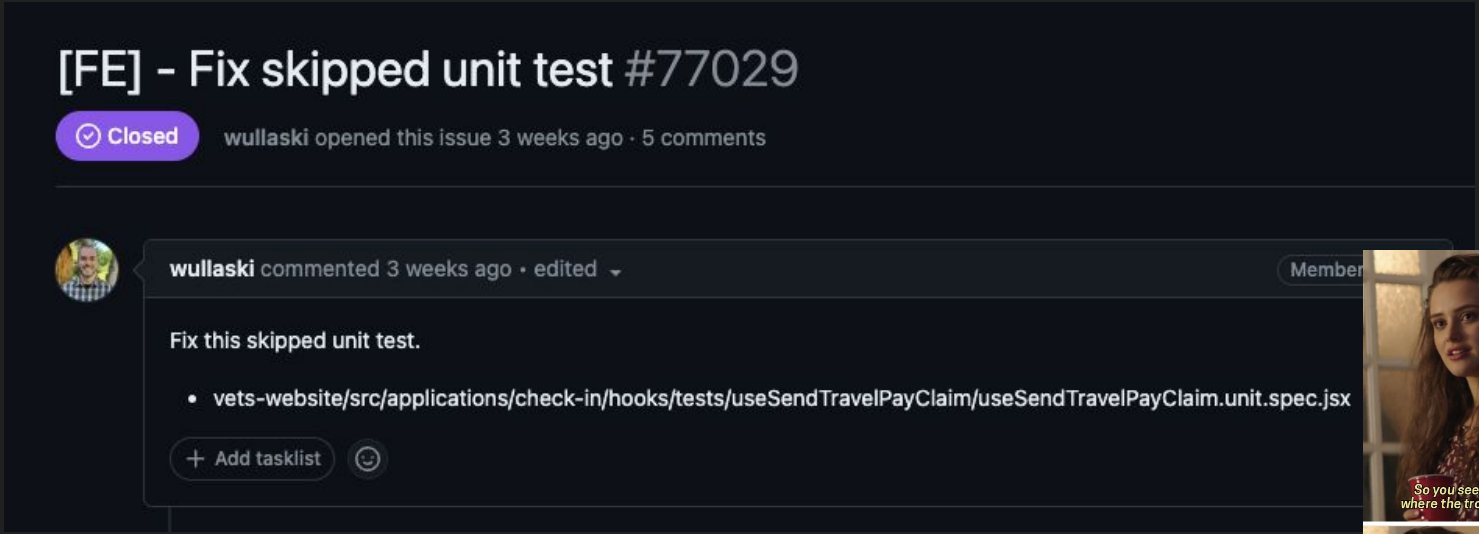
Fixing a Flaky Test



[FE] - Fix skipped unit test #77029

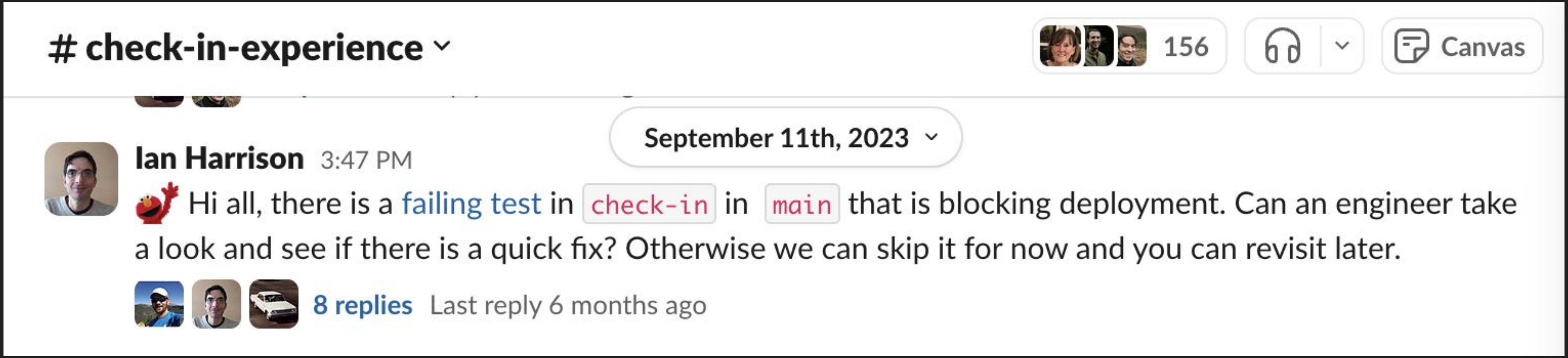
Closed · wullaski opened this issue 3 weeks ago · 5 comments

wullaski commented 3 weeks ago · edited · Member

Fix this skipped unit test.

