Scripting for Multimedia

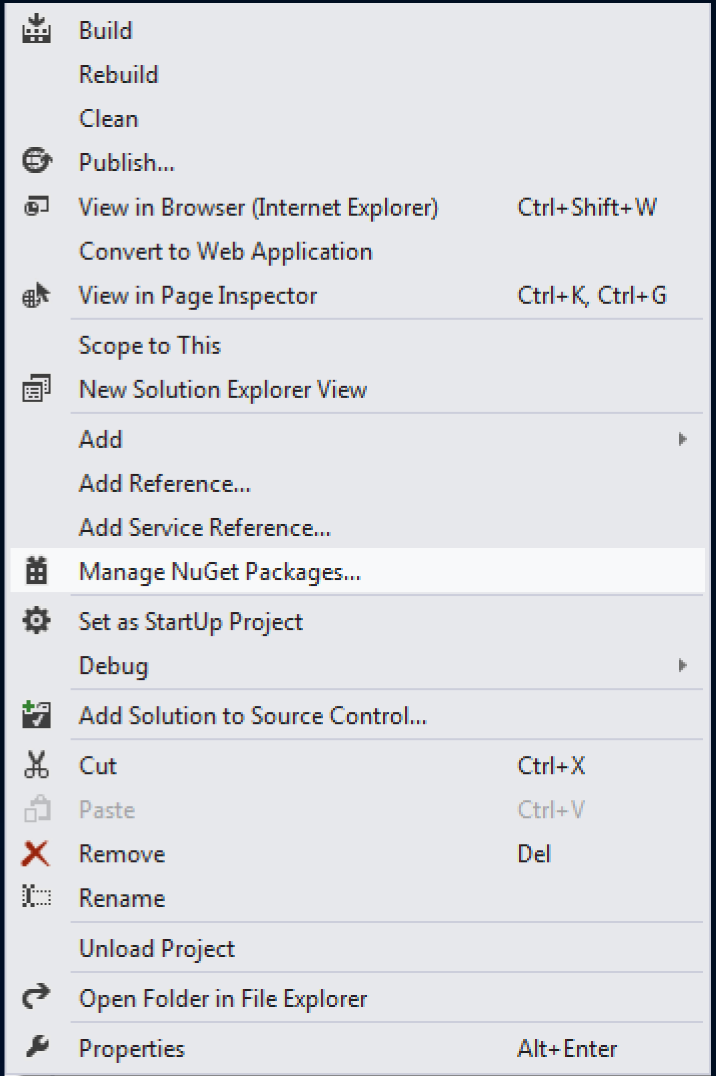
PRE-LAB 2: WRITING, TESTING, AND DEBUGGING JAVASCRIPT

Writing test-driven code

- Test-driven development (TDD) is a great way to write code and learn about code
  - You can write your test without having to write a user interface
  - It's also easy to prototype code

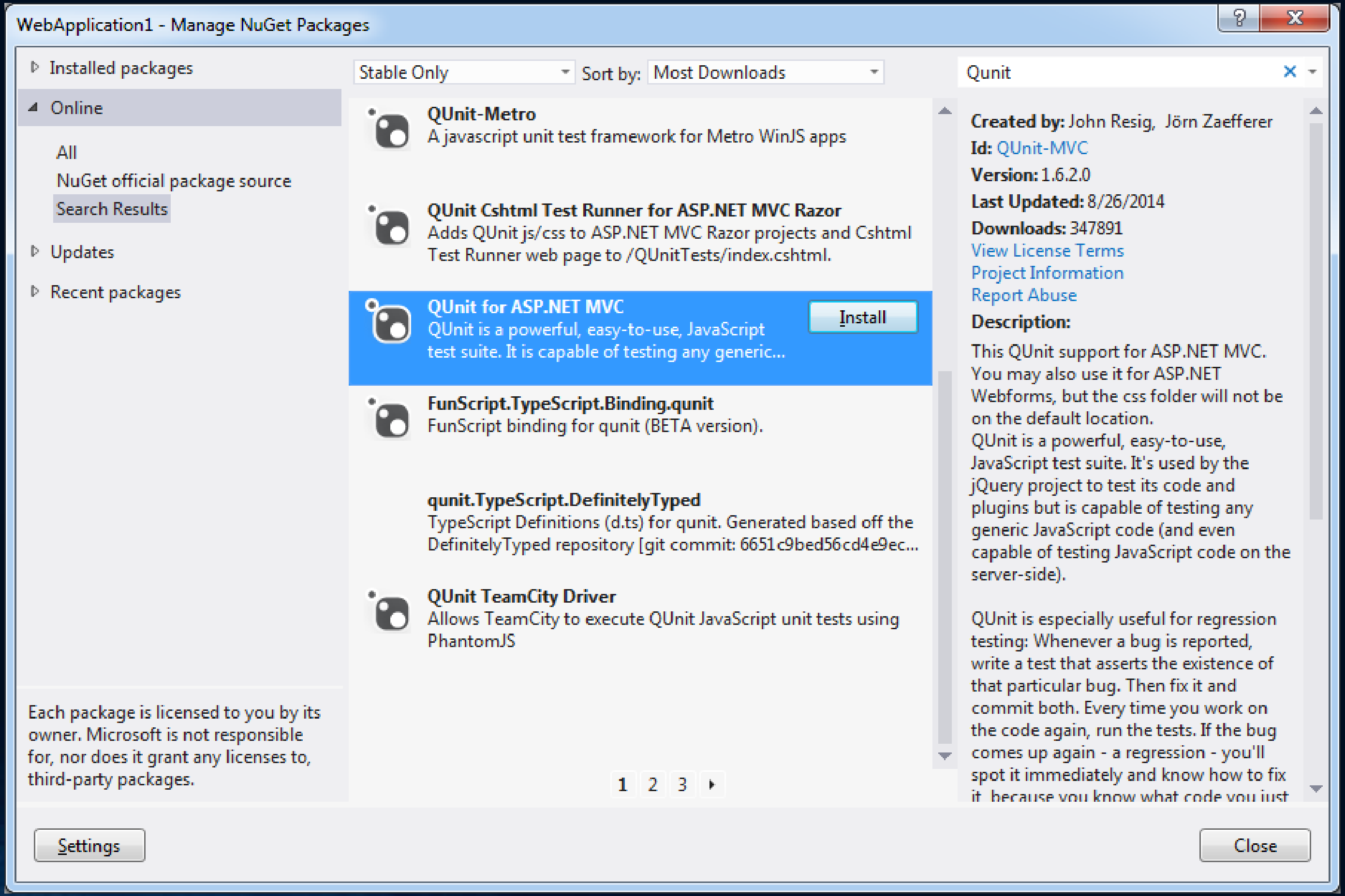
Setting up QUnit with ASP.NET application

