Refactoring

John Papa

@john_papa | www.johnpapa.net

What Is Refactoring?

Coding | Finding | Changing

Some of the Refactorings

Bracket matching

Selection

Cursors

Intellisense

Parameter hints

Emmet

Snippets

Go to definition or symbol

Gutter indicators

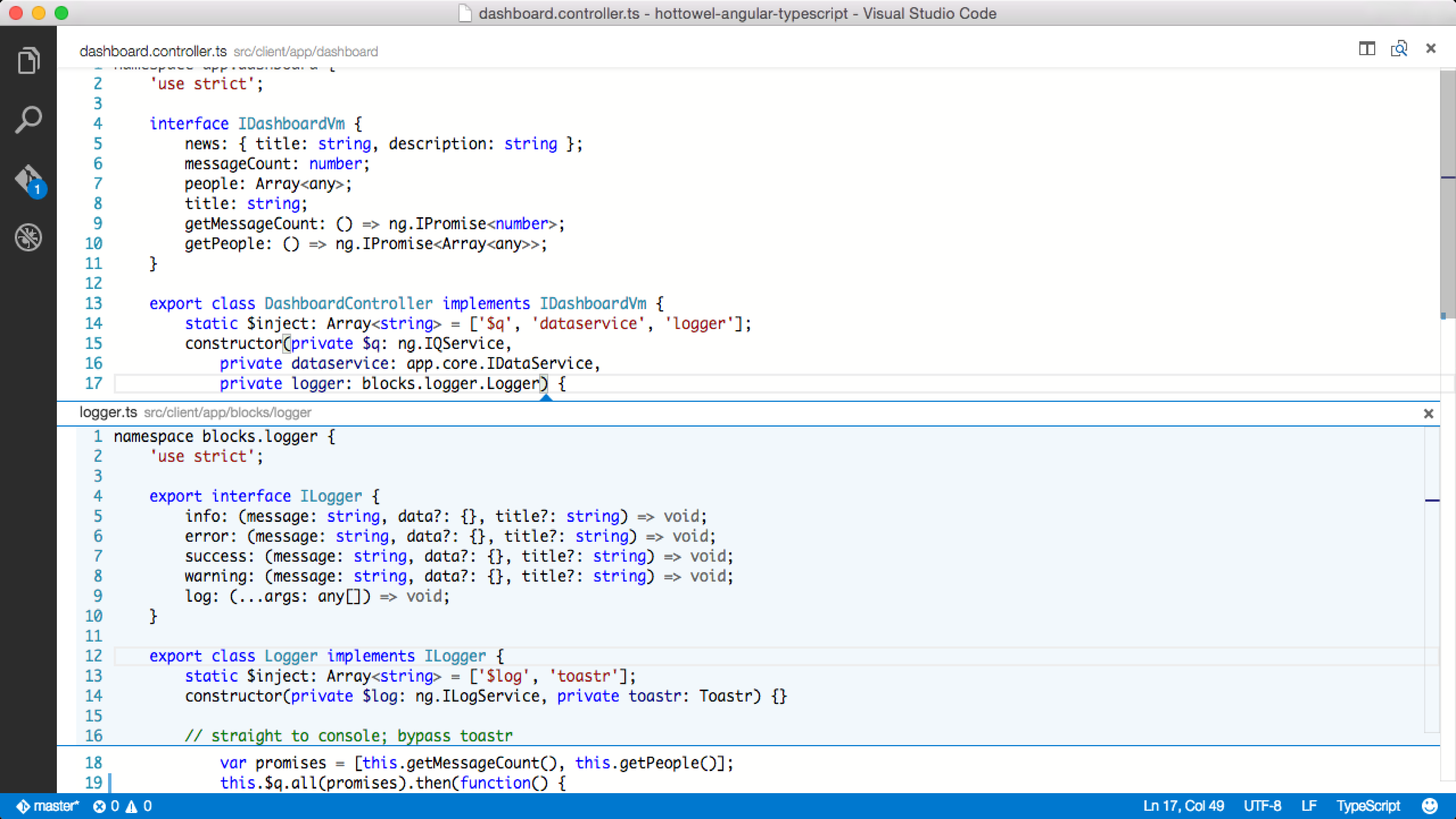
Peek

Hover

Renaming

Code actions

Errors / Warnings



dashboard.controller.ts - hottowel-angular-typescript - Visual Studio Code

dashboard.controller.ts src/client/app/dashboard

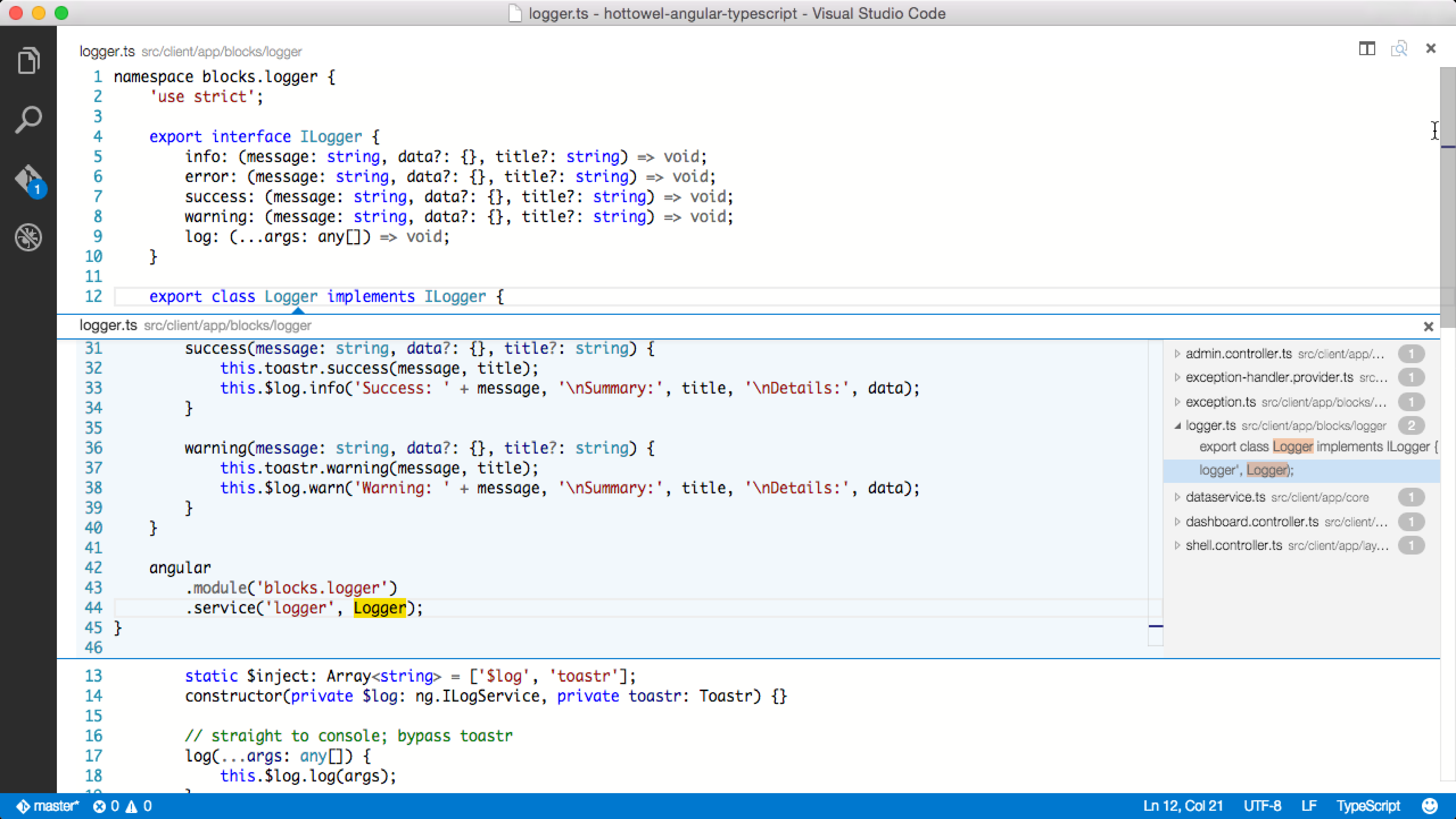
```
namespace app.dashboard {
    'use strict';

    interface IDashboardVm {
        news: { title: string, description: string };
        messageCount: number;
        people: Array<any>;
        title: string;
        getMessageCount: () => ng.IPromise<number>;
        getPeople: () => ng.IPromise<Array<any>>;
    }

    export class DashboardController implements IDashboardVm {
        static $inject: Array<string> = ['$q', 'dataservice', 'logger'];
        constructor(private $q: ng.IQService,
            private dataservice: app.core.IDataService,
            private logger: blocks.logger.Logger) {
```

