Modules

John Papa
http://johnpapa.net
Twitter: @john_papa

pluralsight
hardcore developer training

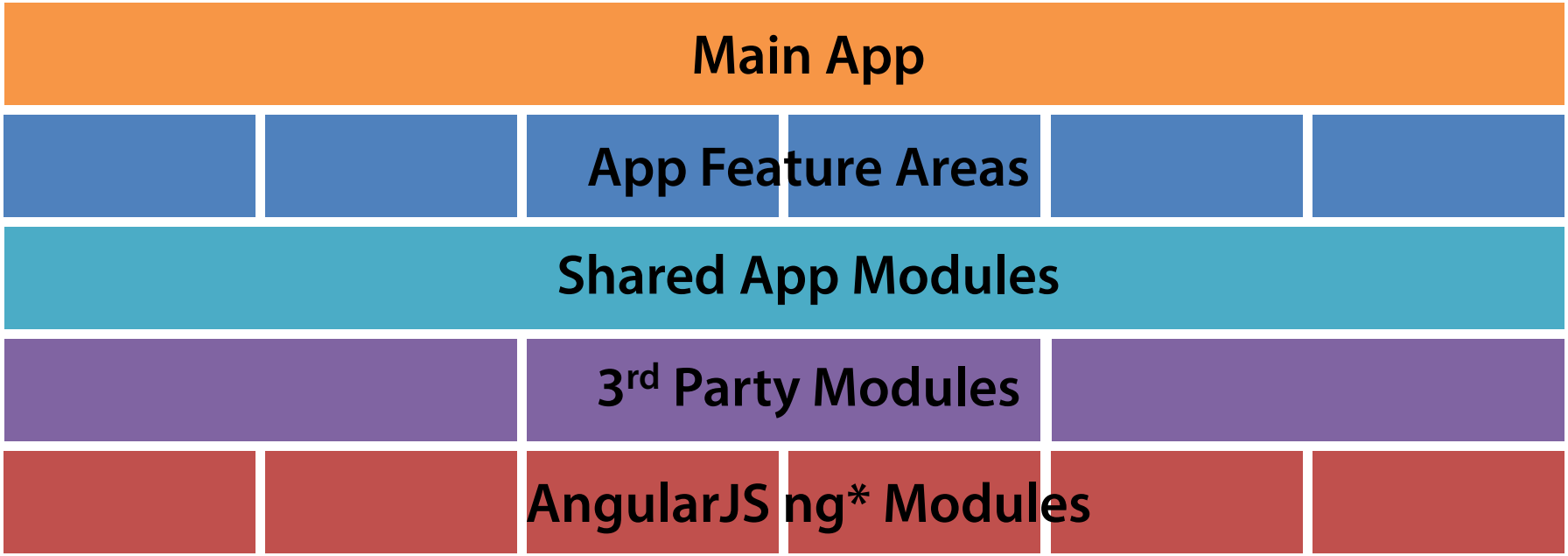
A Module is Another Form of Ravioli

One Monolithic App?

