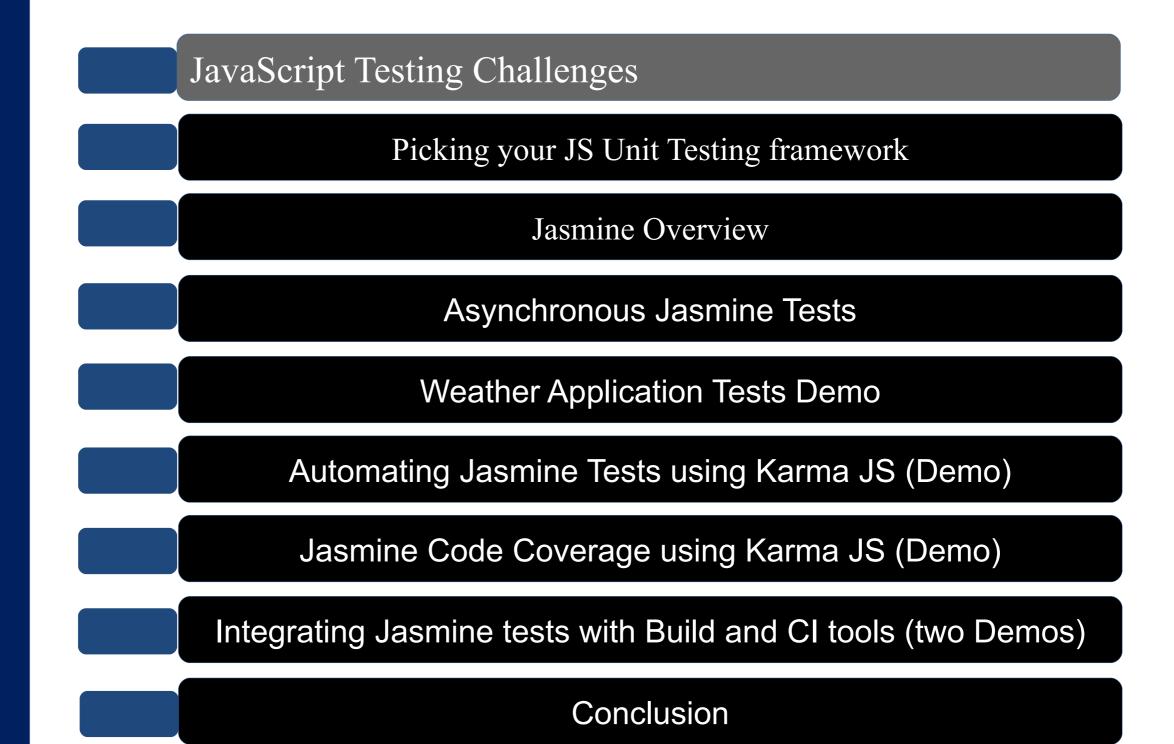


Automated Jasmine Tests for JavaScript

Hazem Saleh





geecon JavaScript Testing Challenges

Slow

Requires a lot of time to test on all the browsers.

