

Test-Driven Development



Test-Driven Development

- ◉ TDD is a popular approach to writing code among professional engineers
 - Write tests that run your code and compare the behavior of your code against an expected outcome
 - Write code that passes the tests
 - Repeat

Test-Driven Development

- ◉ Some benefits:
 - Expectations of how code should behave clearly defined before writing code
 - Automated tests easily verify that your code is working as expected
 - Well-written tests help others understand how your code is supposed to work
- ◉ Some perceived drawbacks
 - Tests take time to write
 - Tests can be difficult to read (that's why we start practicing now)

Test-Driven Development

- ◉ TDD plays a central role at Fullstack
 - Great learning tool
 - Very important skillset to develop before entering job market
- ◉ TDD plays a central role in the Bootcamp
 - You will not write tests yourself initially (we've done that for you)
 - The workshops over the next few weeks will use TDD

TDD Setup: Codepen

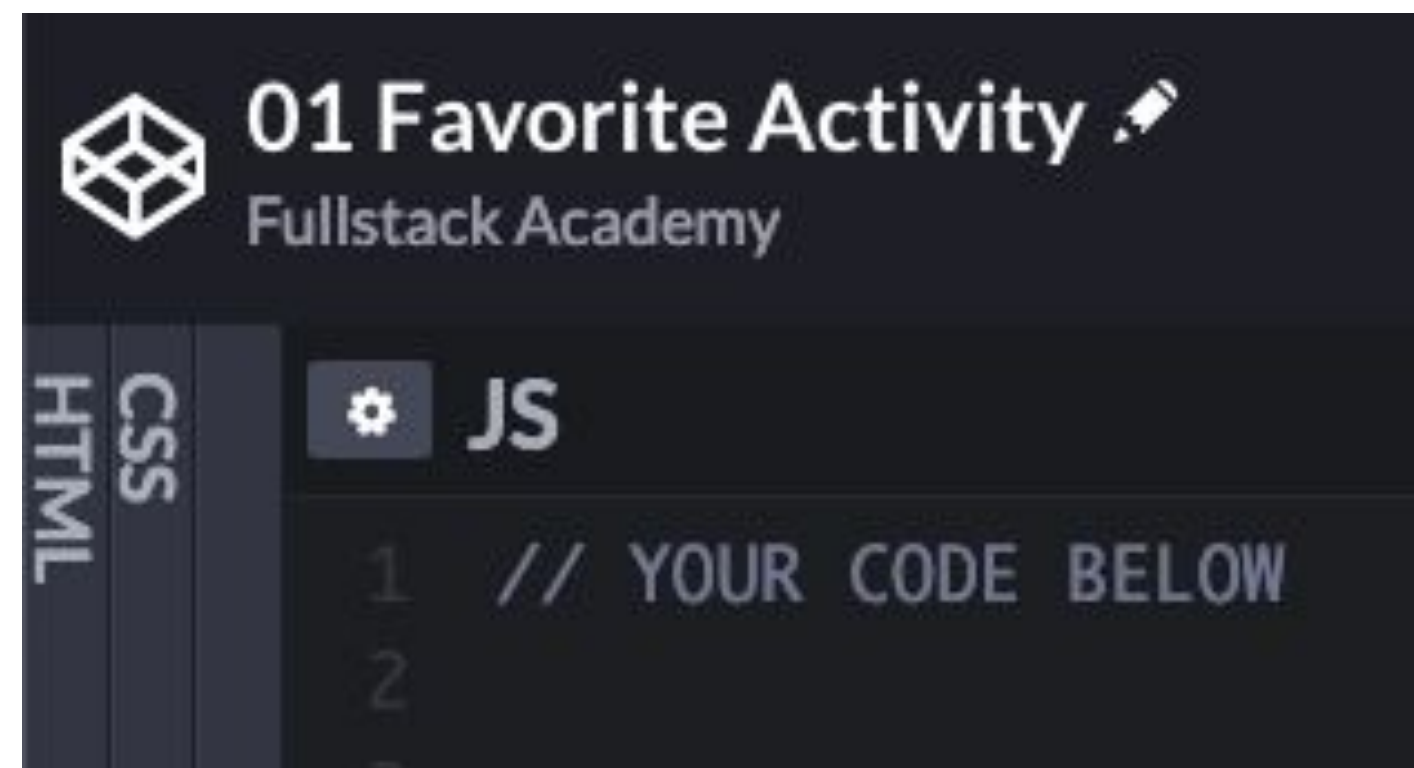
- ◉ Select the CodePen link for an individual problem from the “Practice Problems” section on LearnDot

