Testing with Jasmine

Goals

- Explain what testing is
- Understand why we write tests
- Write tests with Jasmine

Writing Tests

Why Test?

- Manually testing software is boring
  - So we tend to not re-run things that “work”
  - And therefore don’t notice when they break
- Tests often clarify expectations of a function
  - What should legal input/output be?
- Tests are often a great way to understand what code does
- It’s a core skill for professional devs

Jasmine

- *Jasmine* is a popular JavaScript testing framework
- We’ll use Jasmine for testing until we get to Python
- We’ll see another framework, *Jest*, with Node and React

Running Tests in the Browser with Jasmine

```
<!doctype html>
<html>
<head>
<title>Taxes Tests</title>

<!-- include CSS for Jasmine -->
<link rel="stylesheet"
  href="https://unpkg.com/jasmine-core@4.1.0/lib/jasmine-core/jasmine.css" />
</head>
<body>

<!-- include JS for Jasmine -->
<script
  src="https://unpkg.com/jasmine-core@4.1.0/lib/jasmine-core/jasmine.js"></script>
<script
  src="https://unpkg.com/jasmine-core@4.1.0/lib/jasmine-core/jasmine-html.js"></script>
<script
  src="https://unpkg.com/jasmine-core@4.1.0/lib/jasmine-core/boot0.js"></script>
<script
  src="https://unpkg.com/jasmine-core@4.1.0/lib/jasmine-core/boot1.js"></script>

<!-- include your JS & test file -->
```

```
<script src="taxes.js"></script>
<script src="taxes.test.js"></script>
</body>
</html>
```

Then open this HTML page in browser

Writing tests with Jasmine

Write your functions that will be tested:

demo/taxes.js

```
function calculateTaxes(income) {
  if (income > 30000) {
    return income * 0.25;
  } else {
    return income * 0.15;
  }
}

console.log(calculateTaxes(500))
```

Write a test file:

demo/taxes.test.js

```
it('should calculate lower-bracket taxes', function () {
  expect(calculateTaxes(10000)).toEqual(1500);
  expect(calculateTaxes(20000)).toEqual(3000);
});

it('should calculate higher-bracket taxes', function () {
  expect(calculateTaxes(50000)).toEqual(12500);
  expect(calculateTaxes(80000)).toEqual(20000);
});
```

Let’s break that down

demo/taxes.test.js

```
it('should calculate lower-bracket taxes', function () {
  expect(calculateTaxes(10000)).toEqual(1500);
  expect(calculateTaxes(20000)).toEqual(3000);
});

it('should calculate higher-bracket taxes', function () {
  expect(calculateTaxes(50000)).toEqual(12500);
  expect(calculateTaxes(80000)).toEqual(20000);
});
```

- *Test cases* are functions passed to `it(...)`
- First argument become test case name (shown by Jasmine)

demo/taxes.test.js

```
it('should calculate lower-bracket taxes', function () {
  expect(calculateTaxes(10000)).toEqual(1500);
  expect(calculateTaxes(20000)).toEqual(3000);
});

it('should calculate higher-bracket taxes', function () {
  expect(calculateTaxes(50000)).toEqual(12500);
  expect(calculateTaxes(80000)).toEqual(20000);
});
```

- Test cases can contain any normal code plus “expectations”
- Format is `expect(someValue).someMatcher(...)`

Matchers

`.toEqual(obj)`
: Has the same value (eg, different lists with same values match)

`.toBe(obj)`
: Is the same object (eg, different lists with same items don’t)

`.toContain(obj)`
: Does object/array contain this item?

`.not.`
: Add before matcher to invert (eg `expect(...).not.toEqual(7)`)

https://jasmine.github.io/api/edge/matchers.html <https://jasmine.github.io/api/edge/matchers.html>

What To Test

- Test every function in at least one way
- Think about “edges”
  - What if the list were empty?
  - What about non-integer numbers?
  - What if the file can’t be found?
  - Is the first case/last case handled differently?

Testable Code

Write code that is easier to test!

- More functions & smaller functions: easier to test (& debug!)
- Don’t mix logic & UI in a function

```
function playTicTacToe() {
  // ...
  let winner = checkForWinner();
}

function checkForWinner() {
  // code for checking board here...
  if (winner) {
    alert(winner + " wins!");
  }
  return winner;
}
```

```
function playTicTacToe() {
  // ...
  let winner = checkForWinner();
  if (winner) {
    announceWinner(winner);
  }
}

function checkForWinner() {
  // code for checking board here...
  return winner;
}

function announceWinner(winner) {
  alert(winner + " wins!");
}
```