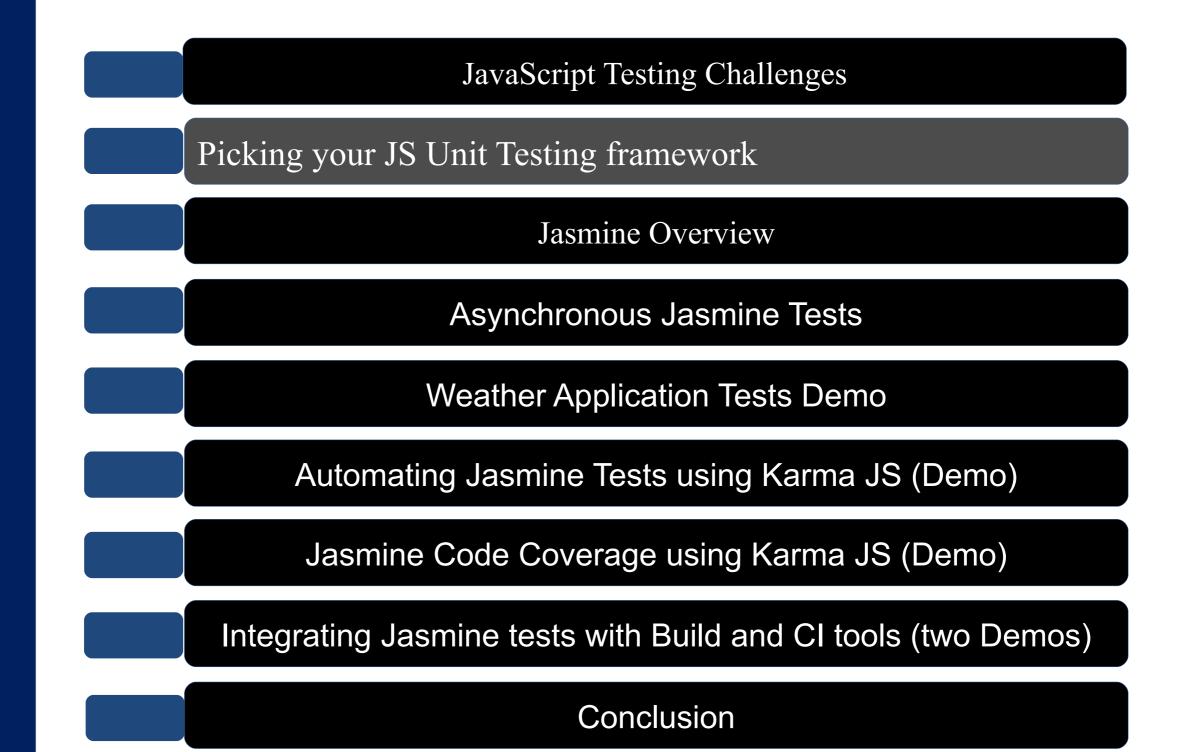
JavaScript code that works on a browser X does not mean that it will work on browser Y.

Inflexible

Supporting a new browser on an existing system means allocating a new budget:

For testing this system on this browser.

For fixing newly discovered defects on this browser.





Picking your JS Unit Testing framework

JavaScript unit testing tool

Executable across browsers (Automated preferred)

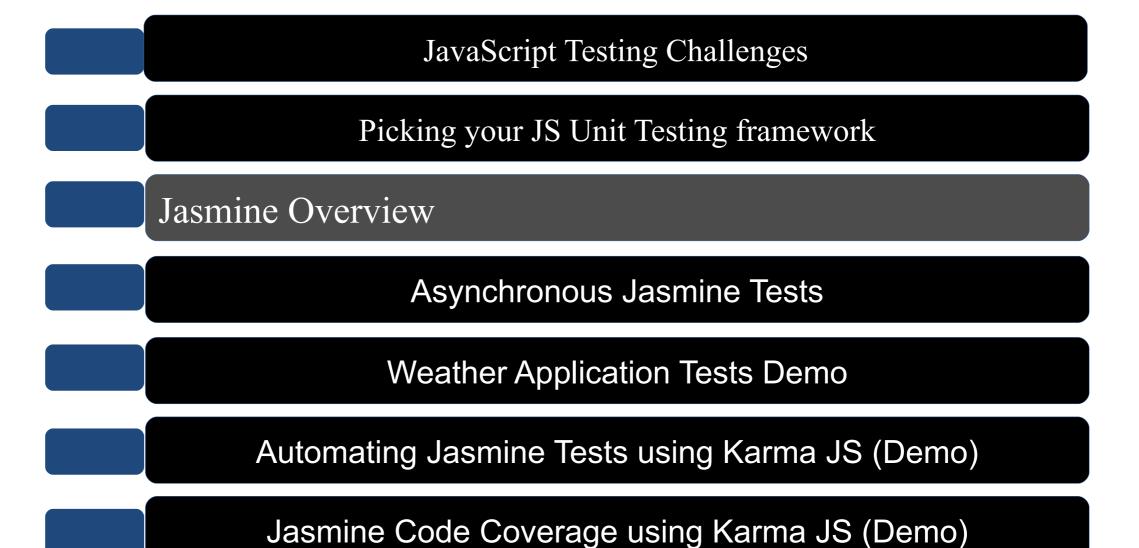
Fast Execution

Easy to setup

Integrated

Easy to configure

Provides a good testing mechanism for Asynchronous code



Integrating Jasmine tests with Build and CI tools (two Demos)

Conclusion

Jasmine is a powerful JavaScript unit testing framework.