- Create an ASP.NET Empty Web Application
- In the solution Explorer window, right-click the project node and click Manage NuGet Packages



Setting up QUnit with ASP.NET application

- Click the Online node and type QUnit in the Search Online text box
- Click the magnifying glass to perform the search
- Click the QUnit for ASP.NET MVC
- Click the Install button
- Click the Close button to close the Manage NuGet Packages screen



QUnit-Metro
A javascript unit test framework for Metro WinJS apps

QUnit Cshtml Test Runner for ASP.NET MVC Razor
Adds QUnit js/css to ASP.NET MVC Razor projects and Cshtml Test Runner web page to /QUnitTests/index.cshtml.

QUnit for ASP.NET MVC
QUnit is a powerful, easy-to-use, JavaScript test suite. It is capable of testing any generic...

FunScript.TypeScript.Binding.qunit
FunScript binding for qunit (BETA version).

qunit.TypeScript.DefinitelyTyped
TypeScript Definitions (d.ts) for qunit. Generated based off the DefinitelyTyped repository [git commit: 6651c9bed56cd4e9ec...

QUnit TeamCity Driver
Allows TeamCity to execute QUnit JavaScript unit tests using PhantomJS

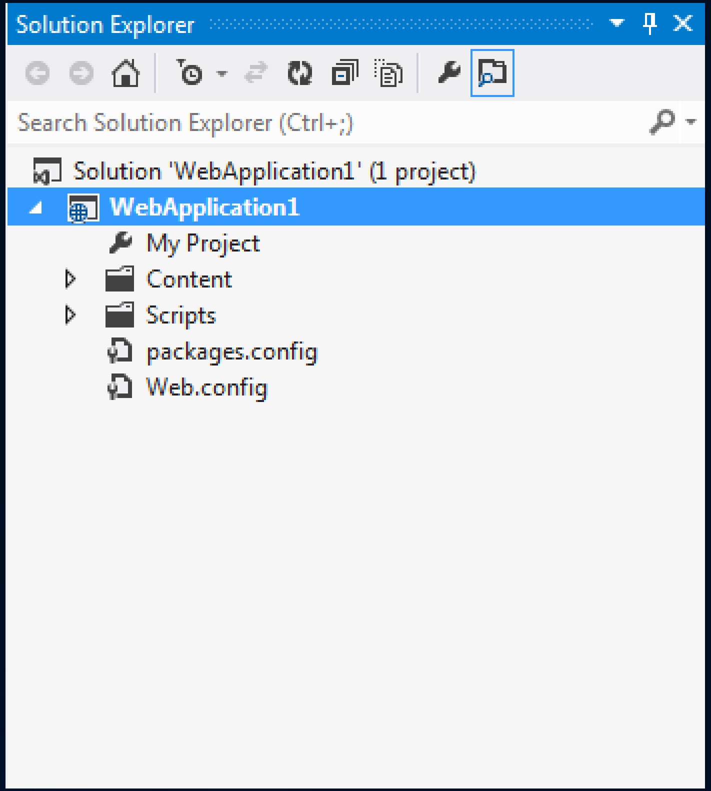
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Created by: John Resig, Jörn Zaefferer
Id: QUnit-MVC
Version: 1.6.2.0
Last Updated: 8/26/2014
Downloads: 347891
View License Terms
Project Information
Report Abuse
Description:
This QUnit support for ASP.NET MVC. You may also use it for ASP.NET Webforms, but the css folder will not be on the default location.
QUnit is a powerful, easy-to-use, JavaScript test suite. It's used by the jQuery project to test its code and plugins but is capable of testing any generic JavaScript code (and even capable of testing JavaScript code on the server-side).

QUnit is especially useful for regression testing: Whenever a bug is reported, write a test that asserts the existence of that particular bug. Then fix it and commit both. Every time you work on the code again, run the tests. If the bug comes up again - a regression - you'll spot it immediately and know how to fix it, because you know what code you just

