다 어떠한 이벤트를 발생시켰을때 화면이 어떻게 변화가 되는지 같은 테스트가 더 주를 이루게 됩니다.

이렇게 가볍게 알아보고 실제 사용하면서 더 자세히 알아보겠습니다

DOM 이란?

앞서 React Testing Library를 설명할 때 DOM에 관한 설명이 나와서 DOM에 대해서 좀 더 자세히 알아보고 가겠습니다.



DOM (Document Object Model)이란 ?

Document: 문서
Object: 객체
Model: 모델

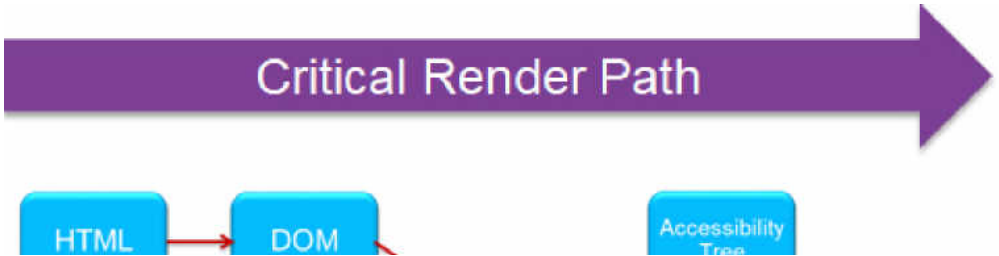
DOM(문서 객체 모델)은 XML, HTML 문서의 각 항목을 계층으로 표현하여 생성, 변형 삭제할 수 있도록 돕는 인터페이스이다. - 위키피디아

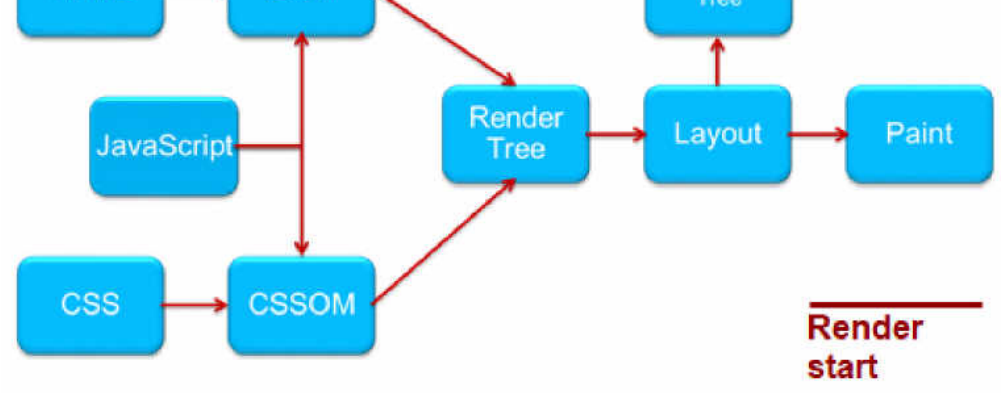
더 자세히 이해하기 위해서 웹페이지의 빌드과정을 보겠습니다.

웹 페이지 빌드 과정(Critical Rendering Path CRP)

브라우저가 서버에서 페이지에 대한 HTML 응답을 받고 화면에 표시하기 전에 여러 단계가 있습니다.

