AngularJS Factory, Provider, Service

From the AngularJS mailing list I got an amazing thread that explains service vs factory vs provider and their injection usage. Compiling the answers:

Services

Syntax: `module.service( 'serviceName', function );`
Result: When declaring serviceName as an injectable argument **you will be provided with an instance of the function. In other words** `new FunctionYouPassedToService()`.

Factories

Syntax: `module.factory( 'factoryName', function );`
Result: When declaring factoryName as an injectable argument you will be provided with **the value that is returned by invoking the function reference passed to module.factory**.

Providers

Syntax: `module.provider( 'providerName', function );`
Result: When declaring providerName as an injectable argument **you will be provided with** `ProviderFunction().$get()`. The constructor function is instantiated before the $get method is called - ProviderFunction is the function reference passed to module.provider.

Providers have the advantage that they can be configured during the module configuration phase.

See here for the provided code.

Here's a great further explanation by Misko:

```
provide.value('a',123);functionController(a){
  expect(a).toEqual(123);}
```

In this case the injector simply returns the value as is. But what if you want to compute the value? Then use a factory

```
provide.factory('b',function(a){return a*2;});functionController(b){
  expect(b).toEqual(246);}
```

So `factory` is a function which is responsible for creating the value. Notice that the factory function can ask for other dependencies.

But what if you want to be more OO and have a class called Greeter?

```
functionGreeter(a){this.greet =function(){return'Hello '+ a;}}
```

Then to instantiate you would have to write

```
provide.factory('greeter',function(a){returnnewGreeter(a);});
```

Then we could ask for 'greeter' in controller like this

```
functionController(greeter){
  expect(greeter instanceofGreeter).toBe(true);
  expect(greeter.greet()).toEqual('Hello 123');}
```

But that is way too wordy. A shorter way to write this would be `provider.service('greeter', Greeter);`

But what if we wanted to configure the `Greeter` class before the injection? Then we could write

```
provide.provider('greeter2',function(){var salutation ='Hello';this.setSalutation =function(s){
    salutation = s;}functionGreeter(a){this.greet =function(){return salutation +' '+ a;}}this.$get =function(a){returnn
ewGreeter(a);};});
```

We can then do this:

```
angular.module('abc',[]).config(function(greeter2Provider){
  greeter2Provider.setSalutation('Halo');});functionController(greeter2){
  expect(greeter2.greet()).toEqual('Halo 123');}
```

As a side note, `service`, `factory`, and `value` are all derived from provider.

```
provider.service =function(name,Class){
  provider.provide(name,function(){this.$get =function($injector){return $injector.instantiate(Class);};});}

provider.factory =function(name, factory){
  provider.provide(name,function(){this.$get =function($injector){return $injector.invoke(factory);};});}

provider.value =function(name, value){
  provider.factory(name,function(){return value;});};
```