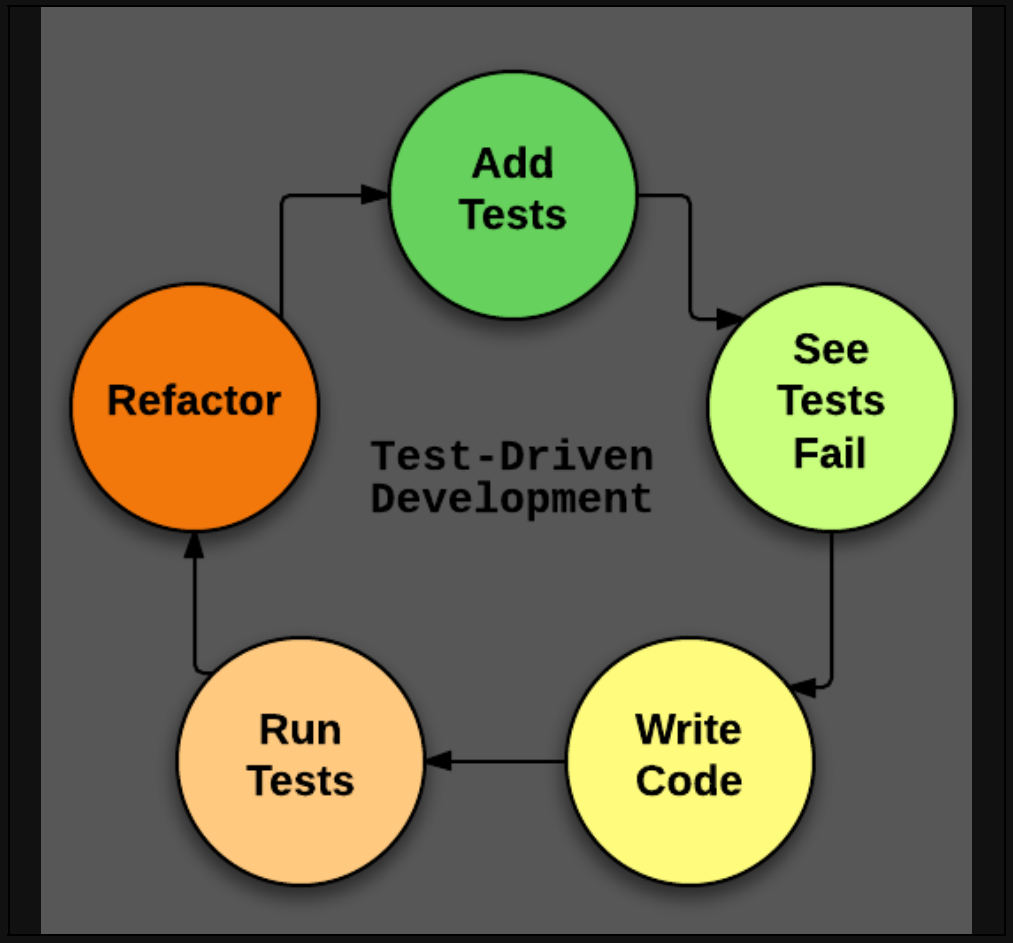
Jasmine

en spiral
DEV
ACADEMY

Test-Driven Development



https://joshldavis.com/2013/05/27/difference-between-tdd-and-bdd/

Let's Build This:

http://euglazer.github.io/ilovecats/

and let's build it well

HTML

```
<!DOCTYPE html>

<head>
  <link href='http://fonts.googleapis.com/css?family=Roboto:300&subset=cyrillic' rel='stylesheet' type='text/css'>
  <link rel="stylesheet" type="text/css" href="./stylesheets/index.css">
</head>

<body>
  <h1>я люблю</h1>
  <div id="column"> <--! all of our line divs go in here --> </div>
  <h1>мой кот</h1>
</body>
```

CSS

```
body { text-align: center; font-family: 'Roboto'; }

h1 { font-size: 50px; margin: 10px auto; }

div#column { margin: 0 auto; width: 200px; }

div.line { width: 100%; height: 1px; margin-bottom: 1px; }
```

