```
var injector = new Injector(function(export) {
  export.listPromiseFactory = function($q) {
    return $q.defer(function(defer) {
      defer.resolve([]);
    });
  };

  export.MyController = function(list) {
  };

  export.Foo = function(list, curry_a, curry_b) {};
});

 // fails since we need promise
injector.get('myController');

// works returns promise
injector.get('myControllerPromise').then(...);

// You can ask for promise
injector.get('listPromise').then(...);

// You can ask for factory
injector.get('listPromiseFactory')();

// You can ask for Type
injector.get('MyControllerFactory');
```

Rules:

- `instance`: constant
- `instanceFactory`: factory which creates constant
- `Instance`: class which when instantiated will create an instance
- `instancePromise`: A promise which will resolve to instance
- `instancePromiseFactory`: A factory which produces a promise which will resolve to instance.
- `InstanceFactory`: a curyable factory which produces instances.
- `InstanceFactoryPromise`: a curyable factory which produces promise instances.

Built in Instances:

- `$injector`:
- `$tickQueue`:
- `$q`: promise API