Setting up QUnit with ASP.NET application

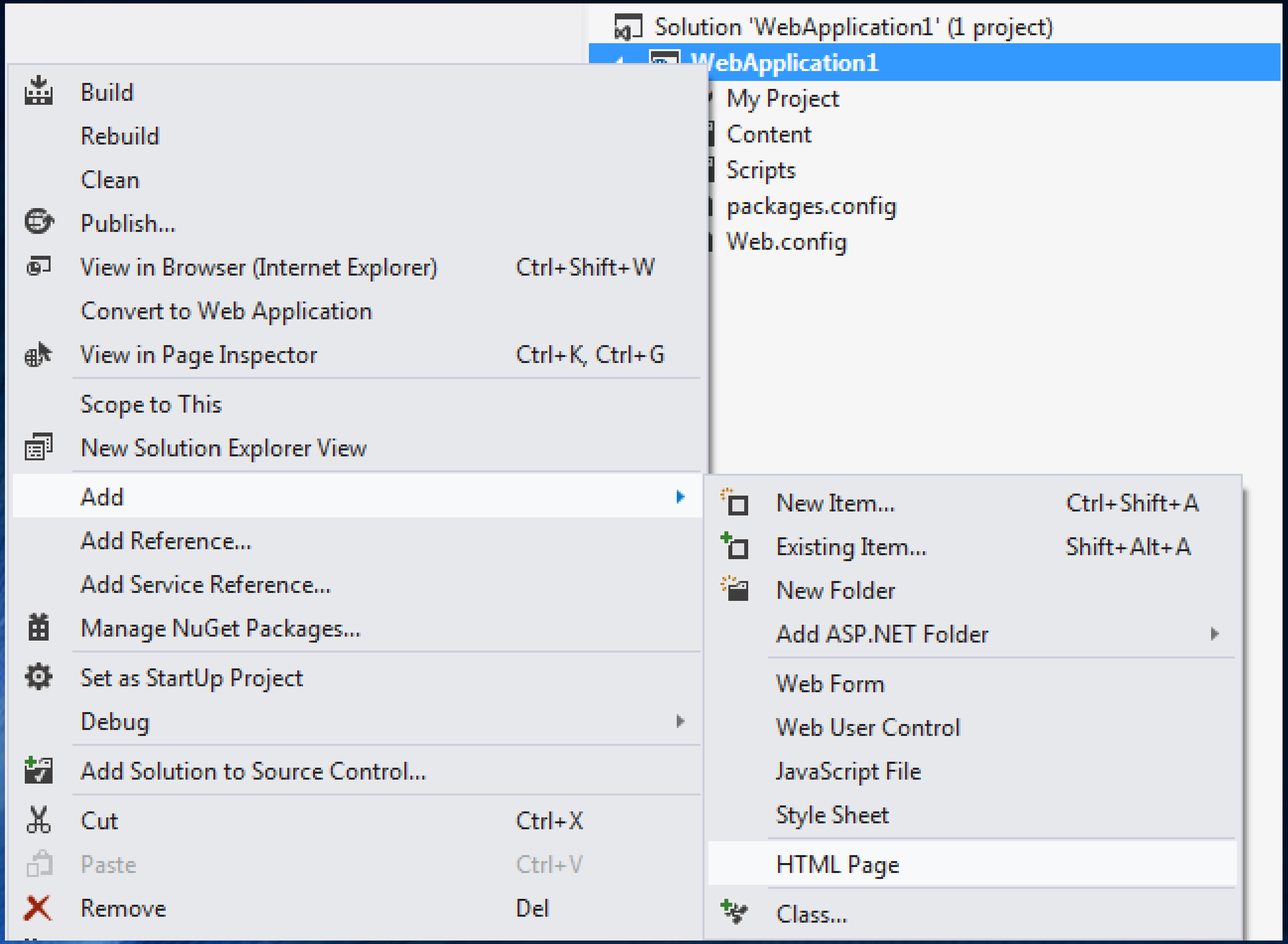
- After the QUnit for ASP.NET MVC package has been added, you see a packages.config file

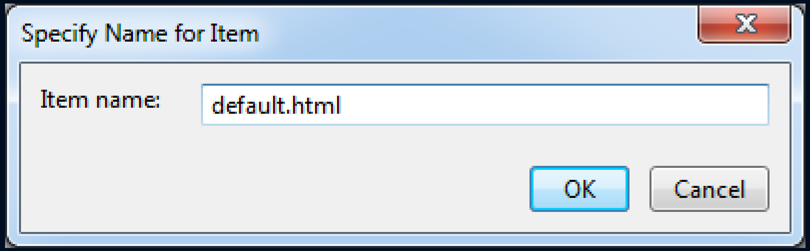


Setting up QUnit with ASP.NET application

- Right-click the project node and click Add; choose HTML Page
- Name the file default.html and click OK
- Right-click the default.html file and choosing Set As Start Page