웹 브라우저가 HTML 문서를 읽고, 스타일 입히고 뷰포트에 표시하는 과정입니다





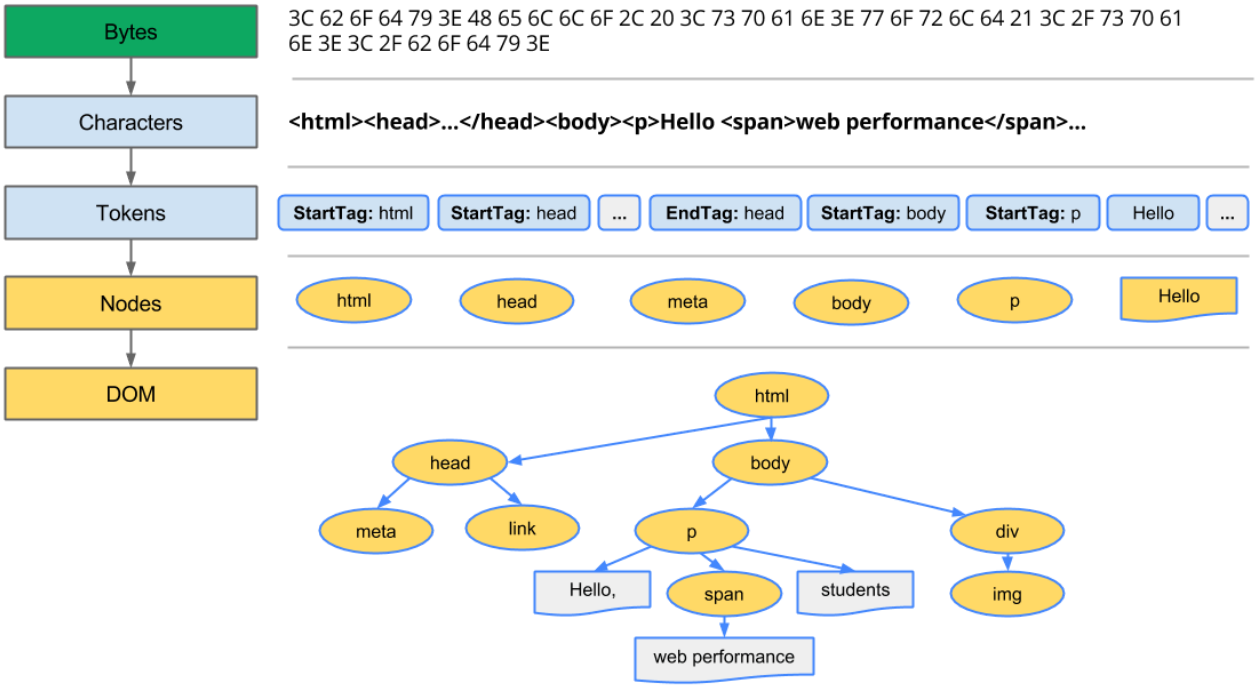
출처 dimension85.com

문서를 읽어들여서 그것들을 파싱하고 어떤 내용을 페이지에 렌더링할 지 결정합니다.
(HTML, CSS) + javascript

이 단계는 브라우저가 DOM과 CSSOM을 결합하는 곳이며, 이 프로세스는 화면에 보이는 모든 콘텐츠의 콘텐츠와 스타일 정보를 모두 포함하는 최종 렌더링 트리를 출력합니다. 즉 화면에 표시되는 모든 노드의 콘텐츠 및 스타일 정보를 포함합니다.

이 단계는 브라우저가 페이지에 표시되는 각 요소의 크기와 위치를 계산하는 단계입니다.

페인트 단계에 도달하면 브라우저는 레이아웃 결과를 선택하고 픽셀을 화면에 페인트해야 합니다.



출처: janpierrsanchez.medium.com

DOM

HTML 요소들의 구조화된 표현

DOM은 HTML이 브라우저의 렌더링 엔진에 의해 분석되고 분석이 모두 끝나고난 HTMl 파일이 DOM입니다.

HTML은 화면에 보이고자 하는 모양과 구조를 문서로 만들어서 단순 텍스트로 구성되어있으며 DOM은 HTML문서의 내용과 구조가 객체 모델로 변화되어 다양한 프로그램에서 사용될 수 있습니다.

HTML 문서가 유효하지 않게 작성됐을때는 브라우저가 올바르게 교정해주며, DOM은 자바스크립트에 의해 수정될 수 있습니다. 하지만 HTML은 수정하지 않습니다.

Create React App으로 리액트 시작하기

원래 리액트 앱을 처음 실행 하기 위해선 webpack 이나 babel같은 것을 설정하기 위해서 엄청 나게 많은 시간이 걸렸다.

Babel

Webpack

최신 자바스크립트 문법을 지원하지 않는 브라우저들을 위해서 최신 자바스크립트 문법을 구형 브라우저에서도 돌수있게 변환 시켜줌.