JS

```
¯\_(ツ)_/¯
```

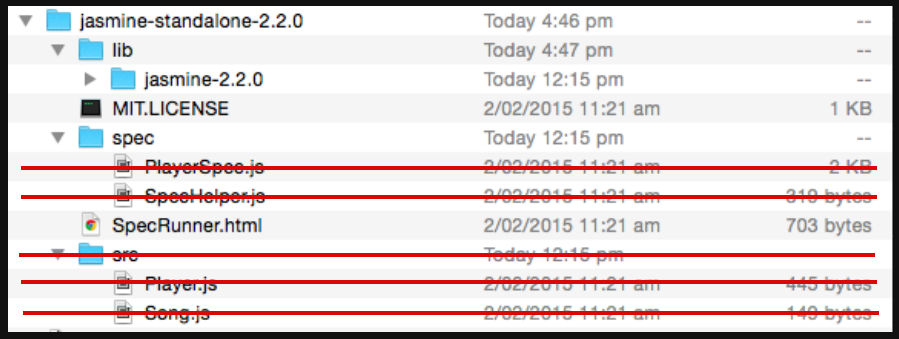
ok, Let's start building!

NOPE WRITE TESTS FIRST

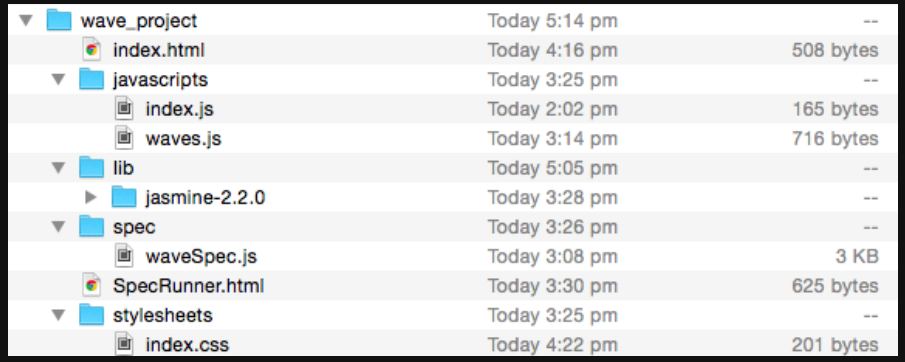
Download Jasmine

https://github.com/jasmine/jasmine/releases/download/v2.2.0/jasmine-standalone-2.2.0.zip

Delete Junk



Add Files/Folders to Project



We need:

- An object constructor function, *WaveColumn()* with:
- Properties:
  - *divArray*
- Methods:
  - *generateRedLine(opacity),* makes a red div with specified opacity and class 'line'
  - *initializeDivArray(),* adds 255 red lines to divArray, each incrementing in opacity
  - *rotateDivArray(),* removes first line in divArray and puts it in the back
  - *print(),* clears everything in div#column, and appends divArray to it.

Edit SpecRunner.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Jasmine Spec Runner v2.2.0</title>

  <link rel="shortcut icon" type="image/png" href="lib/jasmine-2.2.0/jasmine_favicon.png">
  <link rel="stylesheet" href="lib/jasmine-2.2.0/jasmine.css">

  <script src="lib/jasmine-2.2.0/jasmine.js"></script>
  <script src="lib/jasmine-2.2.0/jasmine-html.js"></script>
  <script src="lib/jasmine-2.2.0/boot.js"></script>

  <!-- include source files here... -->
  <script src="javascripts/waves.js"></script> // contains our JS waveColumn() object constructor fx

  <!-- include spec files here... -->
  <script src="spec/waveSpec.js"></script> // contains jasmine tests for waveColumn()

</head>

<body>
</body>
</html>
```

Outline Properties, Methods, + Tests

```
describe('WaveColumn()', function() {
  beforeAll(function() {
    this.waveColumn = new waveColumn();
  }
  describe('divArray', function() {
    it('is an Array', function() {
    });
    it('is initially empty',function() {
    });
  });
  describe('#initializeDivArray()', function() {
    it('populates divArray with 255 divs', function() {
    });
    it('each div has class \'line\'', function() {
    });
    it('each div is red with opacity equal to 1 / 255 of its index in divArray', function() {
    });
  });
  describe('#generateRedDiv(opacity)', function() {
    it('returns a div', function() {
    });
    it('the div has a class of \'line\'', function() {
    });
    it('the div is the color red with opacity 1 / 255 of the input parameter \'opacity\'', function() {
    });
  });
  describe('#rotateDivArray()', function() {
    it('removes the first div in divArray', function() {
    });
    it('places the removed div at the back of divArray', function() {
    });
  });
  describe('#print()', function() {
    it('appends every div in divArray to div#column in the html body', function() {
```

Jasmine

```
describe('divArray', function() {

  it('is an Array', function() {
    expect(this.waveColumn.divArray).toEqual(jasmine.any(Array))
  });

  it('is initially empty',function() {
    expect(this.waveColumn.divArray.length).toEqual(0)
  });