Hello World from JavaScript





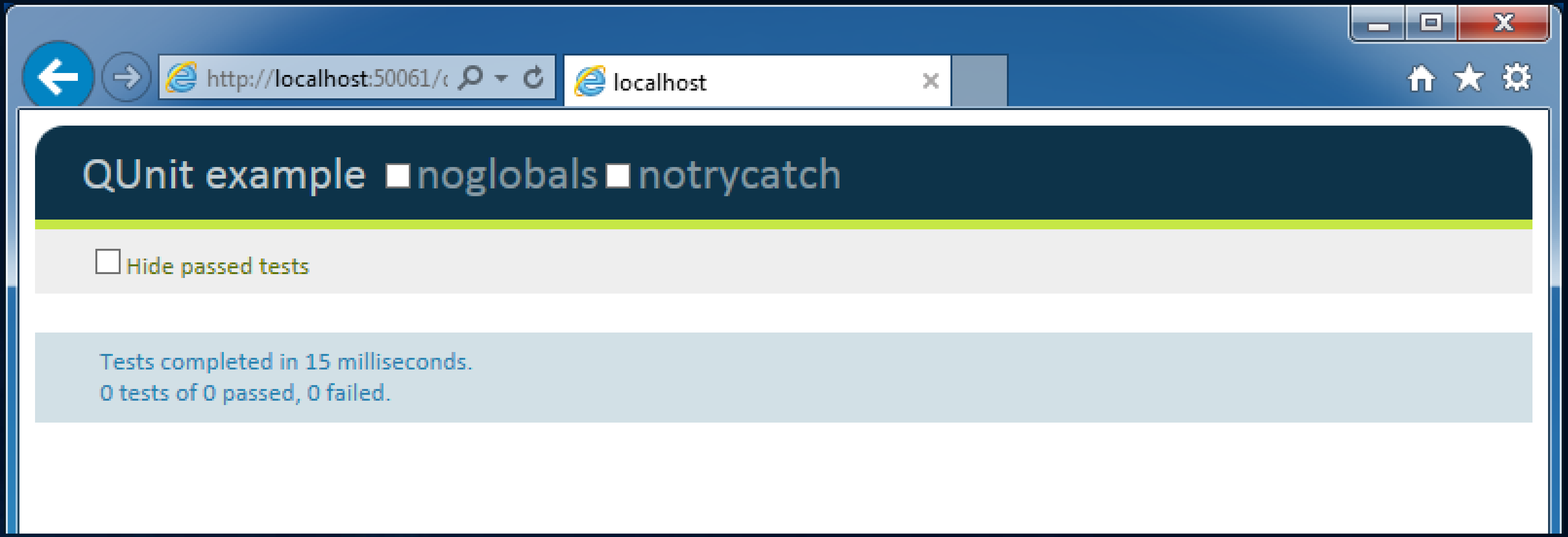
Item name: default.html

Setting up QUnit with ASP.NET application

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <link rel="stylesheet" type="text/css" href="Content/qunit.css" />
    <script type="text/javascript" src="Scripts/qunit.js"></script>
</head>
<body>
    <h1 id="qunit-header">QUnit example</h1>
    <h2 id="qunit-banner"></h2>
    <div id="qunit-testrunner-toolbar"></div>
    <h2 id="qunit-userAgen"></h2>
    <ol id="qunit-tests"></ol>
    <div id="qunit-fixture">test markup, will be hidden</div>
</body>
</html>
```

Setting up QUnit with ASP.NET application

- The QUnit setup is done
- Your code and your tests should be in separate files
- Navigating to Debug and choosing Start Debugging

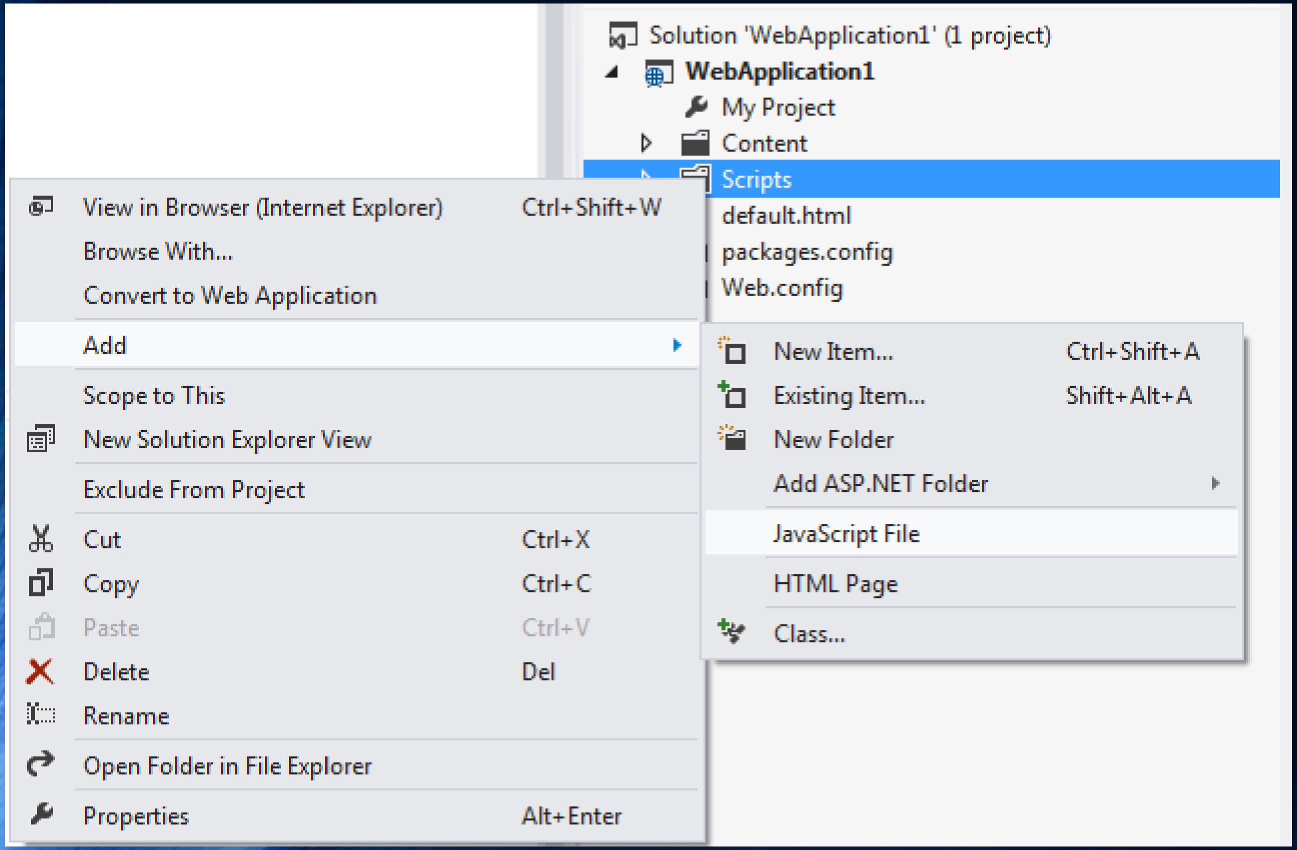


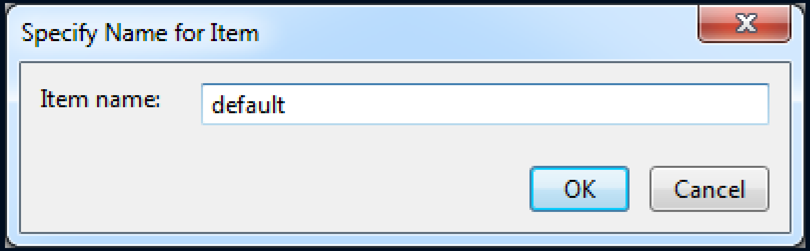
```
Tests completed in 15 milliseconds.