~~One Monolithic App?~~

Reusable Components to Assemble an App

We Build Apps Incrementally



Dependency Chains

Modularity

John Papa
http://johnpapa.net
Twitter: @john_papa

pluralsight
hardcore developer training

AngularJS Apps Depend on Modules

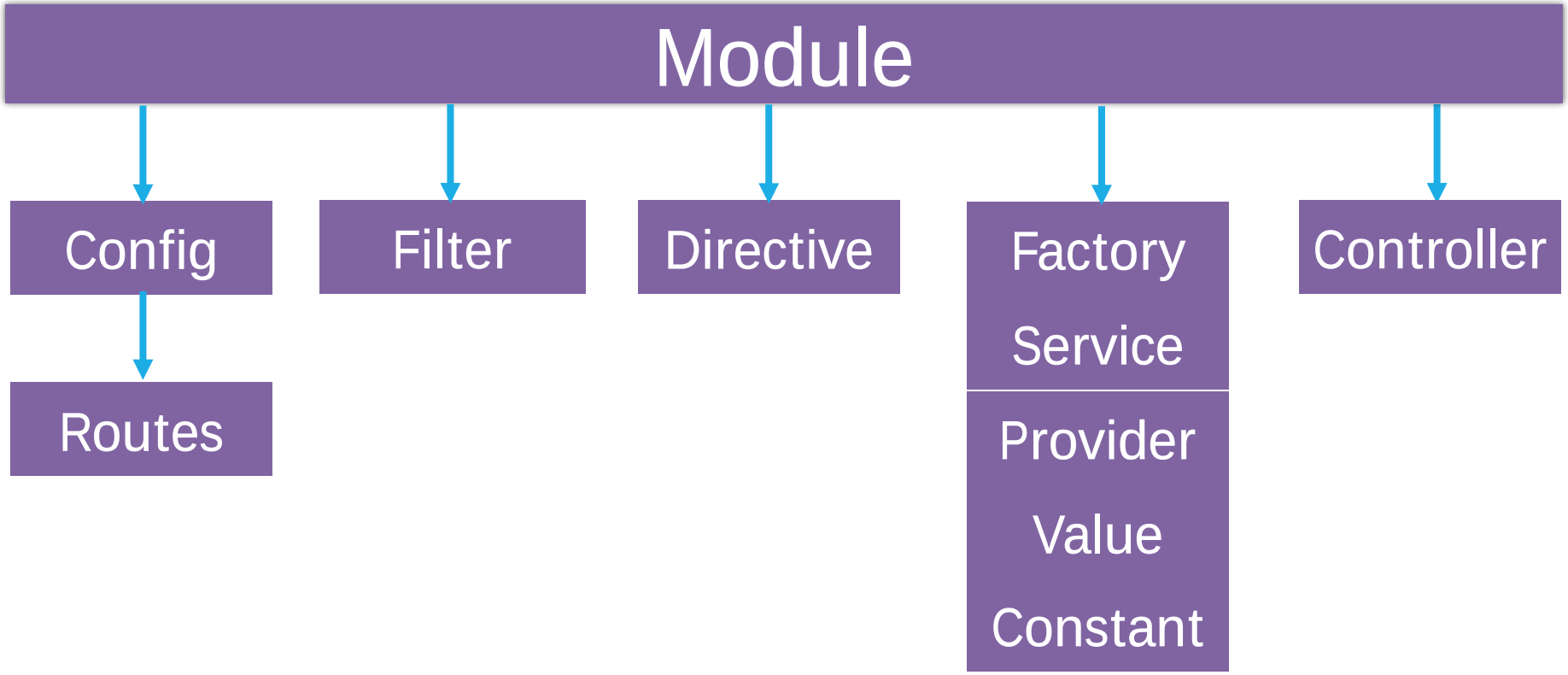
Each module can be an app, services, or widgets

Or all of these

Modules can depend on other modules

Promotes continuous development

Modules are Containers for AngularJS Components



Defining a Module

Created using angular.module()

```
// Setter
var app = angular.module('app', [ ]);
```

Optionally return a reference to the module

```
// Getter
var app = angular.module('app');
```

Modules Depend on Other Modules

```
// Define the module and declare
// its dependencies
angular.module('app', [
    'ngRoute',
    'app.avengers',
    'app.dashboard',
    'app.layout'
]);
```