logger.ts src/client/app/blocks/logger

```
namespace blocks.logger {
    'use strict';

    export interface ILogger {
        info: (message: string, data?: {}, title?: string) => void;
        error: (message: string, data?: {}, title?: string) => void;
        success: (message: string, data?: {}, title?: string) => void;
        warning: (message: string, data?: {}, title?: string) => void;
        log: (...args: any[]) => void;
    }

    export class Logger implements ILogger {
        static $inject: Array<string> = ['$log', 'toastr'];
        constructor(private $log: ng.ILogService, private toastr: Toastr) {}

        // straight to console; bypass toastr
```

```
            var promises = [this.getMessageCount(), this.getPeople()];
            this.$q.all(promises).then(function() {
```

master* ⊗ 0 ⚠ 0   Ln 17, Col 49 UTF-8 LF TypeScript



logger.ts - hottowel-angular-typescript - Visual Studio Code

logger.ts src/client/app/blocks/logger

```
namespace blocks.logger {
    'use strict';

    export interface ILogger {
        info: (message: string, data?: {}, title?: string) => void;
        error: (message: string, data?: {}, title?: string) => void;
        success: (message: string, data?: {}, title?: string) => void;
        warning: (message: string, data?: {}, title?: string) => void;
        log: (...args: any[]) => void;
    }

    export class Logger implements ILogger {
```

logger.ts src/client/app/blocks/logger

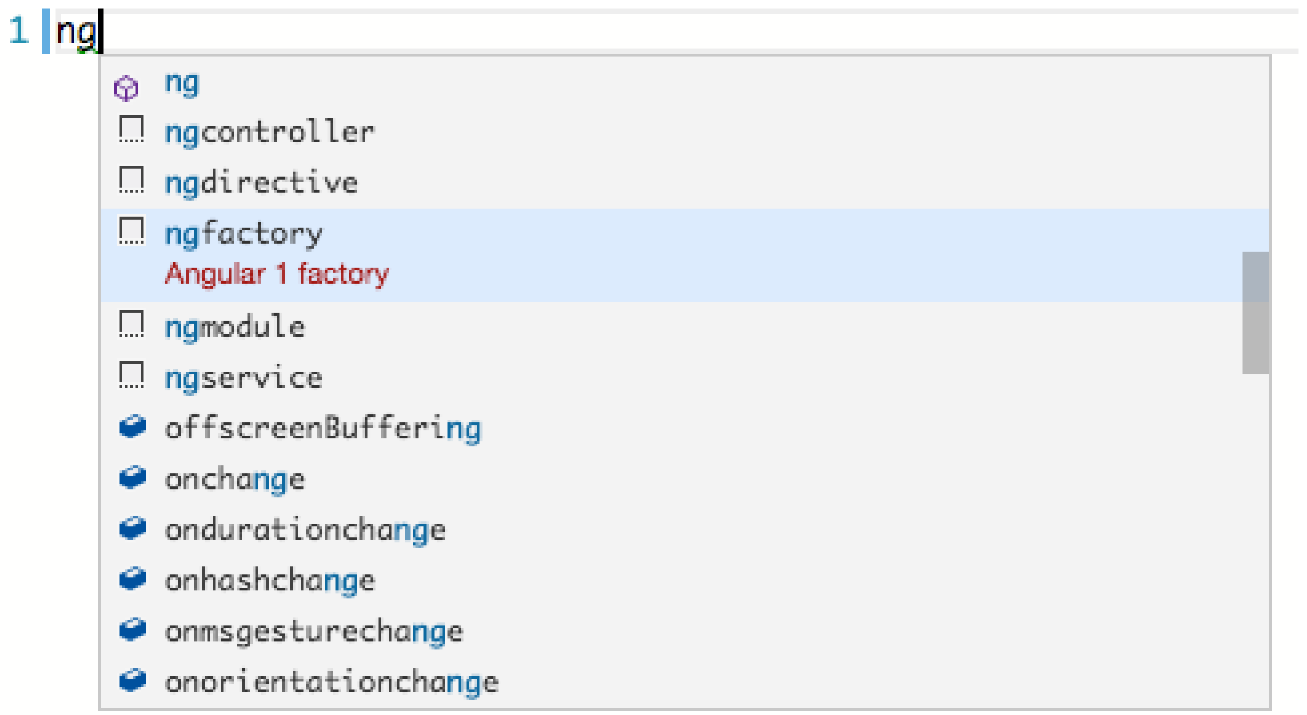
```
        success(message: string, data?: {}, title?: string) {
            this.toastr.success(message, title);
            this.$log.info('Success: ' + message, '\nSummary:', title, '\nDetails:', data);
        }

        warning(message: string, data?: {}, title?: string) {
            this.toastr.warning(message, title);
            this.$log.warn('Warning: ' + message, '\nSummary:', title, '\nDetails:', data);
        }
    }

    angular
        .module('blocks.logger')
        .service('logger', Logger);
}