0 tests of 0 passed, 0 failed.
```

Hello World from JavaScript

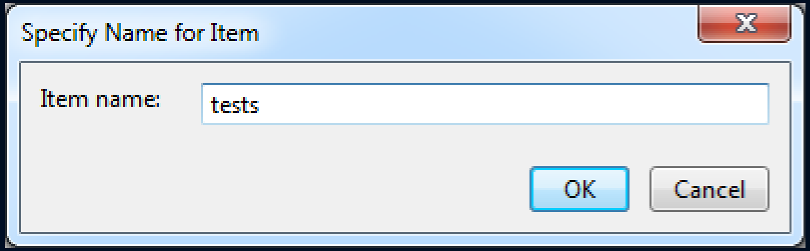
- Right-click the Scripts folder and choosing Add
- Choose the JavaScript file
- Name the file default.js and click OK
- Do the same for the tests.js file

Hello World from JavaScript





Item name: default



Item name: tests

Hello World from JavaScript

- Open the default.html
- Drag the default.js file out and drop the file right after the last ending script tag (</script>)
- Drag the tests.js file our and drop it after the last ending script tag

Hello World from JavaScript

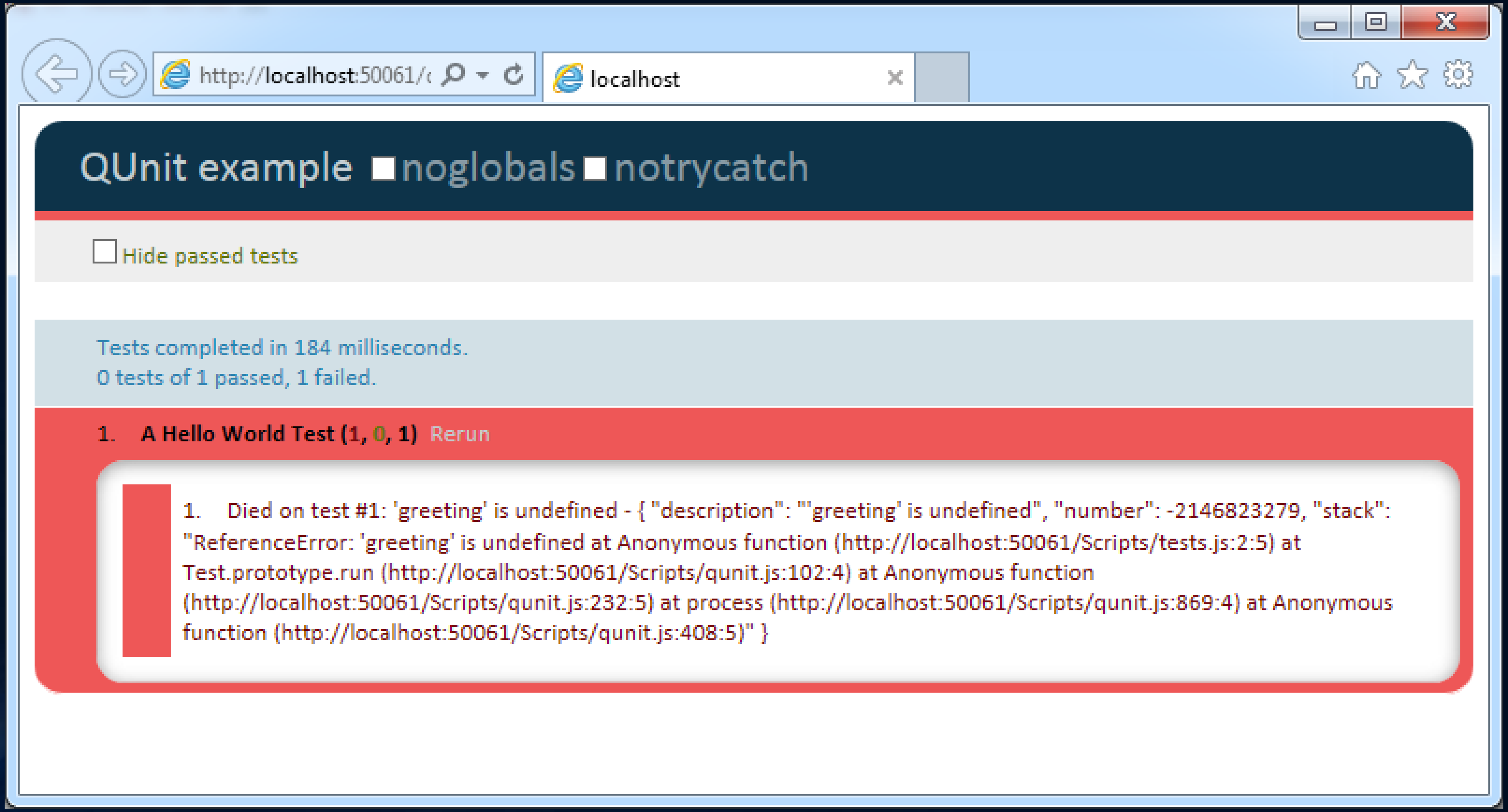
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <link rel="stylesheet" type="text/css"
href="Content/qunit.css" />
    <script type="text/javascript"
src="Scripts/qunit.js"></script>
    <script src="Scripts/default.js"></script>
    <script src="Scripts/tests.js"></script>
</head>
…
```