Practice Problems

1) [Favorite Activity](#)

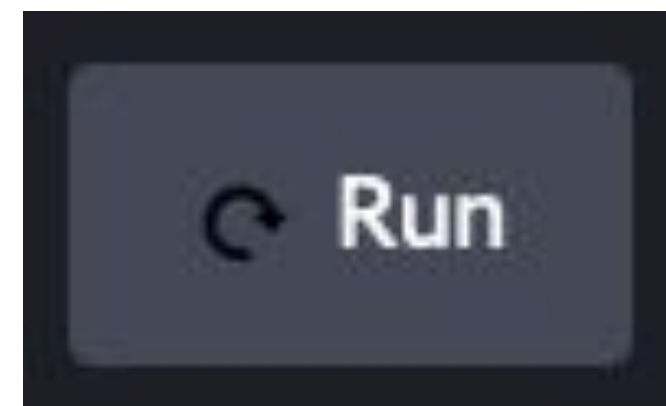
TDD Setup: Codepen

- ◉ Select the CodePen link for an individual problem from the “Practice Problems” section on LearnDot
- ◉ Write your code in the “JS” section of the CodePen, under the “YOU CODE BELOW” comment



TDD Setup: Codepen

- ◉ Select the CodePen link for an individual problem from the “Practice Problems” section on LearnDot
- ◉ Write your code in the “JS” section of the CodePen, under the “YOU CODE BELOW” comment
- ◉ Select the “Run” button to run your code and the tests



TDD Setup: Running Code Locally

- ◉ Download the practice problems for the course
- ◉ Open the downloaded folder in VSCode (VSC)
- ◉ In VSC, open the terminal (View > Debug Terminal > Terminal tab)
- ◉ Type testem
- ◉ Copy the url (similar to `http://localhost:7357/`) and paste it into your browser

TDD Setup: Running Code Locally

- ◉ The site you've navigated to is not from the internet; testem is using your own computer to create this site.
- ◉ This site is showing you the current status of the tests (or specs) being run against your code.

