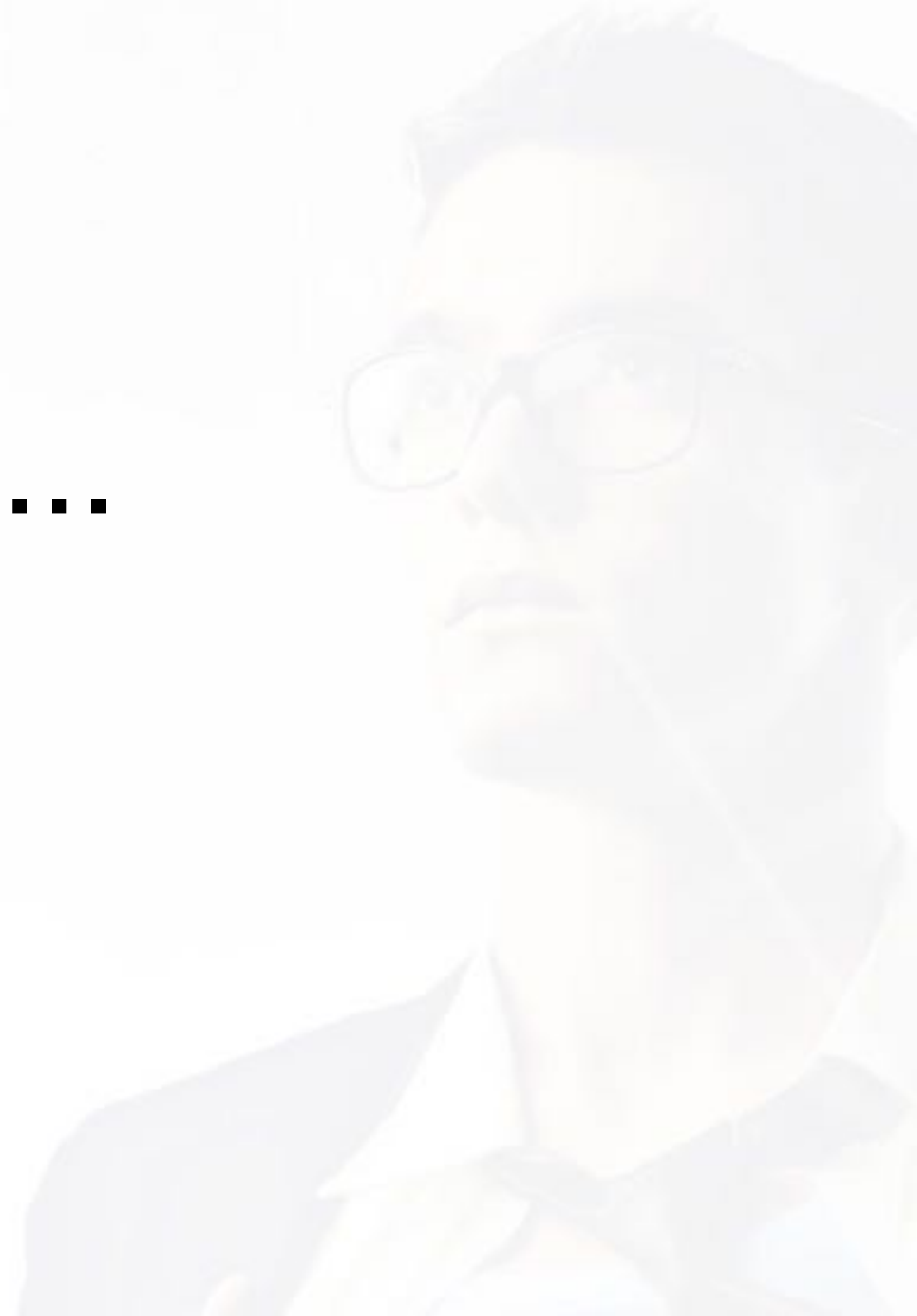
AngularJS Basics (TypeScript)