Jasmine describes its tests in a simple natural language.

Jasmine tests can be read by Non-programmers.

Jasmine provides a clean mechanism for testing synchronous and asynchronous code.

Sample Jasmine Test:

```
describe("A sample suite", function() {
   it("contains a spec with an expectation", function() {
      expect(true).toEqual(true);
   });
});
```

Main Jasmine Constructs:

TestSuite begins with a call to describe().

TestCase "or spec" begins with a call to it().

TestCase can contain one or more matcher(s).



Jasmine Main Matchers:

expect(x).toEqual(y)

expect(x).toBe[Un]Defined()

expect(x).toBeTruthy()
expect(x).toBeFalsy()

expect(x).toBeNull()

expect(x).toBeLessThan(y)
expect(x).toBeGreaterThan(y)

expect(x).toContain(y)

expect(x).toMatch(pattern)

expect(x).toWhatEver(Y)
"custom matcher"

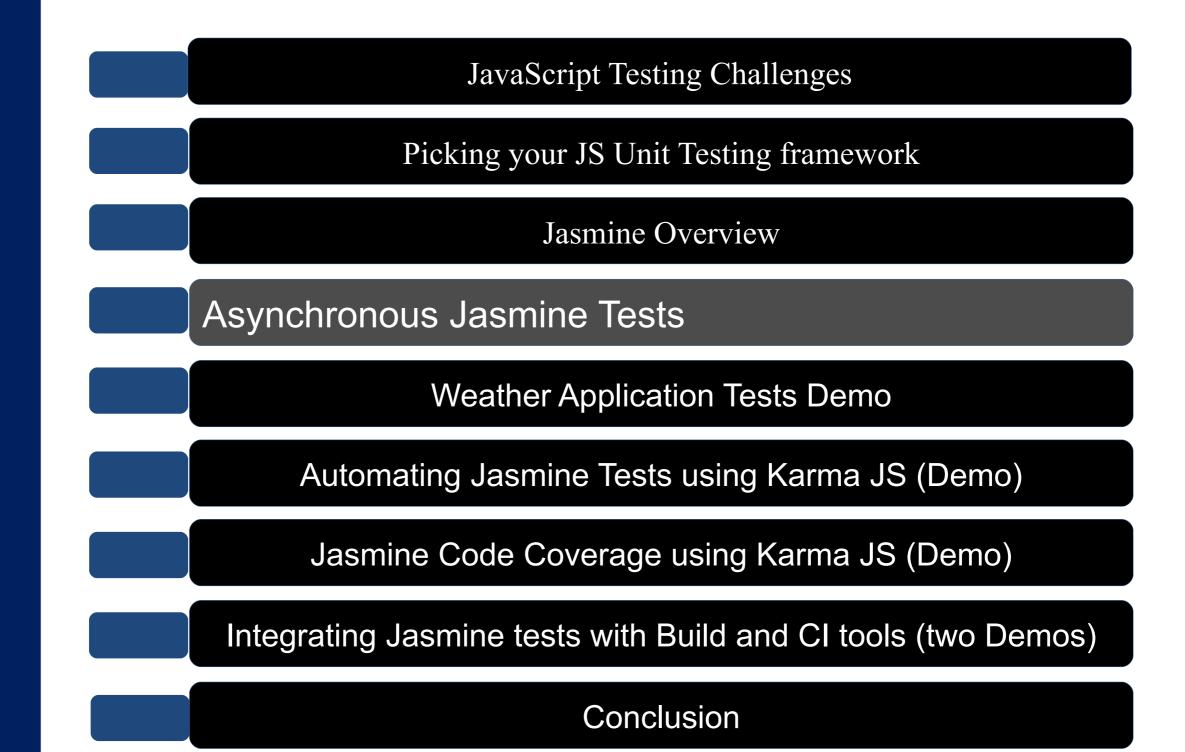


beforeEach and afterEach example:

```
describe("SimpleMath", function() {
  var simpleMath;
  beforeEach(function() {
     simpleMath = new SimpleMath();
  });
  it("should be able to find factorial for positive number", function() {
     expect(simpleMath.getFactorial(3)).toEqual(6);
  });
  it("should be able to find factorial for zero", function() {
     expect(simpleMath.getFactorial(0)).toEqual(1);
  });
  afterEach(function() {
    simpleMath = null;
  });
```