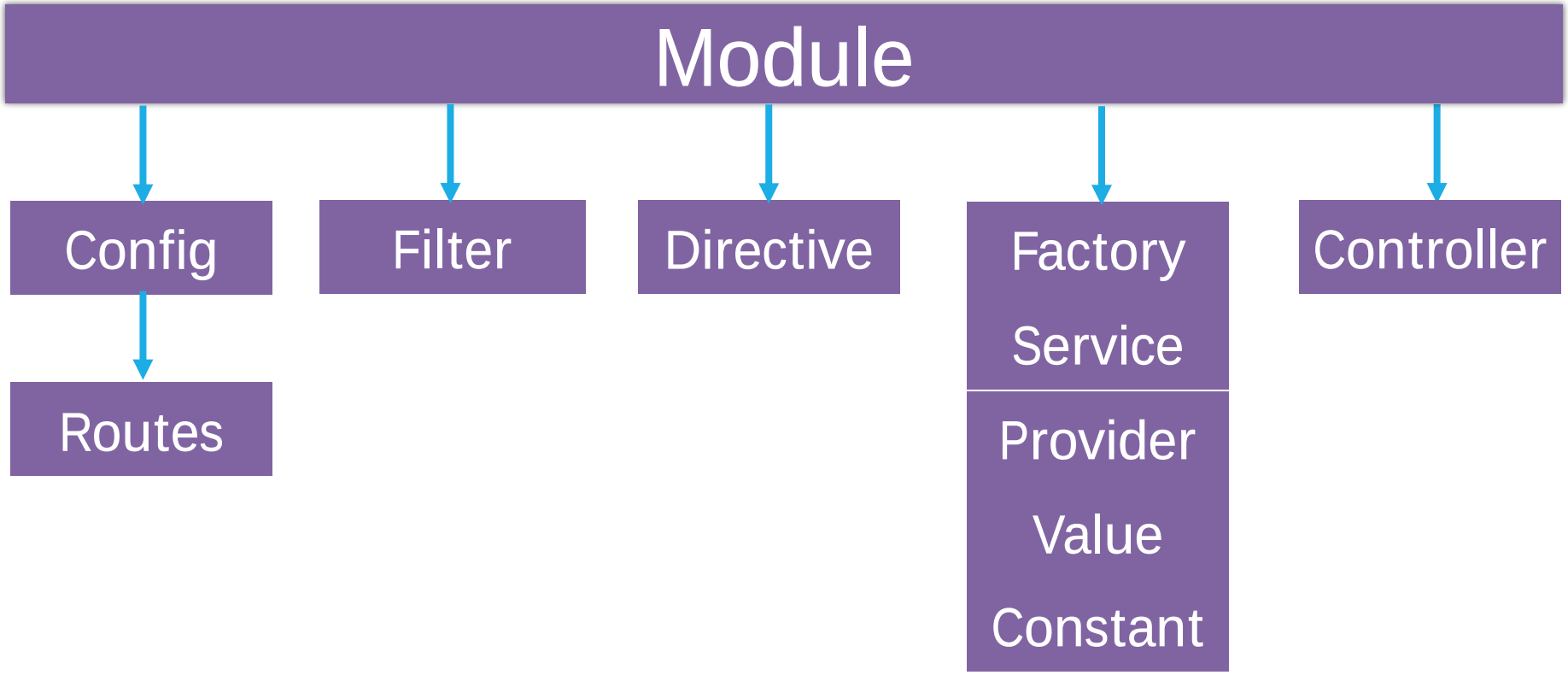
3 Categories of Modules

Examples of Module Categories

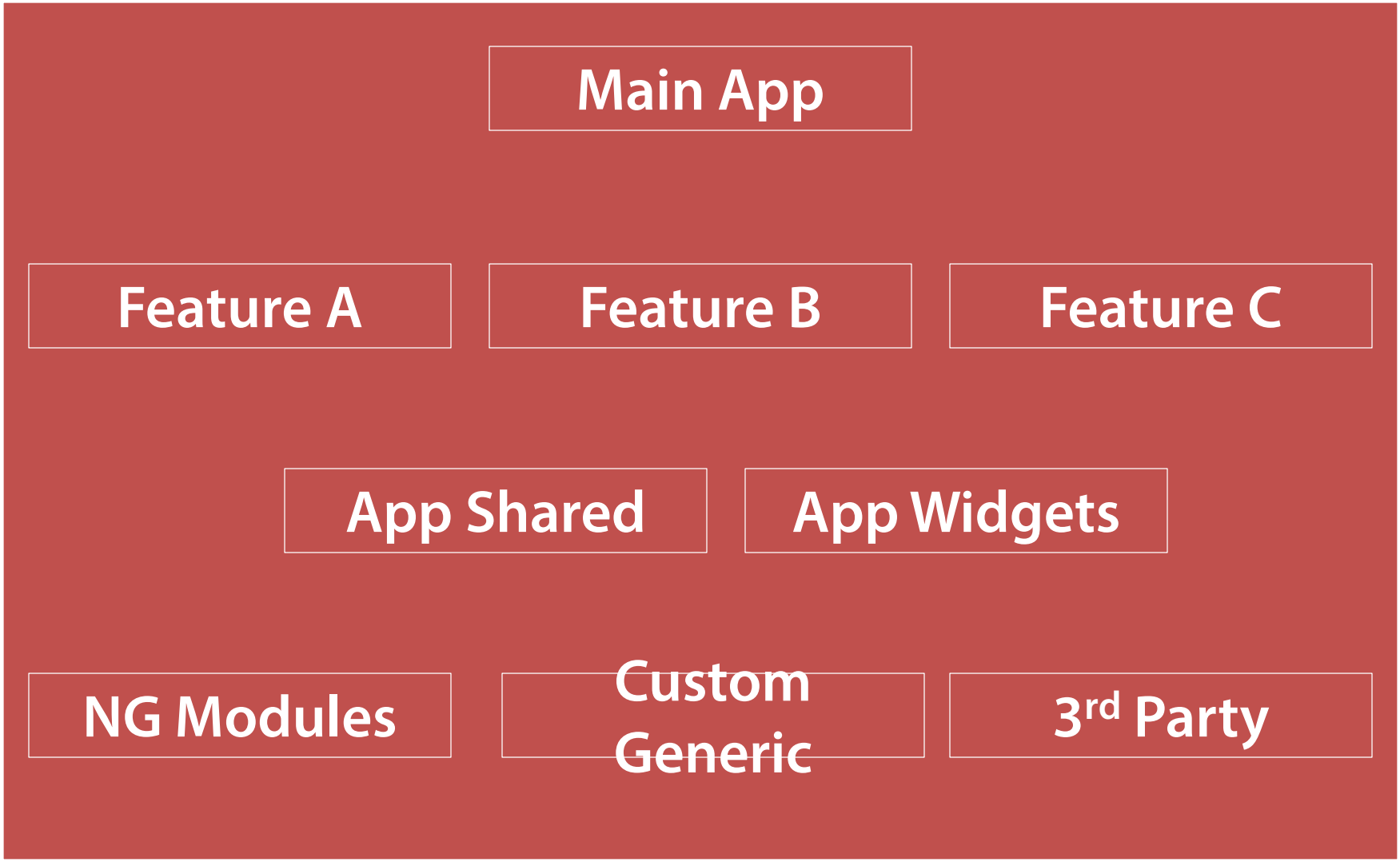
Modules may rely on functionality from other modules
Helper modules can be "injected" into a module:

```
angular.module('app', [
        // Angular modules
        'ngRoute',

        // Custom modules
        'app.dashboard',

        // 3rd Party Modules
        'ui.bootstrap',
        'breeze.angular'
    ]);
```