At its core, **webpack** is a *static module bundler* for modern JavaScript applications. When webpack processes your application, it internally builds a dependency graph which maps every module your project needs and generates one or more *bundles.*

하지만 이제는 create-react-app Command 로
바로 시작할수 있다.

npx create-react-app <프로젝트 이름>

원래 create-react-app을 할때
npm install -g create-react-app
이렇게 했었다. global 디렉토리에 다운받음

npx create-react-app <프로젝트 디렉토리 이름>

NPM (**Node package manager**)

What is NPM ?

it is an online repository for the publishing of open-source Node.js projects

it is a command-line utility for interacting with the said repository that aids in package installation, version management, and dependency management

NPM INSTALL LOCALLY !

Links created at the ***./node_modules/.bin*** directory

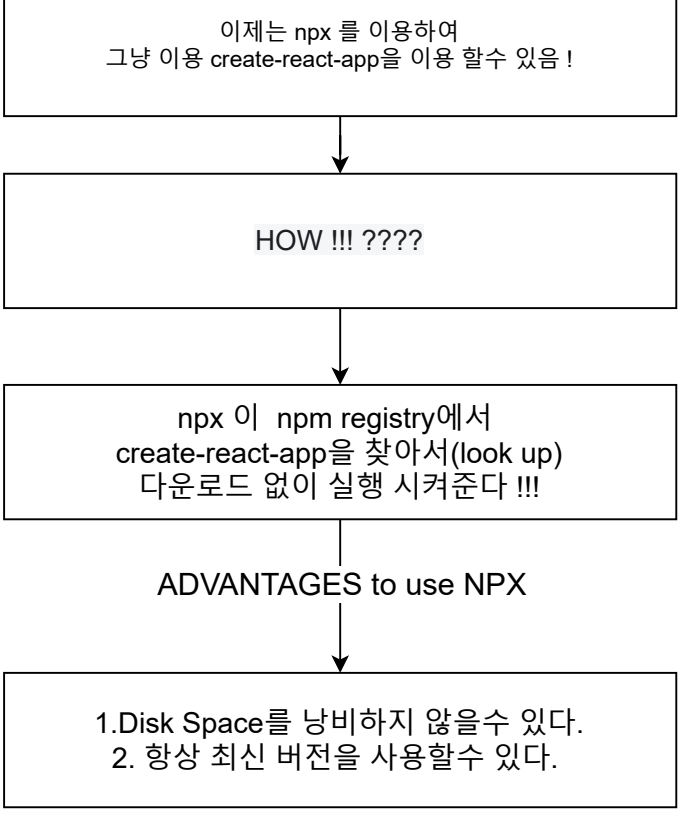
NPM INSTALL Globally !

Links created from the global ***bin/*** directory
ex) ***/usr/local/bin*** on Linux
%AppData%/npm on Windows

SO,

만약 Install 하는 NPM을 다른 프로젝트에서 쓰지 않는 다면 Global로 Install 할 필요가 없음 !
그러므로 Disk Space를 낭비하지 않을수 있다 !

원래 create-react-app을 할때
npm install -g create-react-app
이렇게 했었다. global 디렉토리에 다운받음


이제는 npx 를 이용하여
그냥 이용 create-react-app을 이용 할수 있음 !
HOW !!! ????
npx 이 npm registry에서
create-react-app을 찾아서(look up)
다운로드 없이 실행 시켜준다 !!!
ADVANTAGES to use NPX
1.Disk Space를 낭비하지 않을수 있다.
2. 항상 최신 버전을 사용할수 있다.