Asynchronous Jasmine Tests

Asynchronous JavaScript code refers to the code whose caller will NOT to wait until the execution completes.

In order to get the results, the caller should pass callbacks which will be called with data results parameters in case of operation success or failure.

Asynchronous JavaScript code mainly refers to Ajax code.

In order to support Asynchronous operation testing, Jasmine provides:

- 1. An optional single parameter for its single spec.
- 2. This parameter has to be called if the asynchronous operation completes.
- 3. If this parameter is not called for by default 5 seconds then the test will fail (means operation timeout).



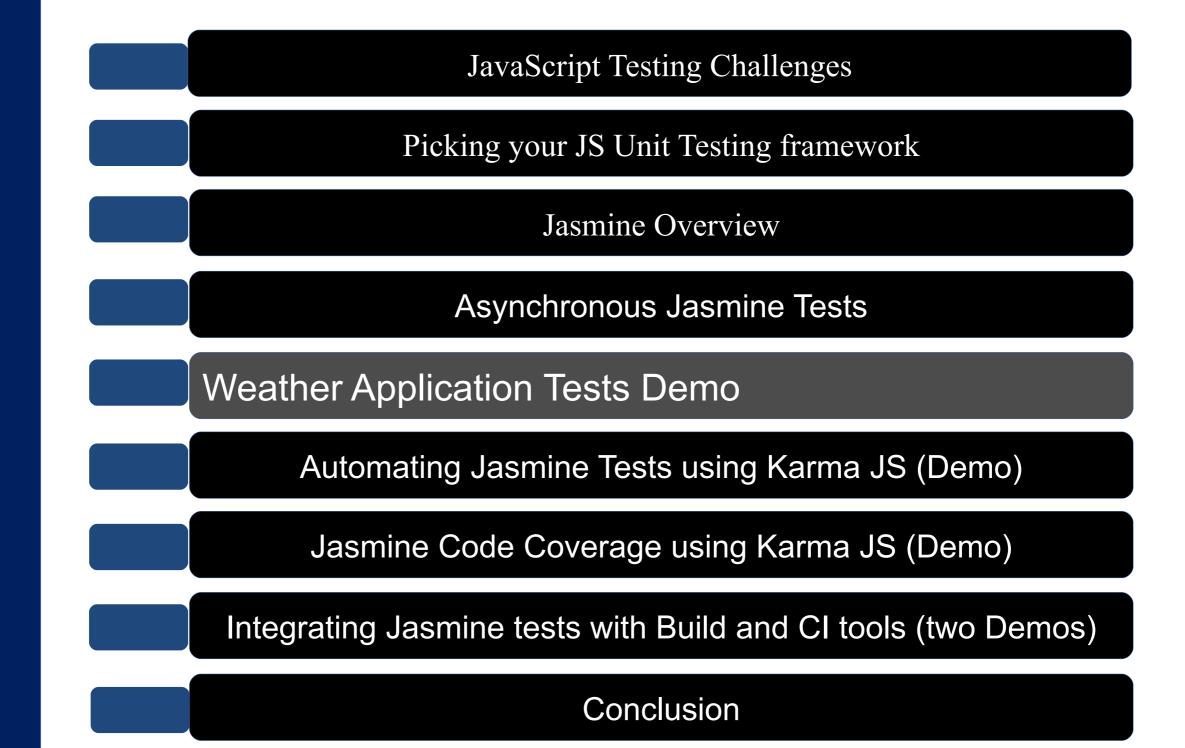
Asynchronous Jasmine Tests

```
describe("when doing asynchronous operation", function() {
 it("should be able to do the asynchronous operation", function(done) {
          var data = \{\};
          var successCallBack = function(result) {
               console.log("success");
               /* validate result parameter */
               done();
                                                          Testing an Async Operation
          };
          var failureCallBack = function() {
               console.log("failure");
               expect("Operation").toBe("passing"); /* force failing test */
               done();
          };
          AsyncObject.asyncOperation(data, successCallBack, failureCallBack);
     });
});
```

Loading Jasmine Fixtures

Fixture module of jasmine-jquery (use jQuery core) allows loading the HTML content to be used by the tests.

