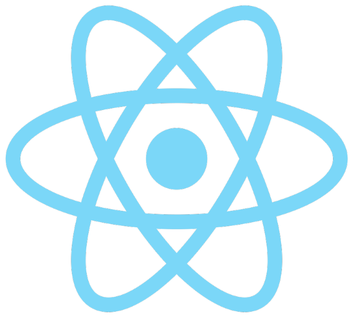
Unit Testing in Javascript


RAILS



AGENDA

- Intro to testing
- Unit testing
- Javascript tools
- React stack examples
- Coverage

WHY?

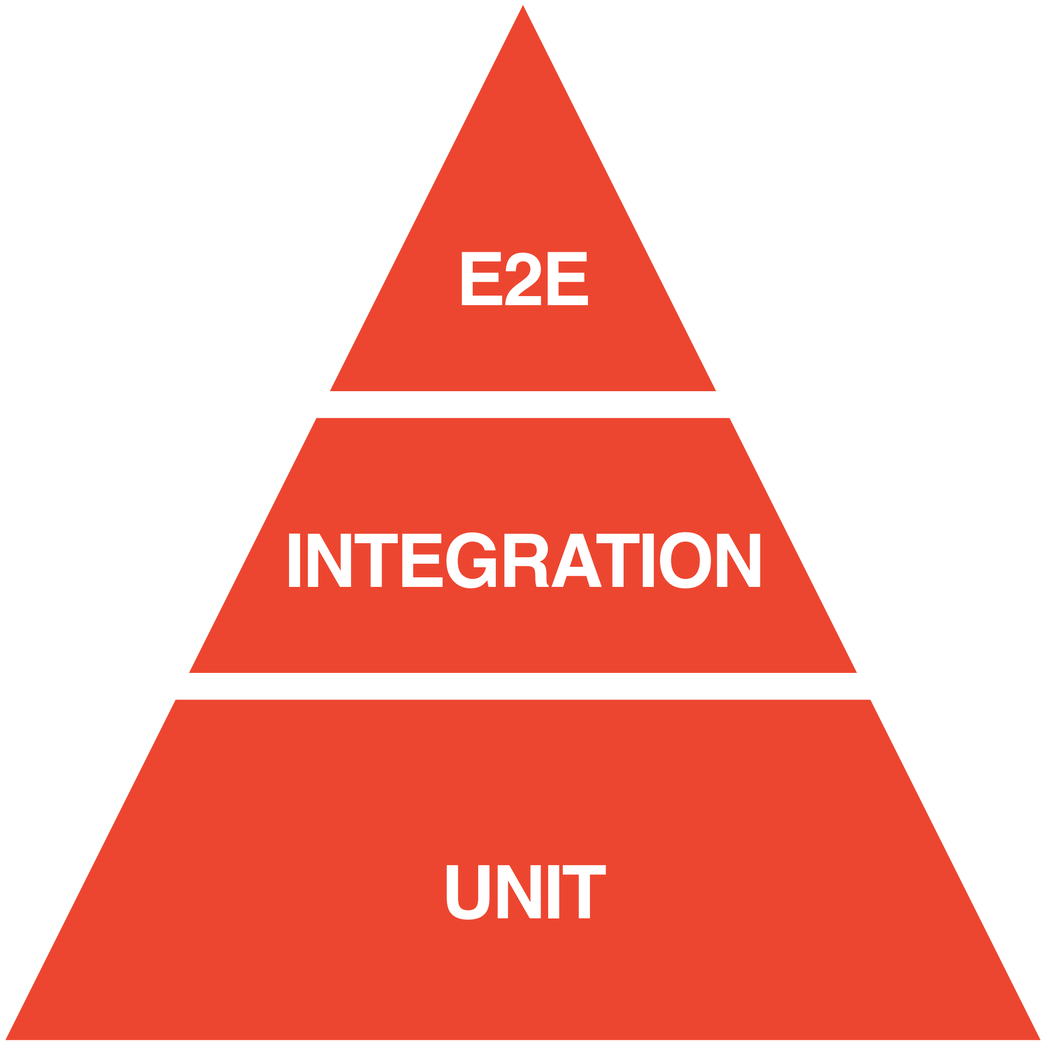
WRITE

REFACTOR

DOCUMENT

CHANGE

UNIT TESTING


E2E
INTEGRATION
UNIT

WHAT TO TEST

- Behavior
- Context
- Logic

WHAT NOT TO TEST

- Libraries
- Other components
- Implementation details

TOOLS

JEST

- Replace Chai, Mocha, & Sinon
- Coverage with Istanbul
- Parallel testing
- Snapshots

