Integration Tests

In this lesson, we'll learn what integration tests are and see one in practice.

What are integration tests?

We've covered what unit tests look like, but what about integration tests?

They're extremely similar. You write integration tests to test that functions work well together. It's possible to have functions that pass comprehensive sets of unit tests but be part of failing integration tests. They usually involve multiple systems interacting together.

Example

Consider our form and how it might be sent to a server. We can have a function called `serialize`, which gives a representation that can be stored as a string and a `deserialize` function, which reads data in that form and populates our field with that data.

```
serialize() {
  // Get form fields
  ...
  return `form_${firstName}_${lastName}...`;
}

deserialize(serializedForm) {
  // Extract form fields and populate form
  ...
}
```

For example, you save a form as `form_walt_disney`, and when they come to a page with a form, we can help them fill out the first and last name with “Walt” and “Disney.”

We can run all sorts of tests to make sure that all these methods work. Also, let’s say we have a function that sends HTTP requests:

```
function httpRequest({url, data}) {
  return this.request(url + `?data=data`);
}
```

And the many tests for `httpRequest` all pass too. However, URLs have a character limit. So if someone exclusively wrote unit tests and shipped the product, one day someone will come along and enter a first name that is longer than the character limit, serialize it, and when they try to retrieve it again, they’ll fail.

You need an integration test to catch that:

```
function TestBugIntegration() {
  let longName = '';
  for (let i = 0; i < 3000; i++) {
    longName += 'blah';
  }

  const firstNameInput = document.getElementsByClassName('form__input--firstName')[0];
  firstNameInput.value = longName;

  httpRequest({
    url: '/saveForm',
    data: serialize()
  });

  const serializedForm = httpRequest({
    url: '/getForm',
  });
  const form = deserialize(serializedForm);

  // This will fail!
  assert(form.firstName === longName);
}
```

This test creates a really long name, sets the input value as such, serializes it, sends the request to an endpoint that saves it, and retrieves it from another endpoint. At no point do we question the integrity of `serialize`, `deserialize`, or `httpRequest`, since we assume there are ample unit tests for them. It’s the interaction of the functions which surface the bug.