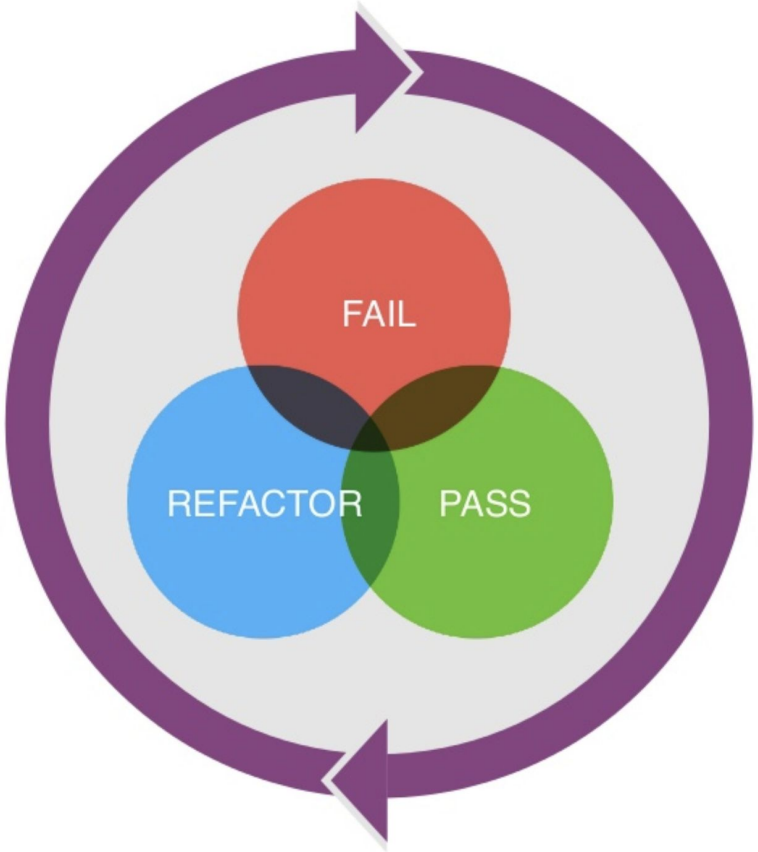
React
Test Driven Development

Writing Unit Tests ~~**Nice to Have**~~

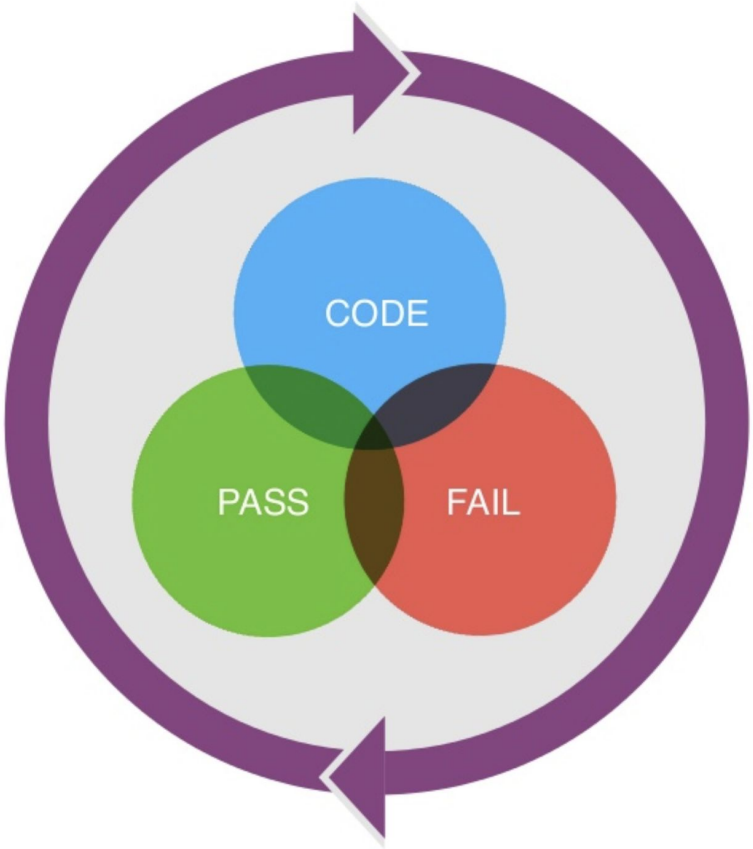
Is Must

Test-Driven Development



1. *Write Tests*
2. *Run Tests (TESTS FAIL)*
3. *Write Code to Make Tests Pass*
4. *Run Tests (TESTS PASS)*
5. *Refactor (repeat until you can't think of anything else to test)*

Test-During Development



1. *Write Some Code*
2. *Write Some Tests*
3. *Run Tests (TESTS FAIL)*
4. *Write More Code*
5. *Run Tests (TESTS PASS)*
6. *Refactor (repeat until all the features and tests are done)*

Jest

Why Jest

- Zero configuration (One dependency - just install jest)
- Very fast

Airbnb switched from Mocha to Jest, and their total test runtime dropped from more than 12 minutes to only 4.5 minutes on a heavy-duty CI machine with 32 cores. Local tests used to take 45 minutes, which dropped to 14.5 minutes.