```

- admin.controller.ts src/client/app/... 1
- exception-handler.provider.ts src... 1
- exception.ts src/client/app/blocks/... 1
- logger.ts src/client/app/blocks/logger 2
  - export class Logger implements ILogger {
  - logger', Logger);
- dataservice.ts src/client/app/core 1
- dashboard.controller.ts src/client/... 1
- shell.controller.ts src/client/app/lay... 1

```
        static $inject: Array<string> = ['$log', 'toastr'];
        constructor(private $log: ng.ILogService, private toastr: Toastr) {}

        // straight to console; bypass toastr
        log(...args: any[]) {
            this.$log.log(args);
```

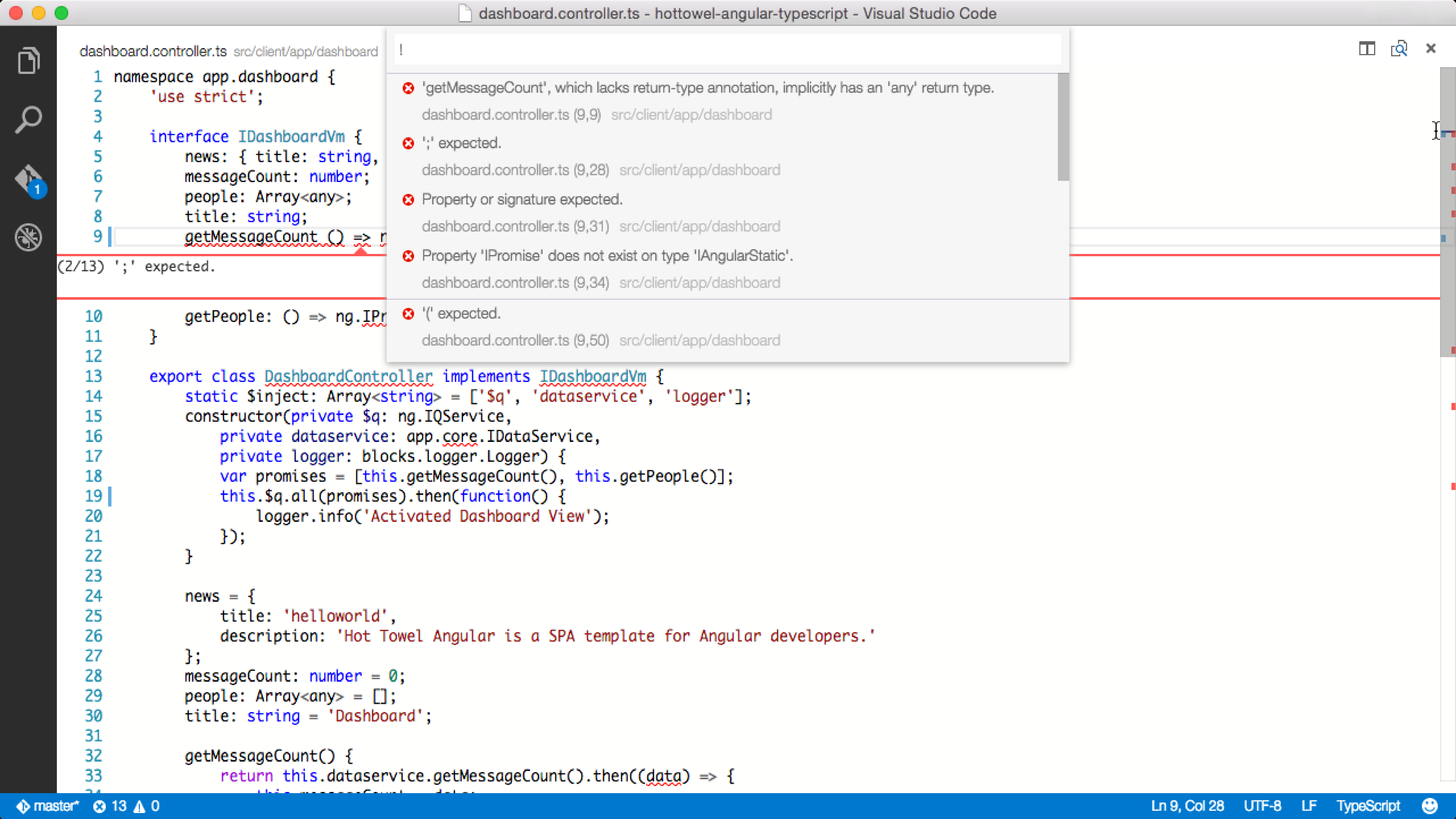
master* ⊗ 0 ⚠ 0 Ln 12, Col 21 UTF-8 LF TypeScript