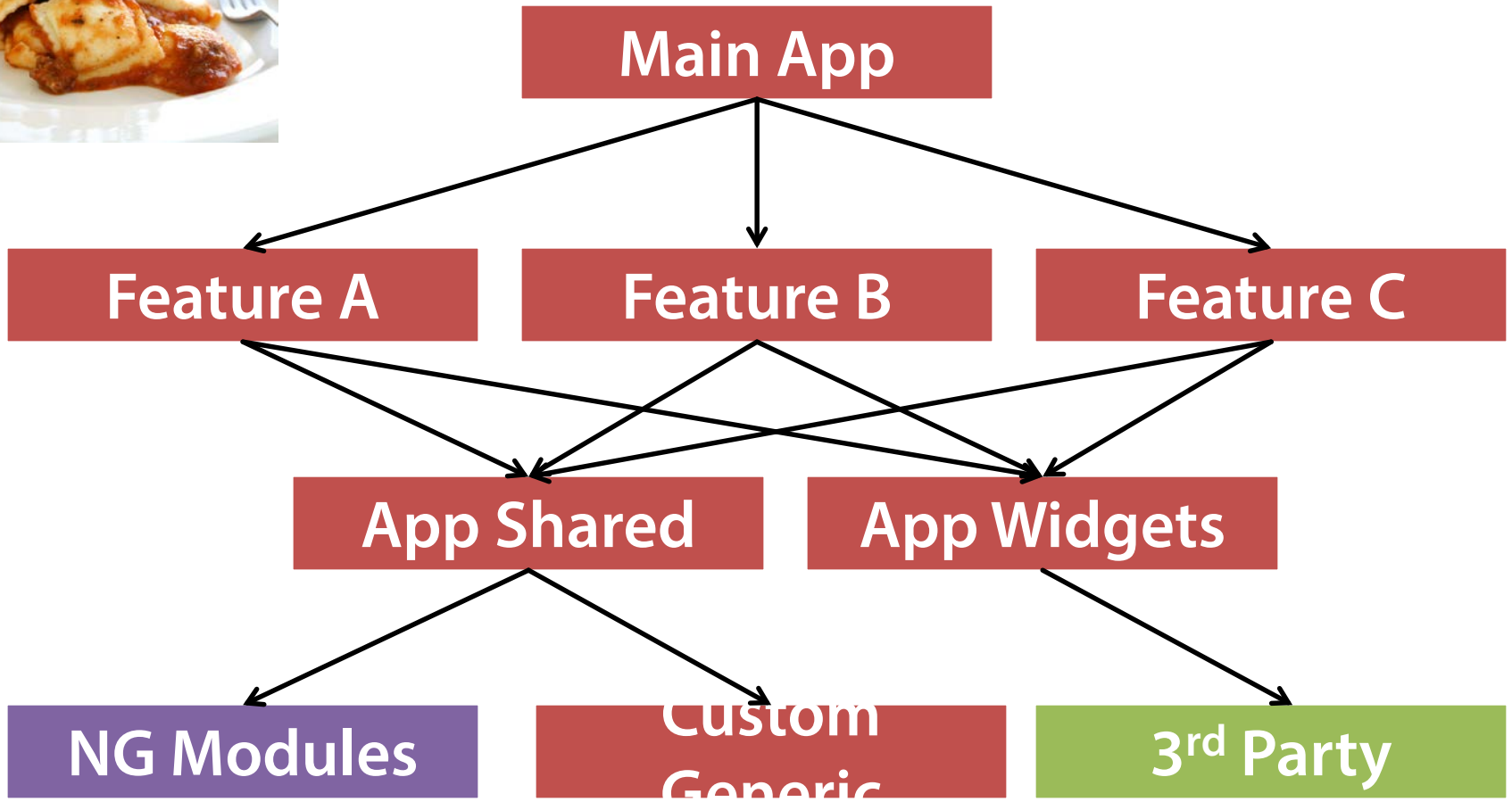
Many Components Combine to Make 1 Logical Module