Put the fixtures you want to load for your tests in the spec\javascripts\fixtures directory.

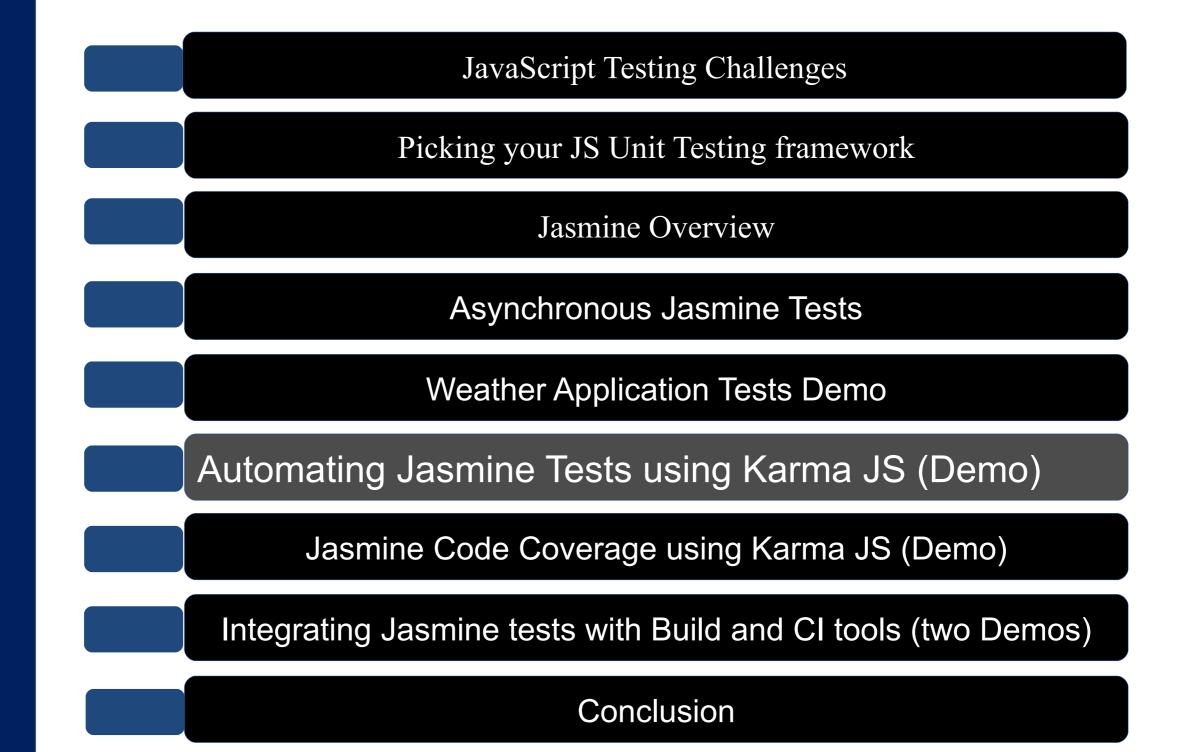




Weather Application Tests Demo

Weather Application Test Demo:

- 1. Loading Fixtures example.
- 2. Performing asynchronous operation testing example.





Jasmine is very cool and powerful JavaScript Unit Testing framework.

However, Jasmine is designed to work from browsers.

In order to automate Jasmine tests, we need to use a JavaScript test runner.

You have two great options from many options:

JsTestDriver

Karma JS



Karma JS is a modern JavaScript test runner that can be used for automating JavaScript tests.