The screenshot displays the Jasmine test runner interface. At the top, there are tabs for 'Specs' and 'Readme'. Below the tabs, the Jasmine logo and version '3.0.0' are shown, along with an 'Options' button. A red banner indicates '4 specs, 4 failures' and 'finished in 0.008s'. Below this, a 'Spec List' section shows four failing specs, each with a red header and a detailed error message. The first spec is 'onlyOdds > is a function' with the error 'Expected 'undefined' to equal 'function''. The second, third, and fourth specs are 'onlyOdds > returns a number', 'onlyOdds > returns the sum of all odd nums between the provided argument and 0', and 'onlyOdds > returns 0 if inputted argument is less than 1' respectively, all with the error 'ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 11)'. Each spec entry includes a code editor snippet showing the test call and the Jasmine prompt.

```
Specs  Readme

Jasmine 3.0.0 Options
xxxx

4 specs, 4 failures finished in 0.008s

Spec List

onlyOdds > is a function
Expected 'undefined' to equal 'function'.
<Jasmine>
@pen.js:108:29
<Jasmine>

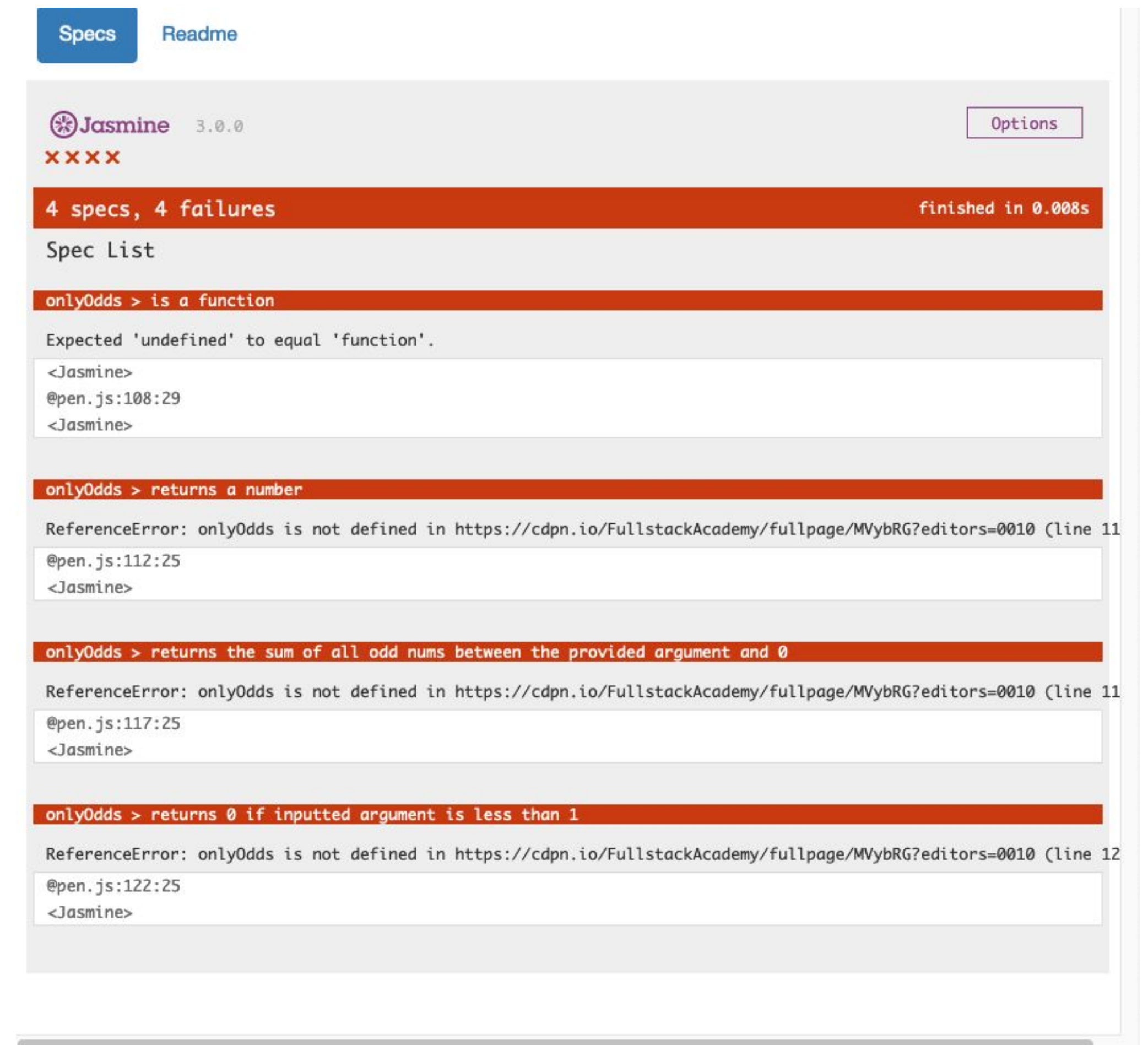
onlyOdds > returns a number
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 11)
@pen.js:112:25
<Jasmine>

onlyOdds > returns the sum of all odd nums between the provided argument and 0
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 11)
@pen.js:117:25
<Jasmine>

onlyOdds > returns 0 if inputted argument is less than 1
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 12)
@pen.js:122:25
<Jasmine>
```

TDD Setup: Running Code Locally

- ◉ All of the tests are initially failing because you haven't written any code yet!
- ◉ Every time you save one of the solution files, the tests will be automatically run again.
- ◉ When all the tests are passing, you're finished with the workshop!