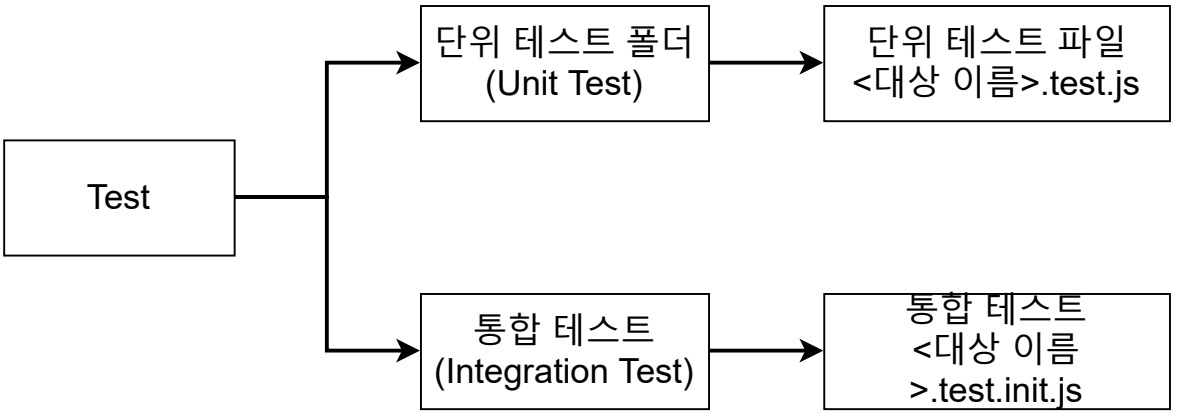
Jest에 대하여

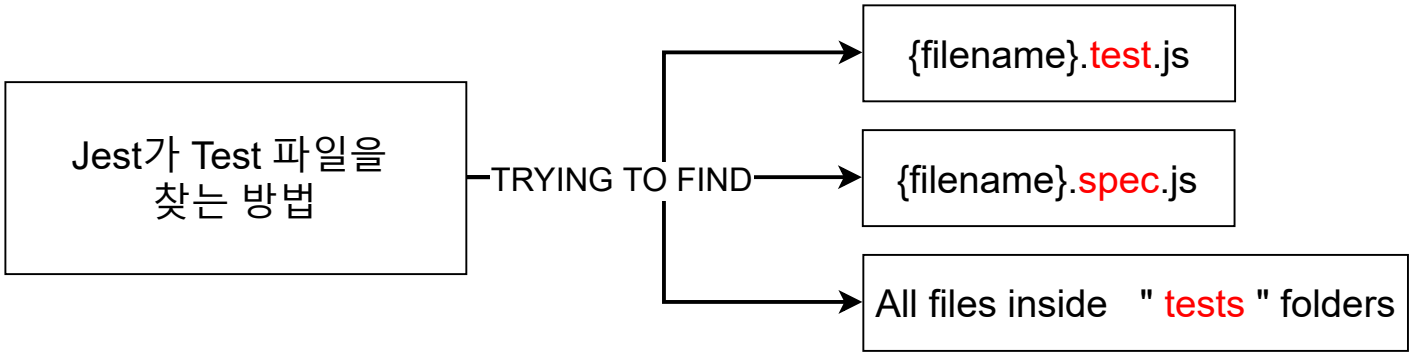
Jest란 무엇인가요?

FaceBook에 의해서 만들어진 테스팅 프레임 워크입니다.
최소한의 설정으로 동작하며 Test Case 를 만들어서 어플리케이션 코드가
잘 돌아가는지 확인해줍니다.
단위 (Unit) 테스트를 위해서 이용합니다.

Jest 시작하기

1. Jest 라이브러리 설치 npm install jest --save-dev
2. Test 스크립트 변경 "test" : "jest" or "jest --watchAll
3. 테스트 작성할 폴더 및 파일 기본 구조 생성




Jest가 Test 파일을
찾는 방법
TRYING TO FIND
{filename}.test.js
{filename}.spec.js
All files inside " tests " folders

Jest 파일 구조 & 사용법

Jest 파일 구조

```
describe("Product Controller create", () => {
  beforeEach(() => {
    req.body = newProduct;
  });
  it("should have a createProduct function", () => {
    expect(typeof productController.createProduct).toBe("function");
  });
  it("should call Product.create", async () => {
    await productController.createProduct(req, res, next);
    expect(Product.create).toBeCalledWith(newProduct);
  });
  it("should return 201 response code", async () => {
    await productController.createProduct(req, res, next);
    expect(res.statusCode).toBe(201);
    expect(res._isEndCalled()).toBeTruthy();
  });
  it("should return json body in response", async () => {
    Product.create.mockReturnValue(newProduct);
    await productController.createProduct(req, res, next);
    expect(res._getJSONData()).toStrictEqual(newProduct);
  });
  it("should handle errors", async () => {
    const errorMessage = { message: "Done property missing" };
    const rejectedPromise = Promise.reject(errorMessage);
    Product.create.mockReturnValue(rejectedPromise);
    await productController.createProduct(req, res, next);
    expect(next).toBeCalledWith(errorMessage);
  });
});
```

describe

test (it)

test (it)

test (it)

Explain

"**describe**"
argument (name, fn)

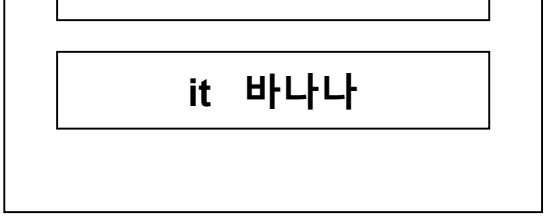
여러 관련 테스트를
그룹화하는 블록을 만듭니다.