});
```

JavaScript

```
function WaveColumn() {
  this.divArray = [];
}
```

Jasmine

```
describe('#initializeDivArray()', function() {

  beforeAll(function() {
    this.waveColumn.initializeDivArray();
  });

  it('populates divArray with 255 divs', function() {
    expect(this.waveColumn.divArray.length).toEqual(255);
  });

  it('each div has class \'line\'', function() {
    expect(this.waveColumn.divArray[184].className).toEqual('line');
  });

  it('each div is red with opacity equal to 1 / 255 of its index in divArray', function() {
    expect(this.waveColumn.divArray[100].style.background).toEqual('rgba(255, 0, 0, 0.392157)
  });

});
```

JavaScript

```
WaveColumn.prototype.initializeDivArray = function()
  for (var i = 0; i < 255; i++) {
    this.divArray.push(this.generateRedDiv(i));
  }
}
```

Jasmine

```
describe('#generateRedDiv(opacity)', function() {
  beforeAll(function() {
    this.opacity = 42;
    this.div = this.waveColumn.generateRedDiv(this.opacity)
  });

  it('returns a div', function() {
    expect(this.div.tagName).toEqual('DIV');
  });

  it('the div has a class of \'line\'', function() {
    expect(this.div.className).toEqual('line')
  });

  it('the div is the color red with opacity 1 / 255 of the input parameter \'opacity\'', funct
    expect(this.div.style.background).toEqual('rgba(255, 0, 0, 0.164706)')
  });
});
```

JavaScript

```
WaveColumn.prototype.generateRedDiv = function(opacity) {
  var div = document.createElement('div');
  div.className = 'line';
  div.style.background = 'rgba(255,0,0,' + opacity / 255 + ')'
  return div;
}
```

Jasmine

```
describe('#rotateDivArray()', function() {
  beforeAll(function() {
    this.div = this.waveColumn.divArray[0]
    this.waveColumn.rotateDivArray();
  });

  it('removes the first div in divArray', function() {
    expect(this.waveColumn.divArray).not.toEqual(this.div);
  });

  it('places the removed div at the back of divArray', function() {
    expect(this.waveColumn.divArray[254]).toEqual(this.div);
  });
});
```

JavaScript

```
WaveColumn.prototype.rotateDivArray = function() {
  var div = this.divArray.shift();
  this.divArray.push(div);
}
```

Jasmine

```
describe('#print()', function() {
  beforeAll(function() {
    this.column = document.createElement('DIV')
    this.column.id = 'column'
    document.body.appendChild(this.column)
    this.waveColumn.print();
  });

  it('appends every div in divArray to div#column in the html body', function()
    expect(this.column.childNodes.length).toEqual(255);
  });

  afterAll(function() {
    document.body.removeChild(this.column)
  });
});
```

JavaScript