All-In-One Module

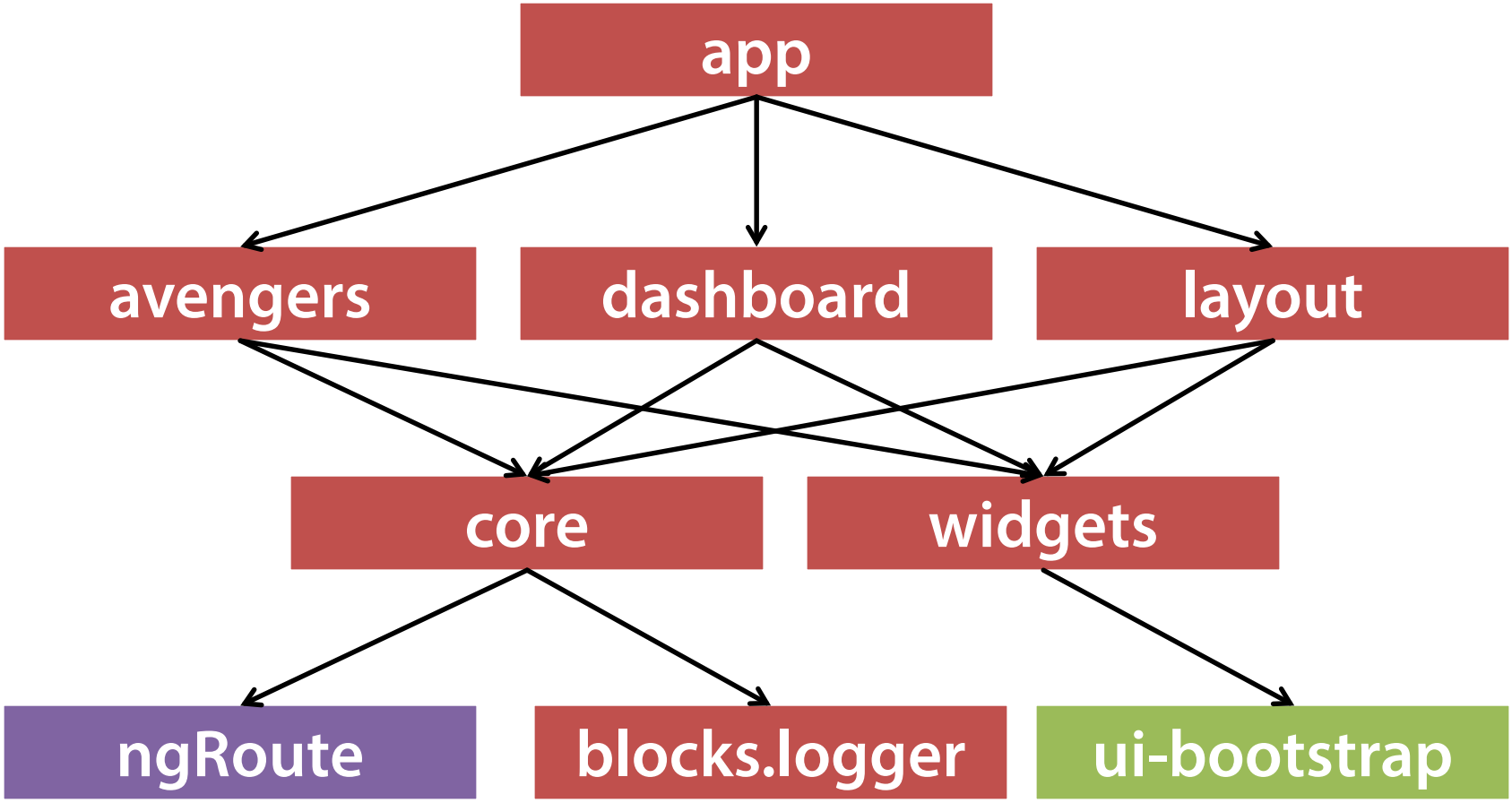


Ravioli Modules

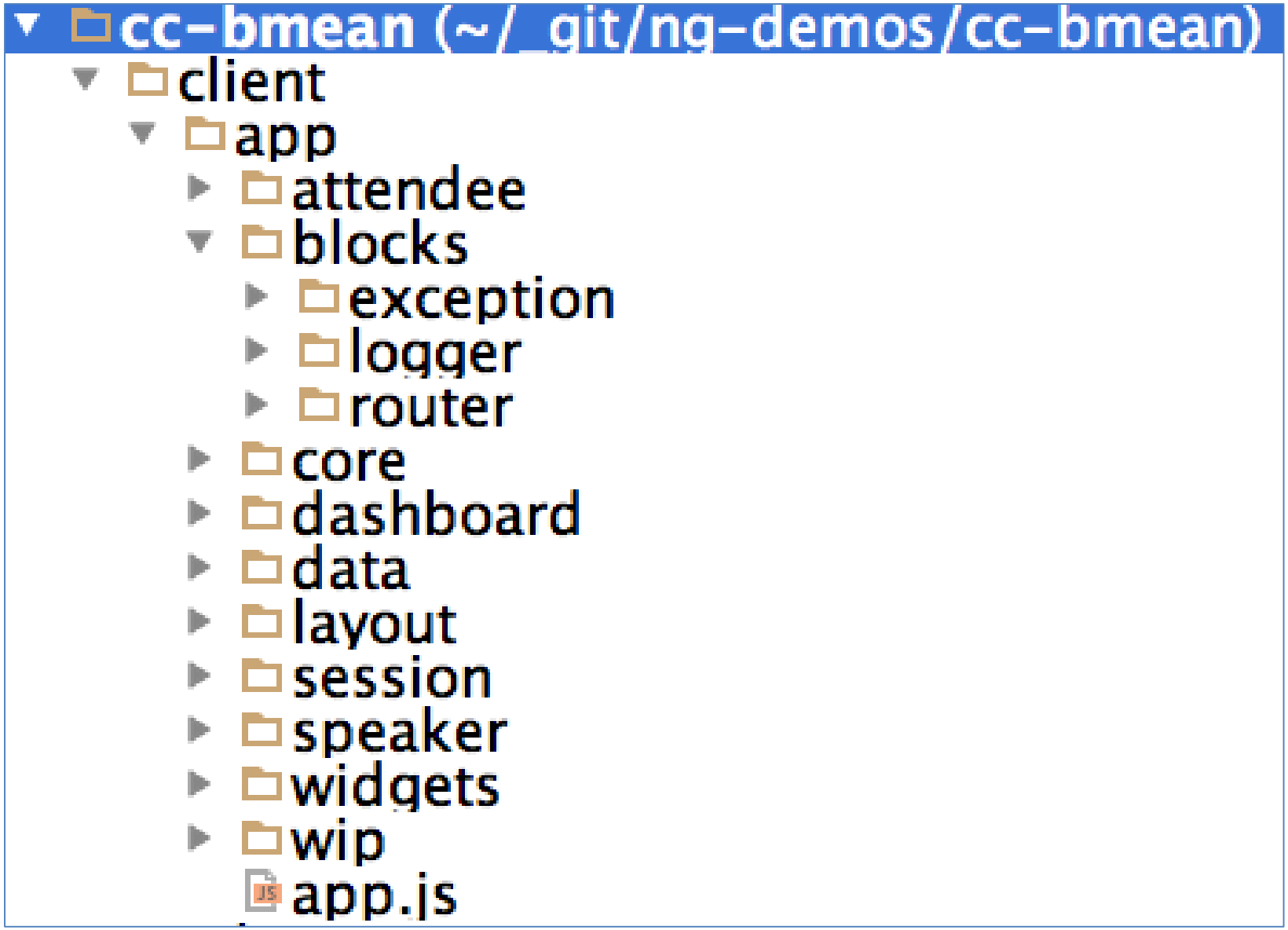


Dependency Chains

Examining a Dependency Chain



Modules by Folder



What Options Do We Have?

Dependency Strategies

Define dependencies for each module in each module

- Will always work
- Fairly common
- May be difficult to follow

Each module explicitly declares its dependencies

Define dependencies at specific levels

- Requires some forethought
- May be easier to maintain

Structuring Your Modules

Main App Module

```
angular.module('app', [
    'app.core',
    'app.widgets',

    /*
     * Feature areas
     */
    'app.avengers',
    'app.dashboard',
    'app.layout
]);
```

App Feature Modules

```
angular.module('app.avengers', []);
```

Cross App Modules

```
angular.module('app.core', [
    /*
     * Angular modules
     */
    'ngAnimate',
    'ngRoute',
    'ngSanitize',
    /*
     * Reusable cross app code modules
     */
    'blocks.exception',
    'blocks.logger',
    'blocks.router',
    /*
     * 3rd Party modules
     */
    'ngplus'
]);
```

Main App Module

Runs the entire app

Has little or no functionality

Aggregates feature area modules

```
angular.module('app', [
    'app.core',
    'app.widgets',

    /*
     * Feature areas
     */
    'app.avengers',
    'app.dashboard',
    'app.layout'
]);
```

Feature Modules

Encapsulates a set of features

```
angular.module('app.avengers', []);
```

Development team can work on these separately

Incrementally add or remove as needed

Can add dependencies here, or in Main App module

App Shareable Modules

Aggregates all AngularJS, 3rd party and app generic custom modules

Examples may be:

- Core
- Data
- Widgets

```
angular.module('app.core', [
    /*
     * Angular modules
     */
    'ngAnimate',
    'ngRoute',
    'ngSanitize',
    /*
     * Reusable cross app code modules
     */
    'blocks.exception',
    'blocks.logger',
    'blocks.router',
    /*
     * 3rd Party modules
     */
    'ngplus'
]);
```

Naming Tips

Name modules consistently

app is the main module

app.avengers is a feature module

Name files consistently

app.js (or app.module.js)

avengers.module.js

Summary

Modules are feature areas

Modules help keep features encapsulated and follows SoC

Don’t create a spider web of module dependency chains

Do aggregate in where it makes sense

app.js module and core.module.js

Good debugging skills can be helpful