- Built-in assertions (No need install Chai.js, should.js or others)
- Built-in code coverage tool
- File names should be under **__tests__** or ***.spec.js** or ***.test.js**

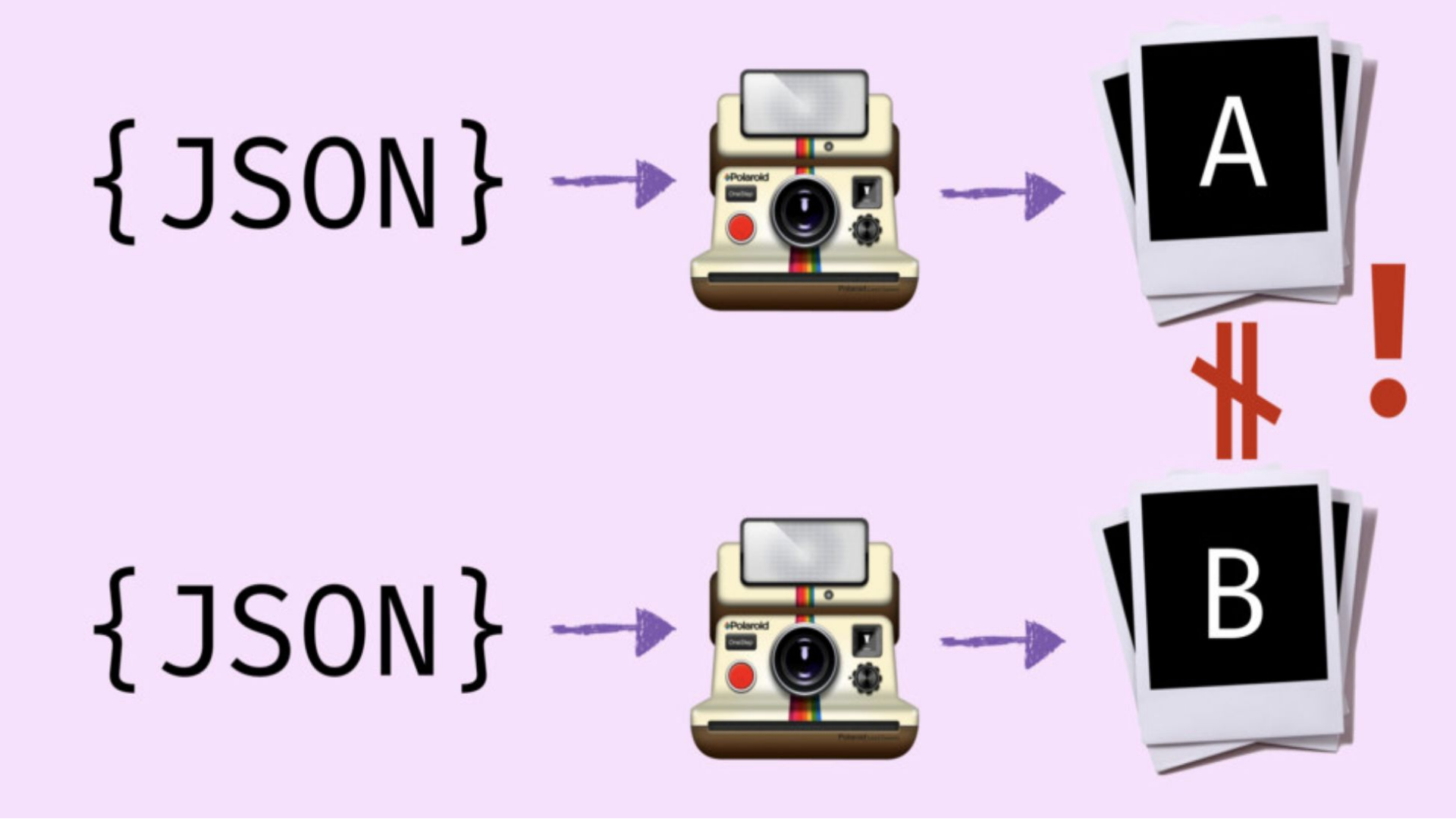
```
yarn add --dev jest
```

or

```
npm install --save-dev jest
```

Snapshot Testing

- It can be memorized how your React components are rendered
- Raising an error if there is a mismatch.
- The first time you run the test, Jest saves the snapshot to the __snapshots__ folder.


{JSON}
A
≠
!
{JSON}
B

Shallow Render Testing

- Renders only component itself without its children.
- Enzyme (jQuery like to find elements read props vs.)