- vets-website/src/applications/check-in/hooks/tests/useSendTravelPayClaim/useSendTravelPayClaim.unit.spec.jsx

Why It Was Skipped



Ian Harrison 3:47 PM

Hi all, there is a failing test in `check-in` in `main` that is blocking deployment. Can an engineer take a look and see if there is a quick fix? Otherwise we can skip it for now and you can revisit later.

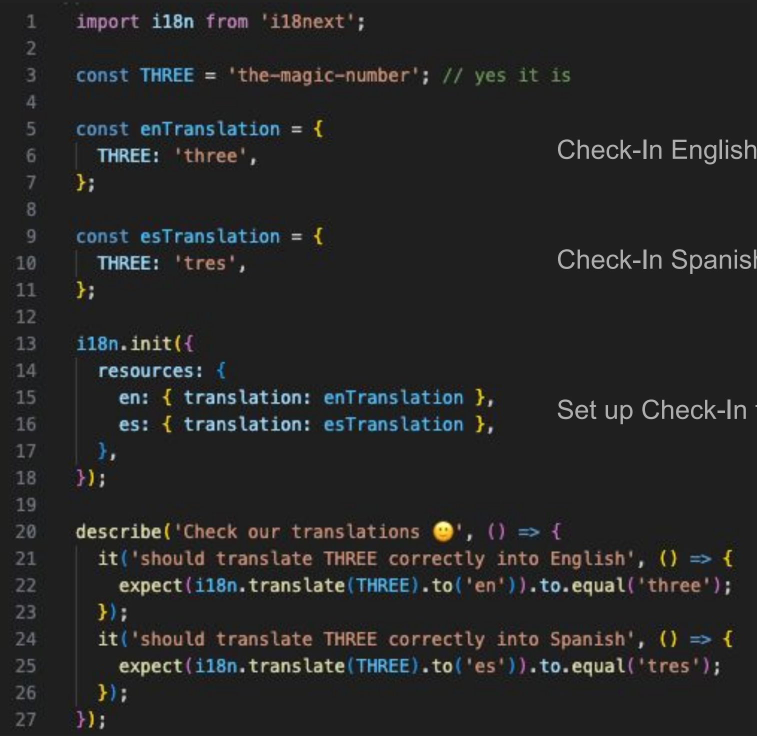
8 replies Last reply 6 months ago

TL;DR: We have ~2560 unit test files over ~70 apps, and any one of them can block deployment.

Why It Failed

- Multiple teams translate user-facing text using `i18next`, directly or indirectly.
- `i18next` is initialized (including its references to translation resources) in a global context.
- In a unit test context, tests spanning multiple teams will share this global context.
- Therefore, changes made by one team can impact the tests of another team.
- This is sensitive to how tests are ordered, which may change from test run to test run.
- Consequently, this can cause flakiness and confusing problems that block deployments.

Illustration (Two Test Files)