ENZYME

- Simple syntax
- Shallow rendering
- Simulate events

COMPONENTS

```
function Header({
  currentUser,
  shouldShowFinePrint = false,
}) {
  return (
    <div>
      Hello, {currentUser}
      {shouldShowFinePrint && 'Restrictions apply'}
    </div>
  );
}
```

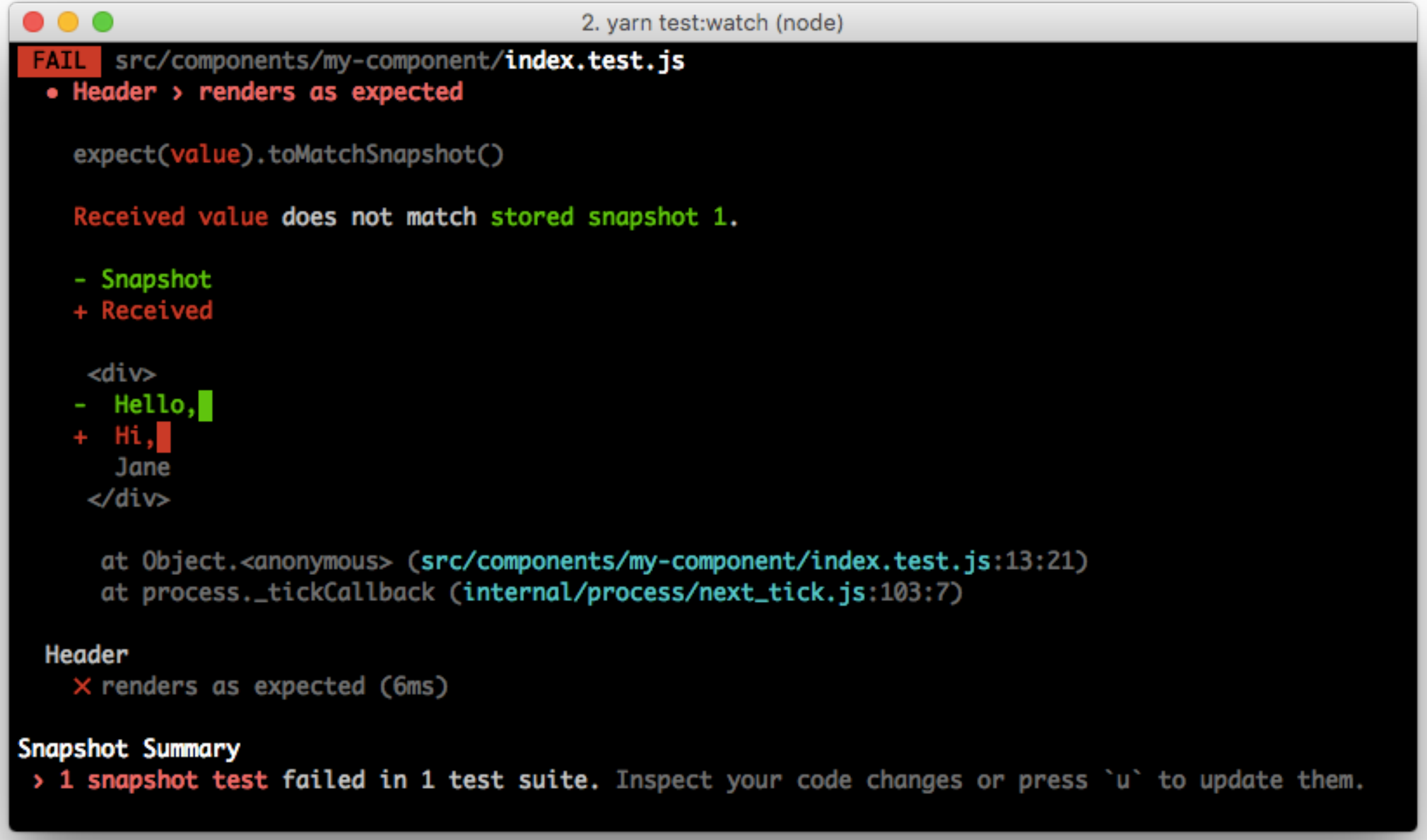
```
describe('Header', () => {
  const defaultProps = {
    currentUser: 'Jane',
  };

  it('renders as expected', () => {
    const wrapper = shallow(<Header { ...defaultProps } />);
    expect(wrapper).toHaveLength(1);
    expect(wrapper.text()).toMatch(`Hello, ${defaultProps.currentUser}`);
  });

  it('renders as expected with fine print turned on', () => {
    const wrapper = shallow(
      <Header { ...defaultProps } shouldShowFinePrint />
    );
    expect(wrapper.text()).toMatch('Restrictions apply');
  });
});
```

```
describe('Header', () => {
  const defaultProps = {
    currentUser: 'Jane',
  };

  it('renders as expected', () => {
    const wrapper = shallow(<Header { ...defaultProps } />);
    expect(wrapper).toHaveLength(1);
    expect(wrapper.text()).toMatch(`Hello, ${defaultProps.currentUser}`);
    expect(wrapper.text()).not.toMatch('Restrictions apply');
  });

  it('renders as expected with fine print turned on', () => {
    const wrapper = shallow(
      <Header { ...defaultProps } shouldShowFinePrint />
    );
    expect(wrapper).toHaveLength(1);
    expect(wrapper.text()).toMatch(`Hello, ${defaultProps.currentUser}`);
    expect(wrapper.text()).toMatch('Restrictions apply');
  });
});
```

```
describe('Header', () => {
  const defaultProps = {
    currentUser: 'Jane',
  };

  it('renders as expected', () => {
    const wrapper = shallow(<Header { ...defaultProps } />);
    expect(wrapper).toMatchSnapshot();
  });

  it('renders as expected with fine print turned on', () => {
    const wrapper = shallow(<Header { ...defaultProps }
shouldShowFinePrint />);
    expect(wrapper).toMatchSnapshot();
  });
});
```

```
exports[`Header renders as expected 1`] = `
<div>
  Hello,
  Jane
</div>
`;

exports[`Header renders as expected with fine print
turned on 1`] = `
<div>
  Hello,
  Jane
  Restrictions apply
</div>
`;
```