"**it**" same as **test**
argument (name, fn, timeout)

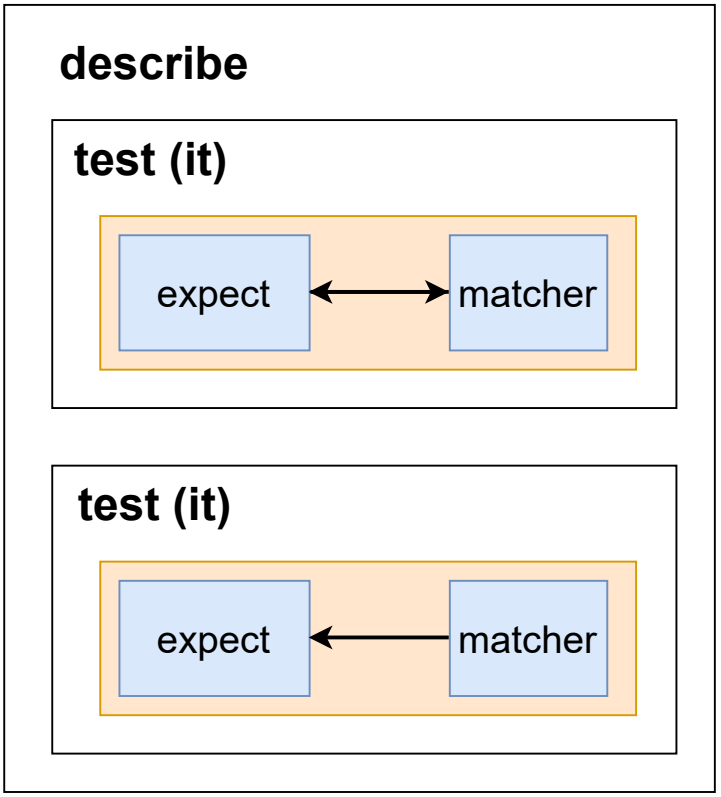
개별 테스트를 수행하는 곳.
각 테스트를 작은 문장처럼
설명합니다.

Describe (과일)

it 사과




```
describe("Product Controller create", () => {
  beforeEach(() => {
    req.body = newProduct;
  });
  it("should have a createProduct function", () => {
    expect(typeof productController.createProduct).toBe("function");
  });
  it("should call Product.create", async () => {
    await productController.createProduct(req, res, next);
    expect(Product.create).toBeCalledWith(newProduct);
  });
  it("should return 201 response code", async () => {
    await productController.createProduct(req, res, next);
    expect(res.statusCode).toBe(201);
    expect(res._isEndCalled()).toBeTruthy();
  });
  it("should return json body in response", async () => {
    Product.create.mockReturnValue(newProduct);
    await productController.createProduct(req, res, next);
    expect(res._getJSONData()).toStrictEqual(newProduct);
  });
  it("should handle errors", async () => {
    const errorMessage = { message: "Done property missing" };
    const rejectedPromise = Promise.reject(errorMessage);
    Product.create.mockReturnValue(rejectedPromise);
    await productController.createProduct(req, res, next);
    expect(next).toBeCalledWith(errorMessage);
  });
});
```



Explain

"expect"

expect 함수는 값을
테스트할 때마다 사용됩니다.
그리고 expect 함수 혼자서는
거의 사용 되지 않으며 matcher와
함께 사용됩니다.