Snippets



```
ng
```

ng
ngcontroller
ngdirective
ngfactory
Angular 1 factory
ngmodule
ngservice
offscreenBuffering
onchange
ondurationchange
onhashchange
onmsgesturechange
onorientationchange



dashboard.controller.ts - hottowel-angular-typescript - Visual Studio Code

dashboard.controller.ts src/client/app/dashboard

```
!
```

- 'getMessageCount', which lacks return-type annotation, implicitly has an 'any' return type.
  dashboard.controller.ts (9,9) src/client/app/dashboard
- ';' expected.
  dashboard.controller.ts (9,28) src/client/app/dashboard
- Property or signature expected.
  dashboard.controller.ts (9,31) src/client/app/dashboard
- Property 'IPromise' does not exist on type 'IAngularStatic'.
  dashboard.controller.ts (9,34) src/client/app/dashboard
- '(' expected.
  dashboard.controller.ts (9,50) src/client/app/dashboard

```
 1 namespace app.dashboard {
 2     'use strict';
 3 
 4     interface IDashboardVm {
 5         news: { title: string,
 6         messageCount: number;
 7         people: Array<any>;
 8         title: string;
 9         getMessageCount () => n
(2/13) ';' expected.
10         getPeople: () => ng.IPr
11     }
12 
13     export class DashboardController implements IDashboardVm {
14         static $inject: Array<string> = ['$q', 'dataservice', 'logger'];
15         constructor(private $q: ng.IQService,
16             private dataservice: app.core.IDataService,
17             private logger: blocks.logger.Logger) {
18             var promises = [this.getMessageCount(), this.getPeople()];
19             this.$q.all(promises).then(function() {
20                 logger.info('Activated Dashboard View');
21             });
22         }
23 
24         news = {
25             title: 'helloworld',
26             description: 'Hot Towel Angular is a SPA template for Angular developers.'
27         };
28         messageCount: number = 0;
29         people: Array<any> = [];
30         title: string = 'Dashboard';
31 
32         getMessageCount() {
33             return this.dataservice.getMessageCount().then((data) => {
```

master* ⊗ 13 ⚠ 0   Ln 9, Col 28   UTF-8   LF   TypeScript