enzyme-to-json


2. yarn test:watch (node)
FAIL src/components/my-component/index.test.js
• Header › renders as expected
expect(value).toMatchSnapshot()
Received value does not match stored snapshot 1.
- Snapshot
+ Received

- Hello,
+ Hi,
Jane

at Object.<anonymous> (src/components/my-component/index.test.js:13:21)
at process._tickCallback (internal/process/next_tick.js:103:7)
Header
✕ renders as expected (6ms)
Snapshot Summary
› 1 snapshot test failed in 1 test suite. Inspect your code changes or press `u` to update them.

SNAPSHOT UTILITY

```
describe('Header', () => {
  const defaultProps = {
    currentUser: 'Jane',
  };

  it('renders as expected', () => {
    const wrapper = shallow(<Header { ...defaultProps } />);
    expect(wrapper).toMatchSnapshot();
  });

  it('renders as expected with fine print turned on', () => {
    const wrapper = shallow(<Header { ...defaultProps }
shouldShowFinePrint />);
    expect(wrapper).toMatchSnapshot();
  });
});
```

```
describe('Header', () => {
  const defaultProps = {
    currentUser: 'Jane',
  };

  itMatchesSnapshots(Header, defaultProps, {
    'with fine print': { shouldShowFinePrint: true },
  });
});
```

REWIRE

```
function Header({
  currentUser,
  shouldShowFinePrint = false,
}) {
  return (
    <div>
      Hi, {currentUser}
      {shouldShowFinePrint && 'Restrictions apply'}
    </div>
  );
}

export default connect(HeaderSelector)(Header);
```

```
export function Header({
  currentUser,
  shouldShowFinePrint = false,
}) {
  return (
    <div>
      Hi, {currentUser}
      {shouldShowFinePrint && 'Restrictions apply'}
    </div>
  );
}

export default connect(HeaderSelector)(Header);
```

```
import module from './index';

const Header = module.__get__('Header');
```

babel-plugin-rewire

SELECTORS

```
export const selectGridType = createSelector(
  selectIsSearchPage,
  selectProductGridType,
  (isSearchPage, gridType) =>
    isSearchPage ? 'SEARCH' : (gridType || 'BASIC')
);
```

```
it('sets pages with search terms to SEARCH', () => {
  const store = {
    routing: {
      locationBeforeTransitions: {
        pathname: '/products',
        query: { q: 'red' },
      },
    },
    cms: {
      '/products': {
        'content-uuid-1231': {
          'content-modules': [
            { template: 'super', type: 'product-grid' },
            { template: 'deluxe', type: 'some-other-type'},
          ],
        },
      },
    },
  };

  expect(selectGridType(store)).toEqual('SEARCH');
});
```

```
export const selectGridType = createSelector(
  selectIsSearchPage,
  selectProductGridType,
  (isSearchPage, gridType) =>
    isSearchPage ? 'SEARCH' : (gridType || 'BASIC')
);
```

```
export const selectGridType = createSelector(
  selectIsSearchPage,
  selectProductGridType,
  (isSearchPage, gridType) =>
    isSearchPage ? 'SEARCH' : (gridType || 'BASIC')
);
```

resultFunc

```
it('sets pages with search terms to SEARCH', () => {
  const result = selectGridType.resultFunc(true, 'SUPER');

  expect(result).toEqual('SEARCH');
});

it('sets pages without search terms to given grid type', () => {
  const result = selectGridType.resultFunc(false, 'SUPER');

  expect(result).toEqual('SUPER');
});

it('defaults to basic grid', () => {
  const result = selectGridType.resultFunc(false, null);

  expect(result).toEqual('BASIC');
});
```