Hello World from JavaScript

- Now write the first test
  - When using TDD, always write the test first
- In the tests.js file add the following test to see whether a greeting variable contains Hello World:

```
test("A Hello World Test", 1, function () {
    equal(greeting, "Hello World", "Expect greeting of Hello
        World");
});
```

Hello World from JavaScript



Tests completed in 184 milliseconds.
0 tests of 1 passed, 1 failed.

1. A Hello World Test (1, 0, 1) Rerun

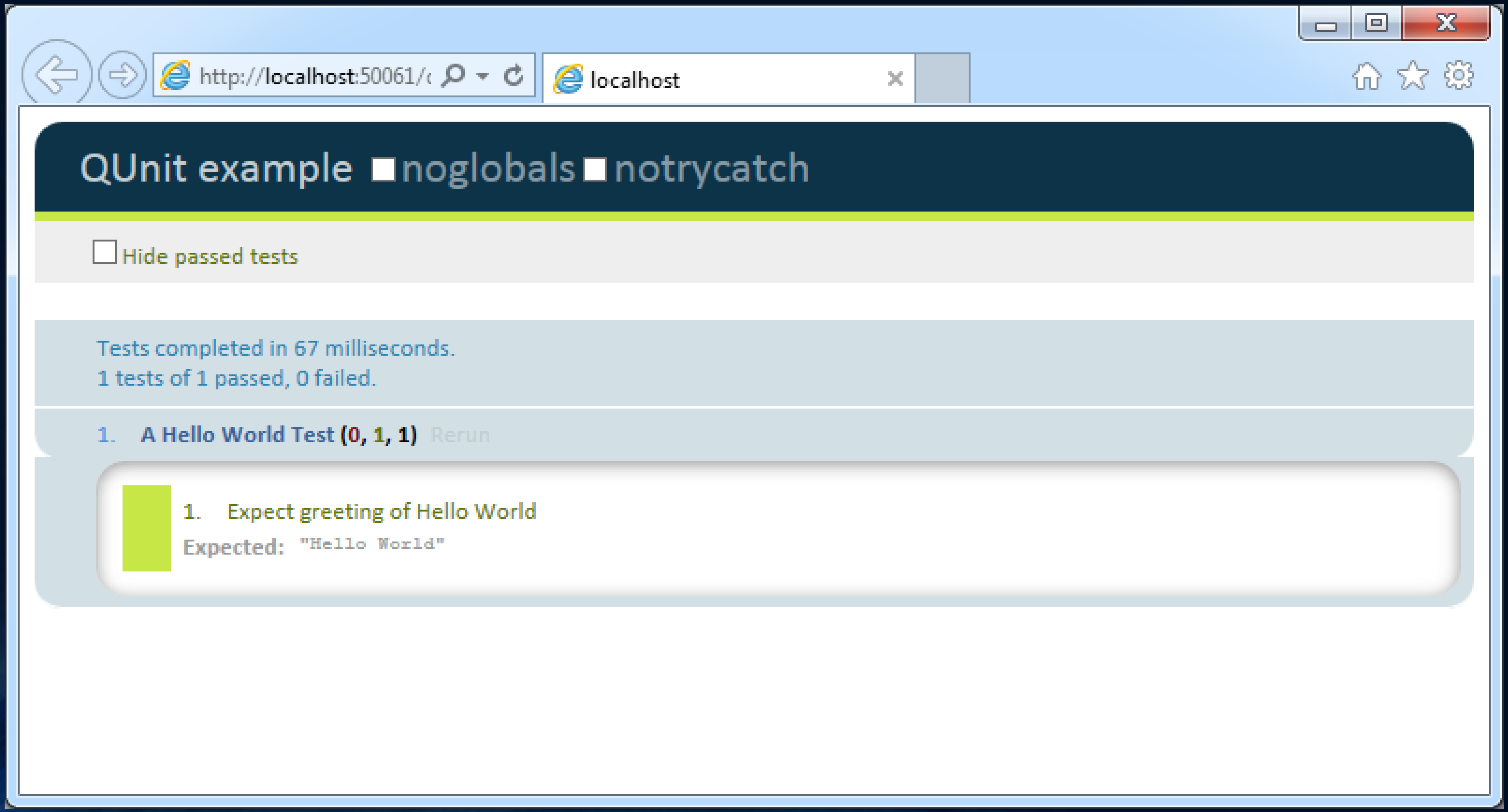
```
1. Died on test #1: 'greeting' is undefined - { "description": "'greeting' is undefined", "number": -2146823279, "stack": "ReferenceError: 'greeting' is undefined at Anonymous function (http://localhost:50061/Scripts/tests.js:2:5) at Test.prototype.run (http://localhost:50061/Scripts/qunit.js:102:4) at Anonymous function (http://localhost:50061/Scripts/qunit.js:232:5) at process (http://localhost:50061/Scripts/qunit.js:869:4) at Anonymous function (http://localhost:50061/Scripts/qunit.js:408:5)" }
```

Hello World from JavaScript

- The test failed because the *greeting* variable has not been created
- To make the test pass, declare a *greeting* variable and assign a value of Hello World in the default.js file:

```
var greeting = 'Hello World';
```

Hello World from JavaScript



QUnit example ■ noglobals ■ notrycatch

☐ Hide passed tests

Tests completed in 67 milliseconds.
1 tests of 1 passed, 0 failed.

1. A Hello World Test (0, 1, 1) Rerun

1. Expect greeting of Hello World
Expected: "Hello World"

Using the script tag

- Inline JavaScript code
  - Example

```
<script type="text/javascript">
<!--
   function Add(x, y) {
      return x + y;
   }
   alert(Add(3, 2));
//-->
</script>
```

Using the script tag

- Referencing an external JavaScript file
  - Example

```
<script type="text/javascript" src="Scripts/tests.js"></script>
```

- Two attributes applied for external JS files
  - async
  - defer

Handling browsers that don't support JS

- When a browser doesn't support the <script> element, use the <noscript> element to specify alternate content
  - Example

```
<script type="text/javascript">
<!--
   function Add(x, y) {
      return x + y;
   }
   alert(Add(3, 2));
//-->
</script>
<noscript>Your browser does not support JavaScript so page
functionality will be significantly reduced.</noscript>
```

Placing your script elements

- Place <script> elements within <head>?
  - The browser will stop parsing the rest of the HTML doc until retrieving and executing the JS file --> empty browser window
- Put <script> **at the end of** the HTML doc and **before** </body> tag
  - Put <script> in <head> if you have JS that must exist early so the page can render properly
  - Place external references after style sheet references so the browser attempts to load both at the same time