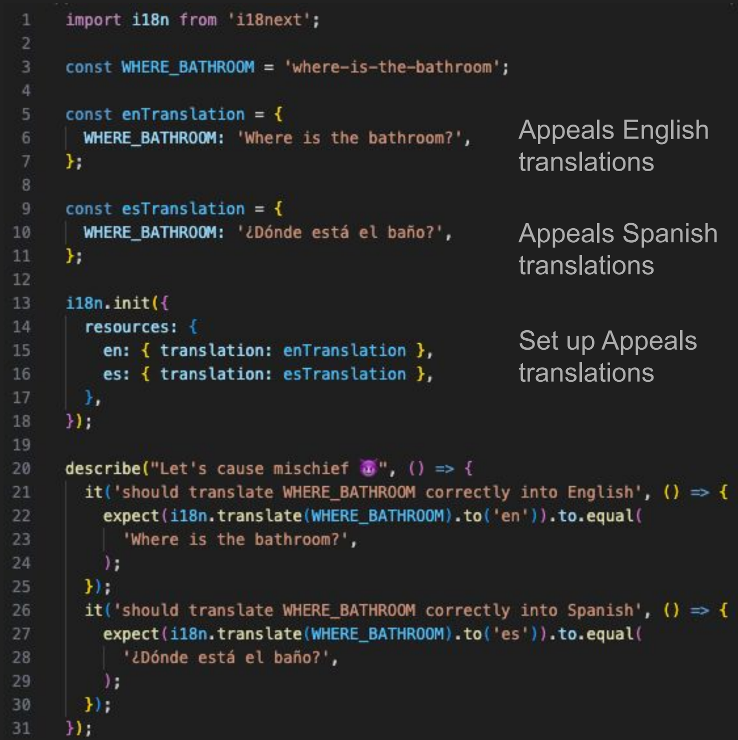
```javascript
import i18n from 'i18next';

const THREE = 'the-magic-number'; // yes it is

const enTranslation = {
  THREE: 'three',
};

const esTranslation = {
  THREE: 'tres',
};

i18n.init({
  resources: {
    en: { translation: enTranslation },
    es: { translation: esTranslation },
  },
});

describe('Check our translations 🙂', () => {
  it('should translate THREE correctly into English', () => {
    expect(i18n.translate(THREE).to('en')).to.equal('three');
  });
  it('should translate THREE correctly into Spanish', () => {
    expect(i18n.translate(THREE).to('es')).to.equal('tres');
  });
});
```

Check-In English translations

Check-In Spanish translations

Set up Check-In translations

Test Check-In translations



```javascript
import i18n from 'i18next';

const WHERE_BATHROOM = 'where-is-the-bathroom';

const enTranslation = {
  WHERE_BATHROOM: 'Where is the bathroom?',
};

const esTranslation = {
  WHERE_BATHROOM: '¿Dónde está el baño?',
};

i18n.init({
  resources: {
    en: { translation: enTranslation },
    es: { translation: esTranslation },
  },
});

describe("Let's cause mischief 😈", () => {
  it('should translate WHERE_BATHROOM correctly into English', () => {
    expect(i18n.translate(WHERE_BATHROOM).to('en')).to.equal(
      'Where is the bathroom?',
    );
  });
  it('should translate WHERE_BATHROOM correctly into Spanish', () => {
    expect(i18n.translate(WHERE_BATHROOM).to('es')).to.equal(
      '¿Dónde está el baño?',
    );
  });
});
```

Appeals English translations

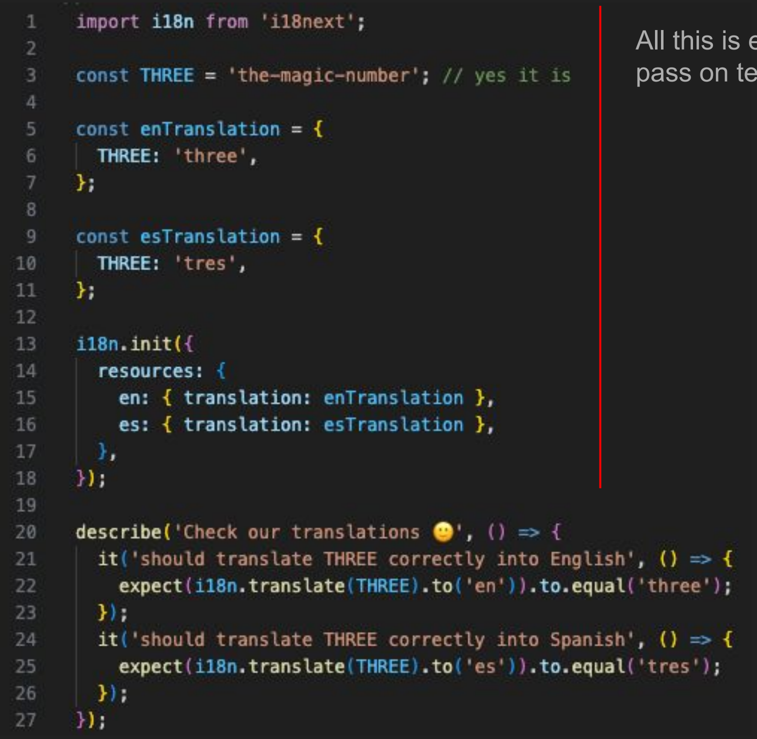
Appeals Spanish translations

Set up Appeals translations

How Tests Are Executed

- `yarn test:unit test1.unit.spec.js test2.unit.spec.js`
  - Execute `test1` file then `test2` file
  - Simple
  - Straightforward
  - Not actually true
- Actually run through things in this order:
  - `test1` global execution context
  - `test2` global execution context
  - `test1` tests
  - `test2` tests