from: **Saban Ünlü**

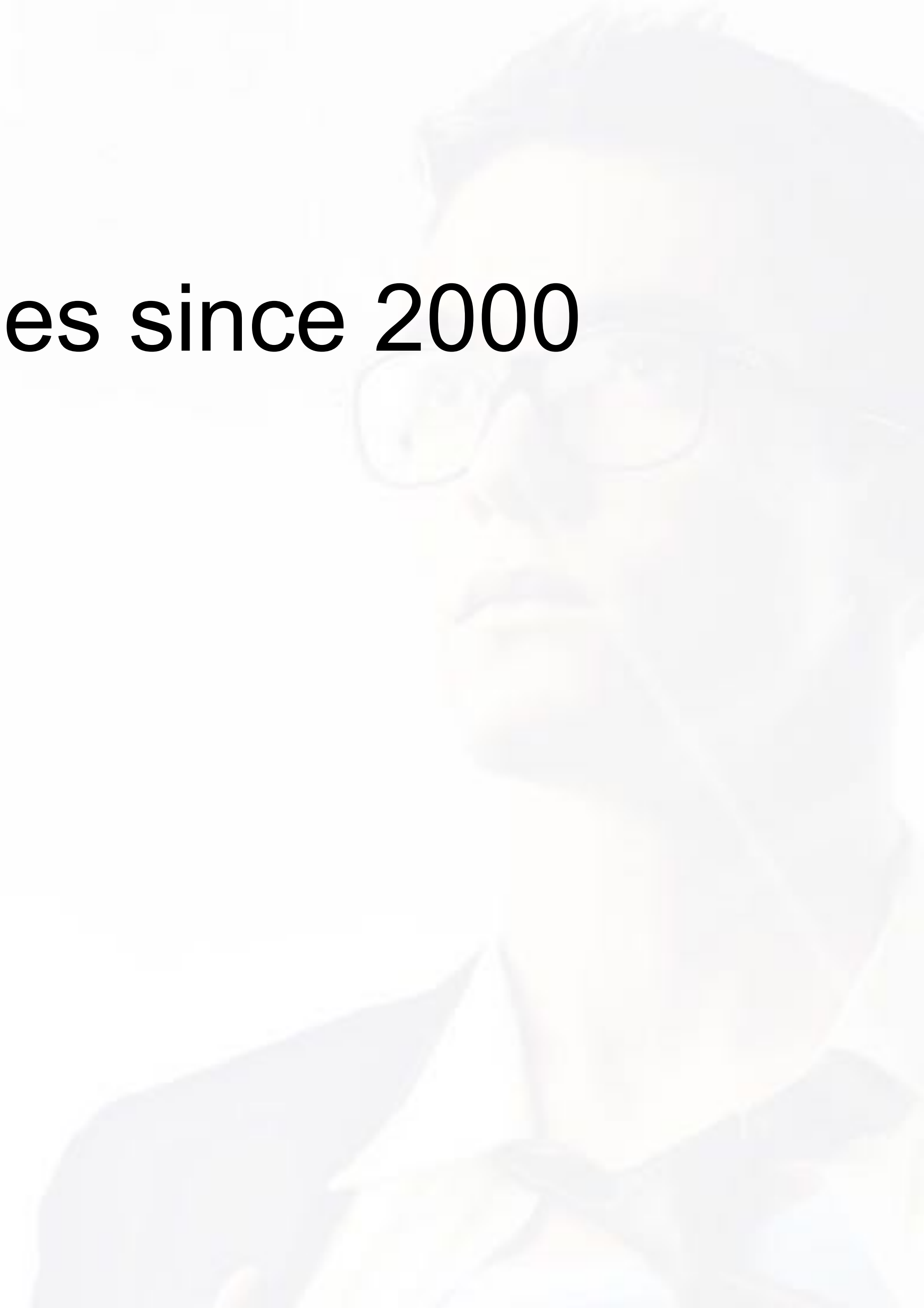
for: **msg**