The screenshot shows the Jasmine 3.0.0 test runner interface. At the top, there are tabs for 'Specs' (selected) and 'Readme'. Below the tabs, the Jasmine logo and version '3.0.0' are displayed, along with an 'Options' button. A summary bar indicates '4 specs, 4 failures' and 'finished in 0.008s'. The 'Spec List' section shows four failing specifications, each with a red header bar:

- onlyOdds > is a function**
Expected 'undefined' to equal 'function'.
@pen.js:108:29
- onlyOdds > returns a number**
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 11)
@pen.js:112:25
- onlyOdds > returns the sum of all odd nums between the provided argument and 0**
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 11)
@pen.js:117:25
- onlyOdds > returns 0 if inputted argument is less than 1**
ReferenceError: onlyOdds is not defined in https://cdpn.io/FullstackAcademy/fullpage/MVyBRG?editors=0010 (line 12)
@pen.js:122:25

Passing the first test

- ◉ Let's look at the prompt for the first practice problem:
 - “Create a variable called `favoriteActivity`. Assign the value ‘coding’ to `favoriteActivity`.”

- ◉ Now let's look at the output for the first test:

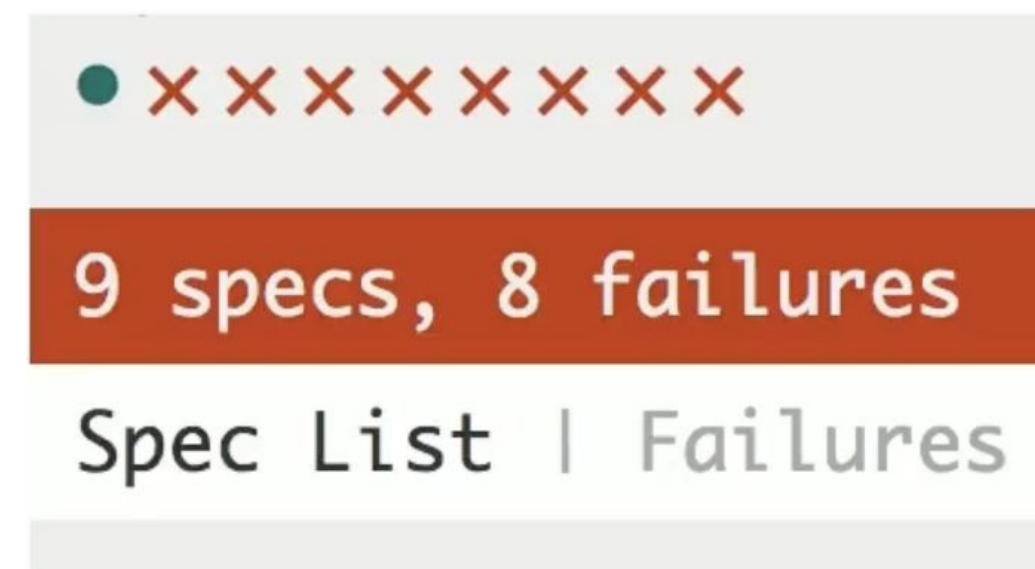
```
favoriteActivity should be coding
```

```
ReferenceError: favoriteActivity is not defined
```

- ◉ The test is looking through your code, trying to find a variable called `favoriteActivity`.

Passing the first test

- ◉ Open the Favorite Activity CodePen (or favorite-activity.js in the 01-favorite-activity folder) and create a `favoriteActivity` variable and set it equal to 'coding'.
- ◉ As soon as you run or save your code, the test site should update



- ◉ One of the red Xs turned into a green dot! You're passing a test!

Debugging a failing test

- ◉ Go back and set the variable `favoriteActivity` equal to the string `'not coding'`
- ◉ What does the output of the test say now?

```
favoriteActivity > should be coding
```

```
Expected 'not coding' to equal 'coding'.
```

- ◉ The test is telling you, “I found the `favoriteActivity` variable, but the value stored in it is `'not coding'`, and it needs to be `'coding'` to pass this test”
- ◉ Use this information to help you fix your code until all of the tests pass

Reading a test

- ◉ It's often useful to read the tests so you can understand what they are testing
- ◉ Go back to the CodePen (or open `favorite-activity-spec.js`).
- ◉ Look for the word `expect`:

```
expect(favoriteActivity).toEqual('coding');
```
- ◉ The test comes down to this line; the test will pass when the value stored in `favoriteActivity` equals the string `'coding'`
- ◉ Look for lines that start with `expect` in the other test specs to better understand exactly how they're working

Recap

/*

- Benefits of Test-Driven Development
- Using CodePen is fine!
- Reading tests and passing specs

*/