ACTION CREATORS & REDUCERS

```
export addTodo = todo =>
  ({ payload: todo, type: 'ADD_TODO' });
```

```
export const todosReducer = (state = [], action) => {
  switch (action.type) {
  case ADD_TODO:
    return [ ...todos, action.payload ];
  default:
    return state;
  }
}
```

```
it('creates add todo action', () => {
  expect(addTodo('Wash dishes')).toEqual({
    type: 'ADD_TODO',
    payload: 'Wash dishes',
  });
});
```

```
it('adds todo', () => {
  const result = todosReducer([], {
    type: 'ADD_TODO',
    payload: 'Wash dishes',
  });

  expect(result).toEqual(['Wash dishes']);
});
```

```
it('creates add todo action', () => {
  expect(addTodo('Wash dishes')).toEqual({
    type: 'ADD_TODO',
    payload: 'Wash dishes',
  });
});

it('adds todo', () => {
  const result = todosReducer([], {
    type: 'ADD_TODO',
    payload: 'Wash dishes',
  });

  expect(result).toEqual(['Wash dishes']);
});
```

```
import { todos, addTodo }

it('adds todo', () => {
  const result = todosReducer(
    [],
    addTodo('Wash dishes’)
  );

  expect(result).toEqual(['Wash dishes']);
});
```

SAGAS

```
export function* watchLoginSuccess() {
  const user = yield select(selectCurrentUser);

  const resp = yield call(fetchUserPosts, user.id);

  if (!resp.errorCode) {
    yield put(actions.getUserPosts(resp))
  }
}
```

```
let next;
const mockUser = { id: '123' };
const mockResponse = { posts: [] };

it('selects current user', () => {
  next = generator.next();
  expect(next.value).toEqual(select(selectCurrentUser));
});

it('calls fetchUserPosts API', () => {
  next = generator.next(mockUser);
  expect(next.value).toEqual(call(fetchUserPosts, mockUser.id));
});

it('dispatches getUserPosts action', () => {
  next = generator.next(mockResponse);
  expect(next.value).toEqual(put(actions.getUserPosts(mockResponse));
});

it('is done', () => {
  next = generator.next();
  expect(next.done).toEqual(true);
})
```

SAGA UTILITY

```
const mockUser = { id: '123' };
const mockResponse = { posts: [] };
const mockError = { errorCode: '456' };

itMatchesFlow('default behavior', generator, [
  { effect: select(selectCurrentUser), returns: mockUser },
  {
    effect: call(fetchUserPosts, mockUser.id),
    returns: mockResponse,
  },
  { effect: put(actions.getUserPosts(mockUser.id)) },
]);

itMatchesFlow('error', generator, [
  { effect: select(selectCurrentUser), returns: mockUser },
  {
    effect: call(fetchUserPosts, mockUser.id),
    returns: mockError,
  }
]);
```

COVERAGE

All files

36.4% Statements `4371/12007` **12.75%** Branches `887/6959` **17.68%** Functions `646/3654` **37.44%** Lines `4275/11419`

File		Statements		Branches		Functions		Lines	
src/selectors/product-detail-page/select-submit-review-modal-props		66.67%	2/3	100%	0/0	0%	0/1	66.67%	2/3
src/test/mocks		100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/api/my-account		47.06%	8/17	100%	0/0	0%	0/6	57.14%	8/14
src/selectors/my-account-profile/personal-information		100%	2/2	100%	0/0	100%	0/0	100%	2/2
src/components/btn/skip		50%	1/2	100%	0/0	0%	0/1	50%	1/2
src/selectors/my-account-profile/password		100%	2/2	100%	0/0	100%	0/0	100%	2/2
src/components/form/field		100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/selectors/my-account-profile		100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/components/highlight/tile		50%	2/4	100%	0/0	0%	0/1	50%	2/4
src/containers/common		100%	0/0	100%	0/0	100%	0/0	100%	0/0
src/constants		100%	132/132	100%	0/0	100%	0/0	100%	132/132
src/config/forms		100%	71/71	100%	0/0	100%	0/0	100%	71/71
src/components/payment-method/card		50%	1/2	100%	0/0	0%	0/1	50%	1/2
src/components/highlight/list		50%	1/2	100%	0/0	0%	0/1	50%	1/2
src/containers/my-account-page		45.45%	5/11	100%	0/0	0%	0/3	45.45%	5/11
src/components		100%	18/18	100%	0/0	100%	0/0	100%	18/18

techcareers@work.co
www.work.co