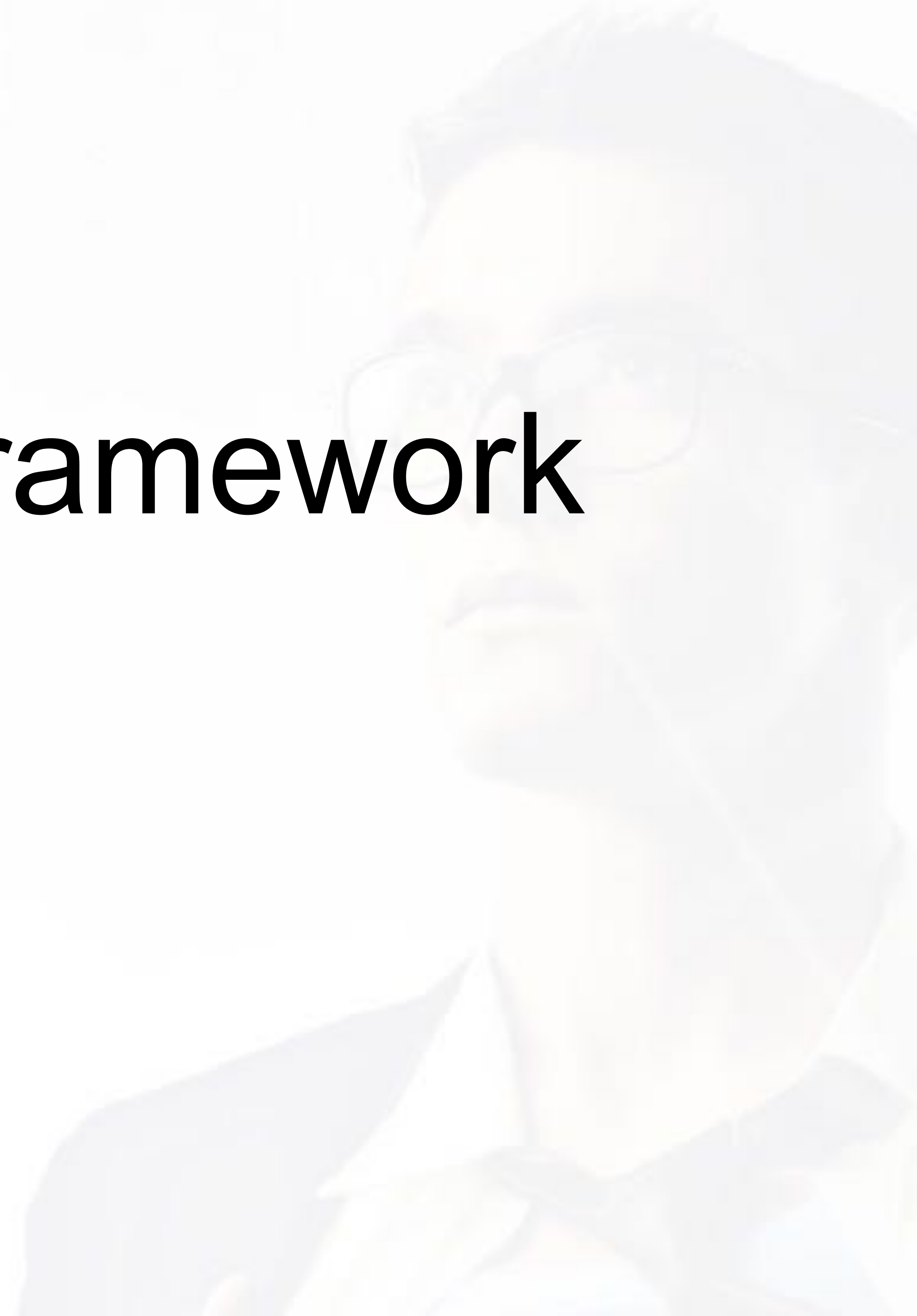
Two words about me...