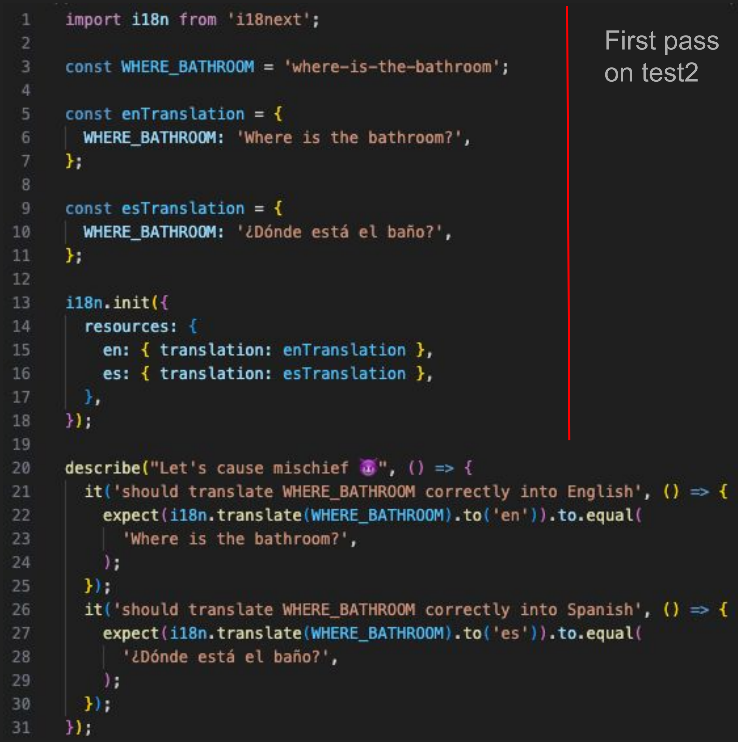
Illustration (Two Test Files)



```javascript
import i18n from 'i18next';

const THREE = 'the-magic-number'; // yes it is

const enTranslation = {
  THREE: 'three',
};

const esTranslation = {
  THREE: 'tres',
};

i18n.init({
  resources: {
    en: { translation: enTranslation },
    es: { translation: esTranslation },
  },
});

describe('Check our translations 🙂', () => {
  it('should translate THREE correctly into English', () => {
    expect(i18n.translate(THREE).to('en')).to.equal('three');
  });
  it('should translate THREE correctly into Spanish', () => {
    expect(i18n.translate(THREE).to('es')).to.equal('tres');
  });
});
```