Karma JS is designed to bring a productive test environment for web developers.

It is based on Node JS and is distributed as a node package.

It provides an easy-to-use command line interface.



In order to use Karma JS:

Install Karma JS:

Generate your test configuration file:

Start your Karma server:



npm install karma

karma init config.js

karma start config.js

Fortunately, Karma JS supports popular testing frameworks, some of them are:

Jasmine

QUnit



Sample Karma JS Jasmine configuration file:



By Default, Karma ONLY outputs the tests results on the console.

In order to output the test results in JUnit XML format, we need to install karma-junit-reporter plugin.

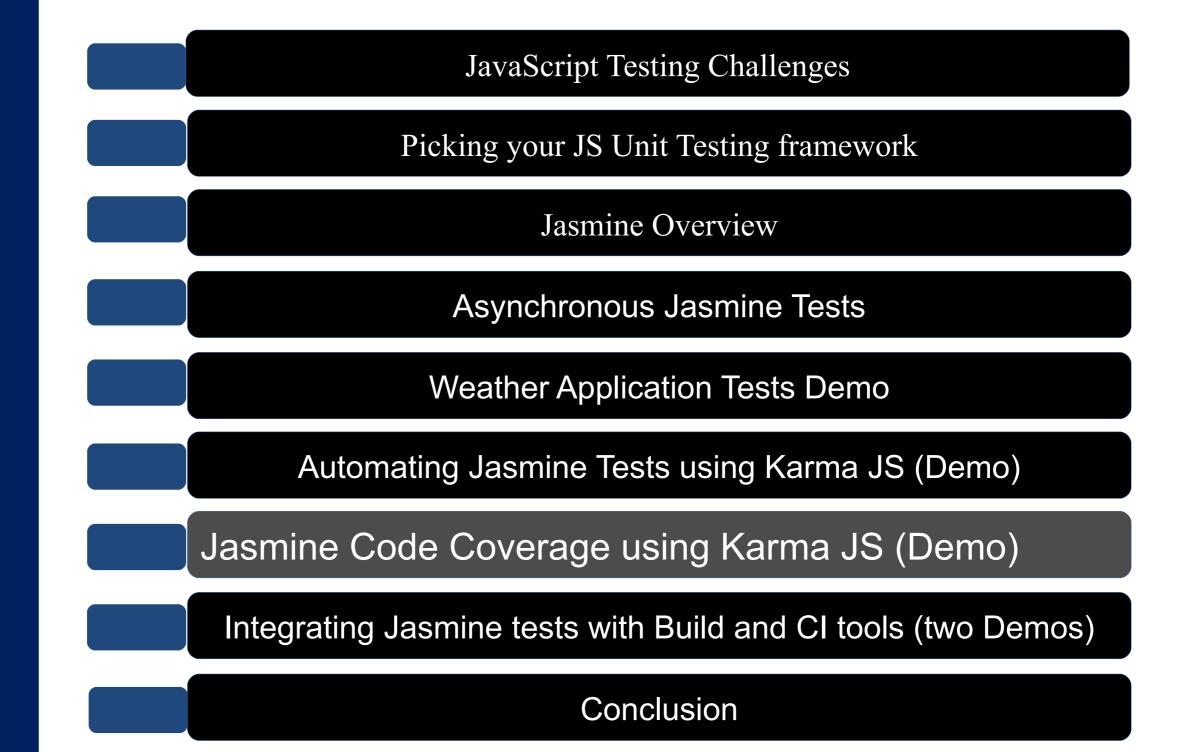
In order to include JUnit reporter plugin to your project:

npm install karma-junit-reporter --save-dev

Add JUnit reporter parameters to your configuration.



- 1. Automating running Jasmine tests in Karma.
- 2. Exploring Karma's JUnit XML results.





Jasmine Code Coverage using Karma JS

Karma JS code coverage can be performed using Istanbul module.

Istanbul supports:

Line coverage

Function coverage

Branch coverage

In order to include Istanbul in your test project:

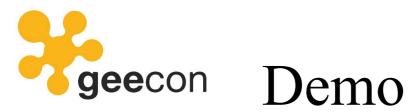
npm install karma-coverage --save-dev

Add Istanbul parameters to your configuration.