Using VS .NET JS debugger

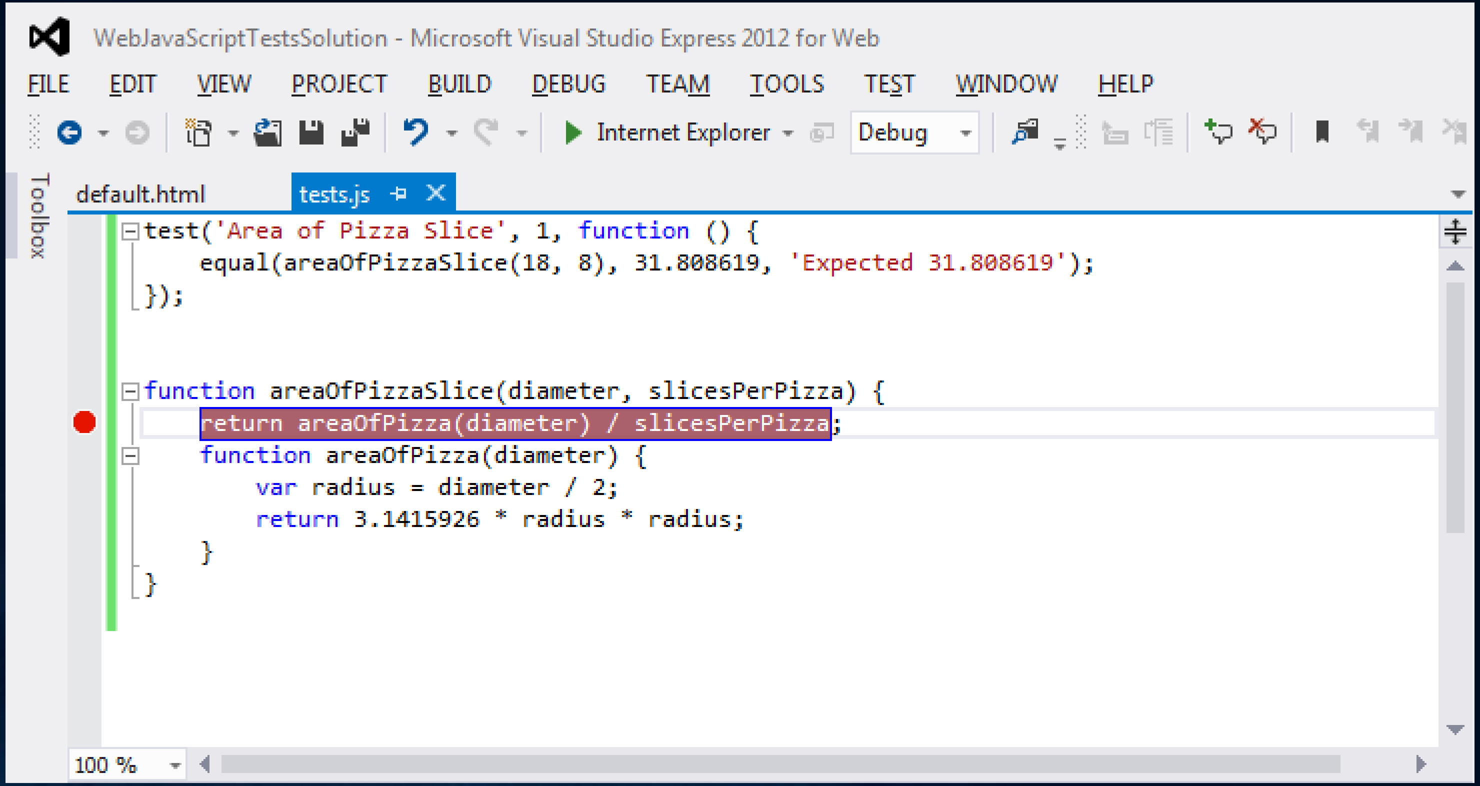
- Example

```
test('Area of Pizza Slice', 1, function() {
   equal(areaOfPizzaSlice(18, 8), 31.808619, 'Expected 31.808619');
});

function areaOfPizzaSlice(diameter, slicesPerPizza) {
     return areaOfPizza(diameter) / slicesPerPizza;
     function areaOfPizza(diameter)  {
          var radius = diameter / 2;
          return 3.1415926 * radius * radius;
     }
}
```

Using VS .NET JS debugger

- Setting a breakpoint

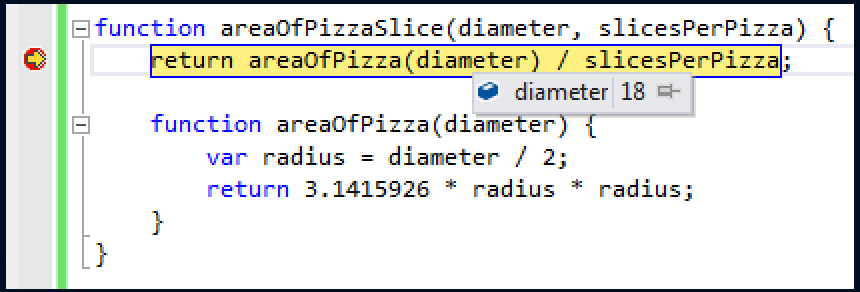

WebJavaScriptTestsSolution - Microsoft Visual Studio Express 2012 for Web
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Internet Explorer
Debug
Toolbox
default.html
tests.js
100 %

```
test('Area of Pizza Slice', 1, function () {
    equal(areaOfPizzaSlice(18, 8), 31.808619, 'Expected 31.808619');
});


function areaOfPizzaSlice(diameter, slicesPerPizza) {
    return areaOfPizza(diameter) / slicesPerPizza;
    function areaOfPizza(diameter) {
        var radius = diameter / 2;
        return 3.1415926 * radius * radius;
    }
}
```

Using VS .NET JS debugger

- Examine variables

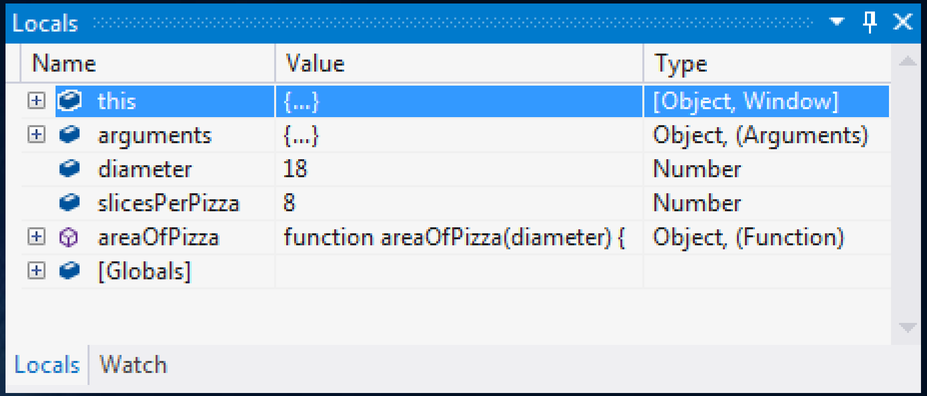


```
function areaOfPizzaSlice(diameter, slicesPerPizza) {
    return areaOfPizza(diameter) / slicesPerPizza;
                                  diameter 18

    function areaOfPizza(diameter) {
        var radius = diameter / 2;
        return 3.1415926 * radius * radius;
    }
}
```