All this is executed in the first pass on test1

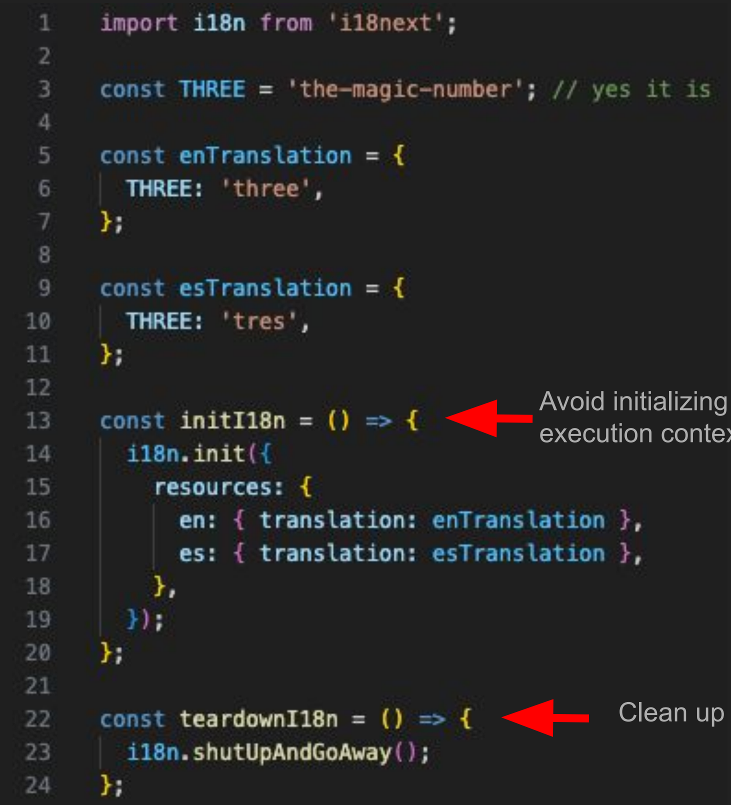
Second pass on test1



```javascript
import i18n from 'i18next';

const WHERE_BATHROOM = 'where-is-the-bathroom';

const enTranslation = {
  WHERE_BATHROOM: 'Where is the bathroom?',
};

const esTranslation = {
  WHERE_BATHROOM: '¿Dónde está el baño?',
};

i18n.init({
  resources: {
    en: { translation: enTranslation },
    es: { translation: esTranslation },
  },
});

describe("Let's cause mischief 😈", () => {
  it('should translate WHERE_BATHROOM correctly into English', () => {
    expect(i18n.translate(WHERE_BATHROOM).to('en')).to.equal(
      'Where is the bathroom?',
    );
  });
  it('should translate WHERE_BATHROOM correctly into Spanish', () => {
    expect(i18n.translate(WHERE_BATHROOM).to('es')).to.equal(
      '¿Dónde está el baño?',
    );
  });
});
```

First pass on test2

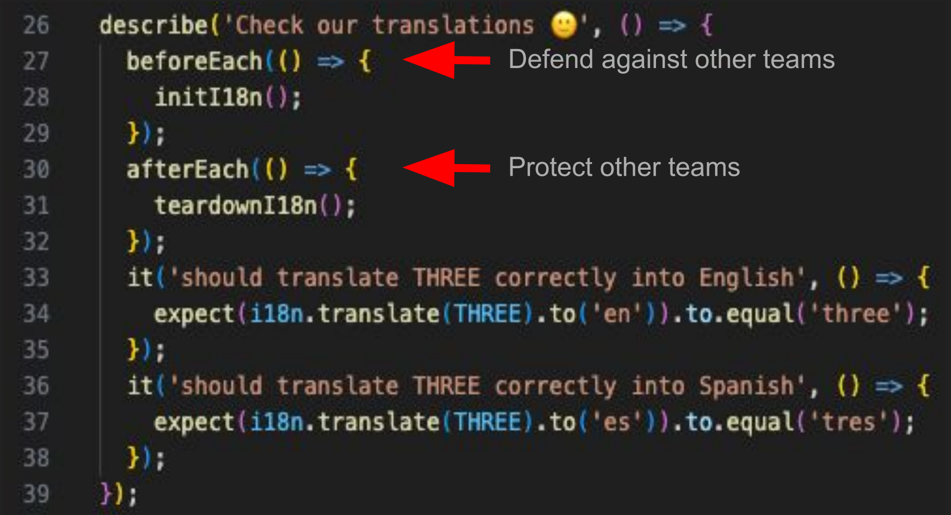
Solution (Nasty Hacks)



```
import i18n from 'i18next';

const THREE = 'the-magic-number'; // yes it is

const enTranslation = {
  THREE: 'three',
};

const esTranslation = {
  THREE: 'tres',
};

const initI18n = () => {
  i18n.init({
    resources: {
      en: { translation: enTranslation },
      es: { translation: esTranslation },
    },
  });
};

const teardownI18n = () => {
  i18n.shutUpAndGoAway();
};
```

Avoid initializing in global execution context

Clean up after ourselves



```
describe('Check our translations 🙂', () => {
  beforeEach(() => {
    initI18n();
  });
  afterEach(() => {
    teardownI18n();
  });
  it('should translate THREE correctly into English', () => {
    expect(i18n.translate(THREE).to('en')).to.equal('three');
  });
  it('should translate THREE correctly into Spanish', () => {
    expect(i18n.translate(THREE).to('es')).to.equal('tres');
  });
});
```

Defend against other teams

Protect other teams

Conclusions

- Structural Problems:
  - Don’t put all your eggs in one basket
  - Don’t pollute global state
  - Don’t put teams in the position of compromising their own products’ integrity to unblock other teams’ deployments