```
WaveColumn.prototype.print = function() {
  var column = document.getElementById('column');
  column.innerHTML = '';
  for (var i = 0; i < this.divArray.length; i++) {
    column.appendChild(this.divArray[i]);
  }
}
```

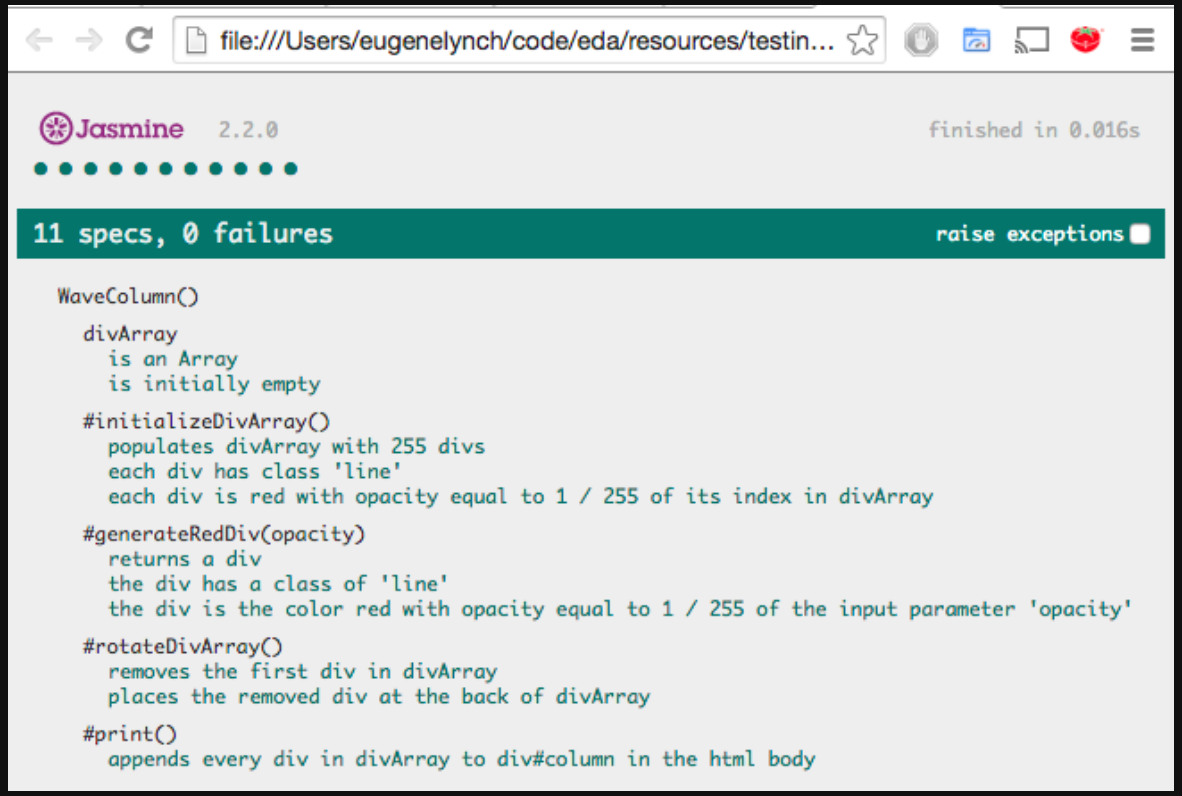
Finished JavaScript

```
function WaveColumn() {
  this.divArray = [];
}

WaveColumn.prototype.initializeDivArray = function() {
  for (var i = 0; i < 255; i++) {
    this.divArray.push(this.generateRedDiv(i));
  }
}

WaveColumn.prototype.generateRedDiv = function(opacity) {
  var div = document.createElement('div');
  div.className = 'line';
  div.style.background = 'rgba(255,0,0,' + opacity / 255 + ')';
  return div;
}

WaveColumn.prototype.rotateDivArray = function() {
  var div = this.divArray.shift();
  this.divArray.push(div);
}

WaveColumn.prototype.print = function() {
  var column = document.getElementById('column');
  column.innerHTML = '';
  for (var i = 0; i < this.divArray.length; i++) {
    column.appendChild(this.divArray[i]);
  }
}
```

Finished Jasmine Tests

```
describe('WaveColumn()', function() {
  beforeAll(function() {
    this.waveColumn = new WaveColumn();
  });

  describe('divArray', function() {

    it('is an Array', function() {
      expect(this.waveColumn.divArray).toEqual(jasmine.any(Array));
    });

    it('is initially empty',function() {
      expect(this.waveColumn.divArray.length).toEqual(0)
    });

  });

  describe('#initializeDivArray()', function() {
    beforeAll(function() {
      this.waveColumn.initializeDivArray();
    });

    it('populates divArray with 255 divs', function() {
      expect(this.waveColumn.divArray.length).toEqual(255);
    });

    it('each div has class \'line\'', function() {
      expect(this.waveColumn.divArray[184].className).toEqual('line');
    });

    it('each div is red with opacity equal to 1 / 255 of its index in divArray', function() {
      expect(this.waveColumn.divArray[100].style.background).toEqual('rgba(255, 0, 0, 0.392157)');
    });
```

Open SpecRunner.html in Chrome



And Finally ...

```
<!DOCTYPE html>

<head>
  <link href='http://fonts.googleapis.com/css?family=Roboto:300&subset=cyrillic' rel='stylesheet' type='text/css'>
  <link rel="stylesheet" type="text/css" href="./stylesheets/index.css">
</head>

<body>
  <h1>я люблю</h1>
  <div id="column"></div>
  <h1>мой кот</h1>
</body>
```

```
// index.css

body { text-align: center; font-family: 'Roboto'; }

h1 { font-size: 50px; margin: 10px auto; }

div#column { margin: 0 auto; width: 200px; }
```

http://euglazer.github.io/ilovecats/

```
// index.js
var waveColumn = new WaveColumn();
waveColumn.initializeDivArray();

var waves = setInterval(function() {
  waveColumn.rotateDivArray();
  waveColumn.print();
}, 10);
```