"matcher"

다른 방법으로 값을 테스트 하도록
"매처"를 사용합니다.

```
test('two plus two is four', () => {
  expect(2 + 2).toBe(4);
});
```

matcher

```
PASS test/unit/products.test.js
  ✓ two plus two is four (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.279 s, estimated 1 s
Ran all test suites.

Watch Usage: Press w to show more.
```

```
test('two plus two is not five', () => {
  expect(2 + 2).not.toBe(5);
});
```

matcher

```
PASS test/unit/products.test.js
  ✓ two plus two is four (2 ms)
  ✓ two plus two is not five (1 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.281 s, estimated 1 s
Ran all test suites.

Watch Usage: Press w to show more.
```

React Testing Library 주요 API

앱을 키고 기본 테스트 진행 → a

q quit

App.test.js → 기본 테스트가 진행되는 곳

```
import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

"render" 함수

DOM에 컴포넌트를 랜더링하는 함수

인자로 랜더링할 React 컴포넌트가 들어감

Return은 RTL에서 제공하는 쿼리 함수와 기타 유틸리티 함수를 담고 있는 객체를 리턴(Destructuring 문법으로 원하는 쿼리 함수만 얻어올 수 있다.)
====> 소스 코드가 복잡해지면 비추천 !!! screen 객체를 사용하기
왜냐면 사용해야 할 쿼리가 많아질수록 코드가 복잡해질 수 있음

```
test('renders learn react link', () => {
  const { getByText } = render(<App />);
```

```
  const linkElement = getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

쿼리 함수에 대해서

공식 문서 → https://testing-library.com/docs/queries/about/

쿼리 함수란 ?

쿼리는 페이지에서 요소를 찾기 위해 테스트 라이브러리가 제공하는 방법입니다. 여러 유형의 쿼리("get", "find", "query")가 있습니다. 이들 간의 차이점은 요소가 발견되지 않으면 쿼리에서 오류가 발생하는지 또는 Promise를 반환하고 다시 시도하는지 여부입니다. 선택하는 페이지 콘텐츠에 따라 다른 쿼리가 다소 적절할 수 있습니다.

```
import {render, screen} from '@testing-library/react'

test('should show login form', () => {
  render(<Login />)
  const input = screen.getByLabelText('Username')
  // Events and assertions...
})
```

get, find, query 의 차이점

쿼리는 페이지에서 요소를 찾기 위해 테스트 라이브러리가 제공하는 방법입니다. 여러 유형의 쿼리("get", "find", "query")가 있습니다. 이들 간의 차이점은 요소가 발견되지 않으면 쿼리에서 오류가 발생하는지 또는 Promise를 반환하고 다시 시도하는지 여부입니다. 선택하는 페이지 콘텐츠에 따라 다른 쿼리가 다소 적절할 수 있습니다.

"getBy..."	"queryBy..."	"findBy..."
쿼리에 대해 일치하는 노드를 반환하고 일치하는 요소가 없거나 둘 이	쿼리에 대해 일치하는 노드를 반환하고 일치하는 요소가 없으면 null을 반환합니다. 이것은 존재하지 않	주어진 쿼리와 일치하는 요소가 발견되면 Promise를 반환합니다. 요소가 발견되지 않거나 기본 제한 시간

상의 일치가 발견되면 실행 오류를 발생시킵니다(둘 이상의 요소가 예상되는 경우 대신 getAllBy 사용).

는 요소를 어설션하는 데 유용합니다. 둘 이상의 일치 항목이 발견되면 오류가 발생합니다(확인된 경우 대신 queryAllBy 사용).

인 1000ms 후에 둘 이상의 요소가 발견되면 약속이 거부됩니다. 둘 이상의 요소를 찾아야 하는 경우 findAllBy를 사용하십시오.

getBy + waitFor = findBy

Type of Query	0 Matches	1 Match	>1 Matches	Retry (Async/Await)
Single Element				
`getBy...`	Throw error	Return element	Throw error	No
`queryBy...`	Return `null`	Return element	Throw error	No
`findBy...`	Throw error	Return element	Throw error	Yes
Multiple Elements				
`getAllBy...`	Throw error	Return array	Return array	No
`queryAllBy...`	Return `[]`	Return array	Return array	No
`findAllBy...`	Throw error	Return array	Return array	Yes

"waitFor"

일정 기간 동안 기다려야 할 때 waitFor를 사용하여 기대가 통과할 때까지 기다릴 수 있습니다.