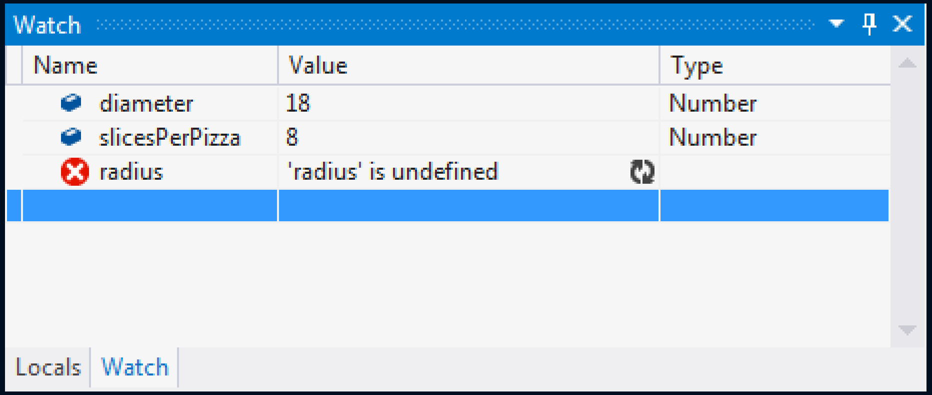
Using VS .NET JS debugger

- Examine variables



Locals

Name	Value	Type
⊞ this	{...}	[Object, Window]
⊞ arguments	{...}	Object, (Arguments)
diameter	18	Number
slicesPerPizza	8	Number
⊞ areaOfPizza	function areaOfPizza(diameter) {	Object, (Function)
⊞ [Globals]		

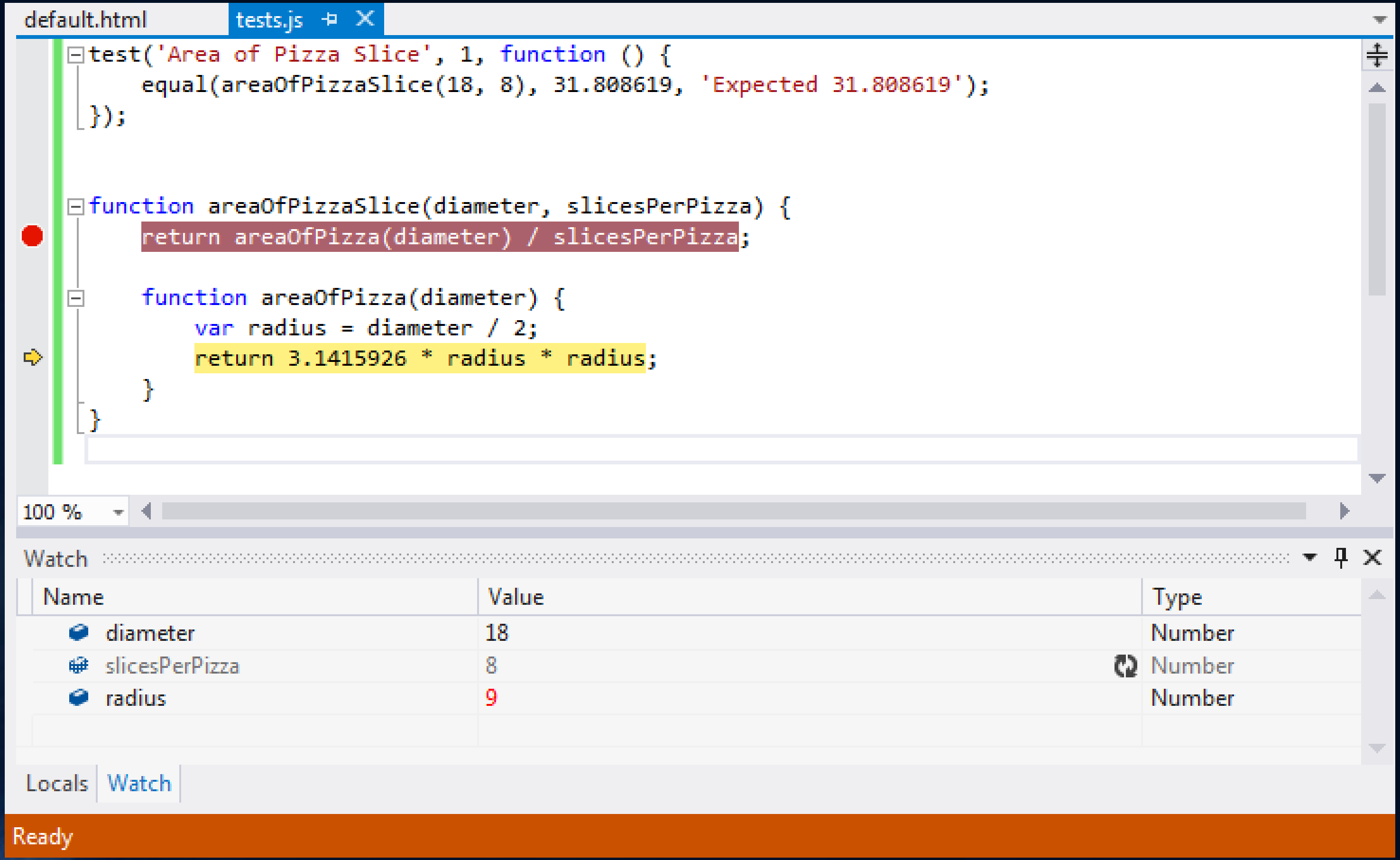


Watch

Name	Value	Type
diameter	18	Number
slicesPerPizza	8	Number
radius	'radius' is undefined	

Using VS .NET JS debugger

- Stepping through the code
  - F11 (Debug | Step into)
  - F10 (Debug | Step Over)
  - Shitf+F11 (Debug | Step Out)



```
test('Area of Pizza Slice', 1, function () {
    equal(areaOfPizzaSlice(18, 8), 31.808619, 'Expected 31.808619');
});


function areaOfPizzaSlice(diameter, slicesPerPizza) {
    return areaOfPizza(diameter) / slicesPerPizza;

    function areaOfPizza(diameter) {
        var radius = diameter / 2;
        return 3.1415926 * radius * radius;
    }
}
```

Name	Value	Type
diameter	18	Number
slicesPerPizza	8	Number
radius	9	Number