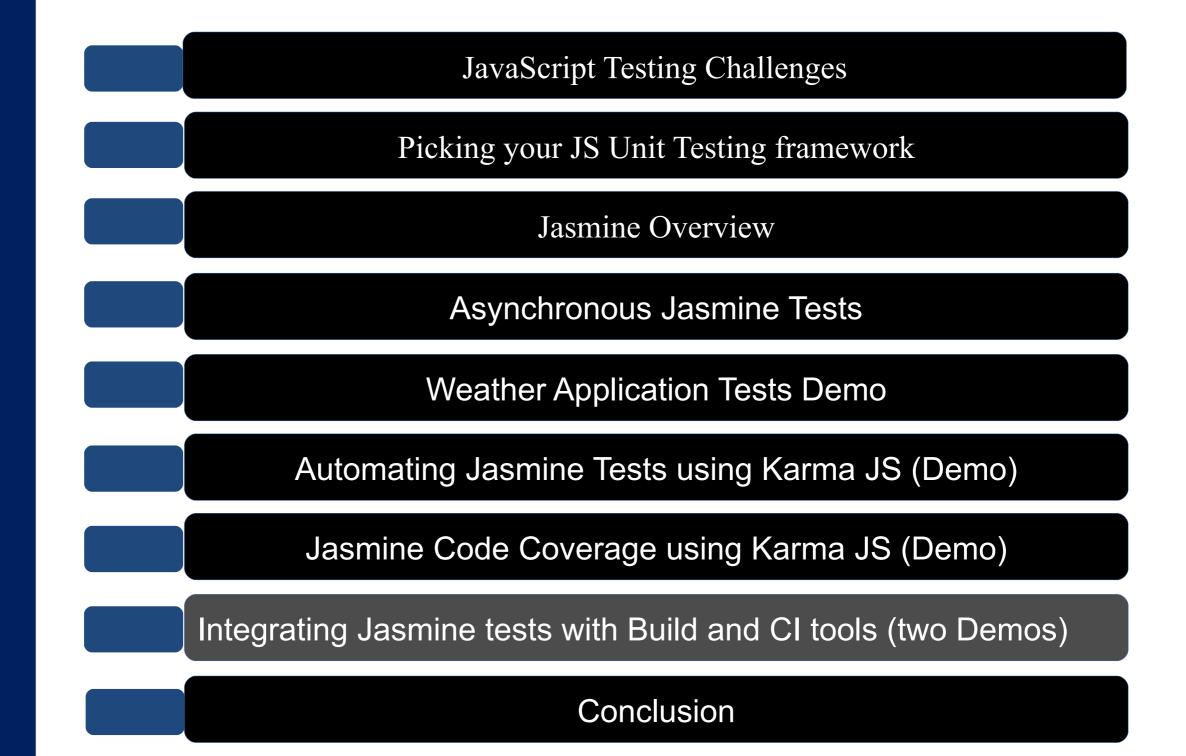
geecon Jasmine Code Coverage using Karma JS

```
module.exports = function(config) {
 config.set({
    //...
    reporters: ['progress', 'coverage'],
    preprocessors: {
      // src files to generate coverage for
      'src/*.js': ['coverage']
    },
    // The reporter configuration
    coverageReporter: {
      type : 'html',
      dir : 'coverage/'
  });
```





N E





Integrating tests with Build and Integration Management tools

N E

JavaScript Testing Challenges

Picking your JS Unit Testing framework

Jasmine Overview

Asynchronous Jasmine Tests

Weather Application Tests Demo

Automating Jasmine Tests using Karma JS (Demo)

Jasmine Code Coverage using Karma JS (Demo)

Integrating Jasmine tests with Build and CI tools (two Demos)

Conclusion

geecon Conclusion

Jasmine is a powerful unit testing framework that allows you to develop readable maintainable JavaScript tests.

Karma JS is a modern Node JS test runner which allows automating JavaScript tests.

Thanks to Karma JS, we can now have automated Jasmine tests.

Thanks to Karma JS and Jasmine, testing JavaScript code becomes fun:).