Saban Ünlü

- Software architect and programmer
- Consultant and Trainer for web technologies since 2000
- Author
- Adobe Influencer
- Founder of netTrek



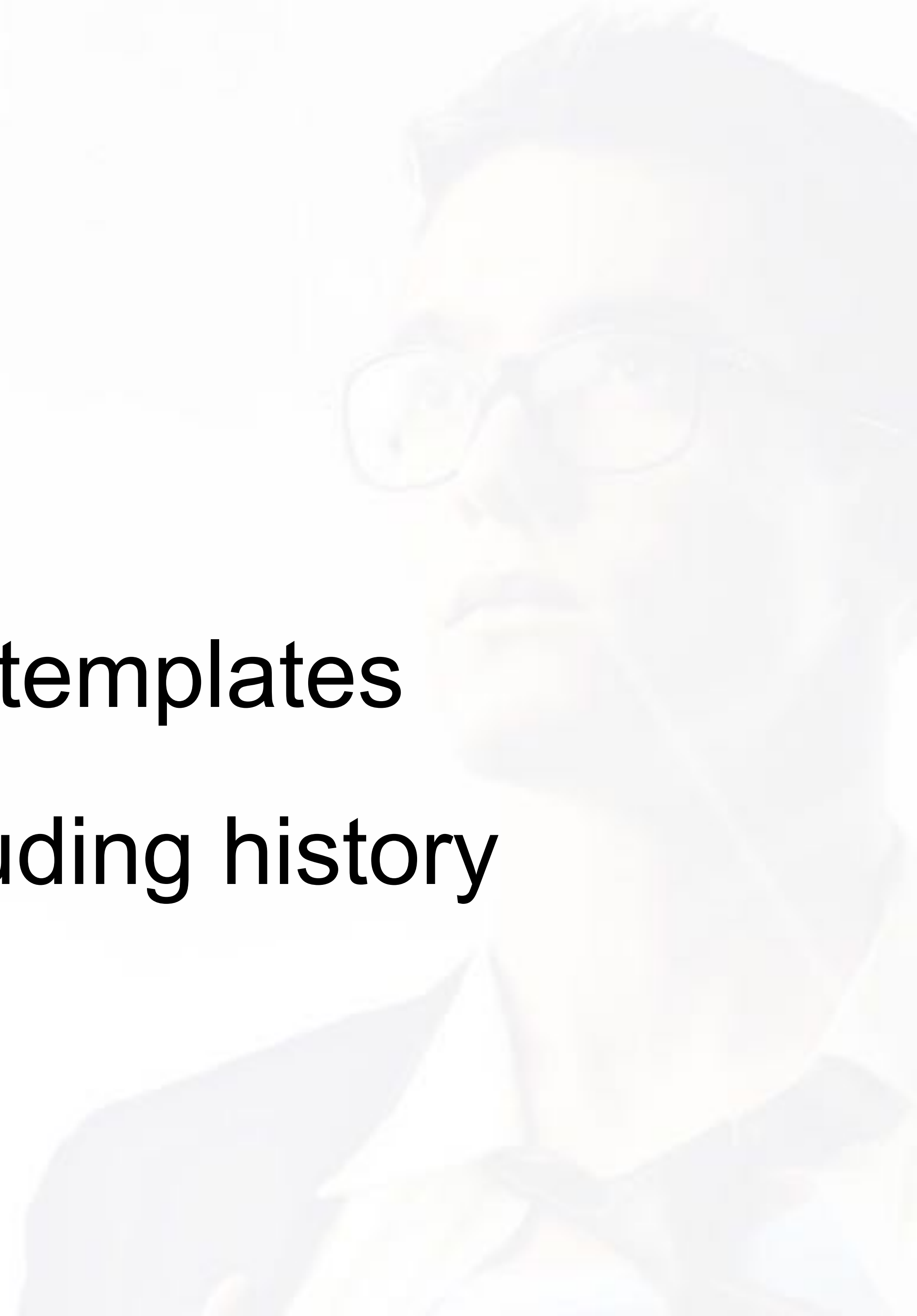
What is AngularJS?



A Single Page Application Framework

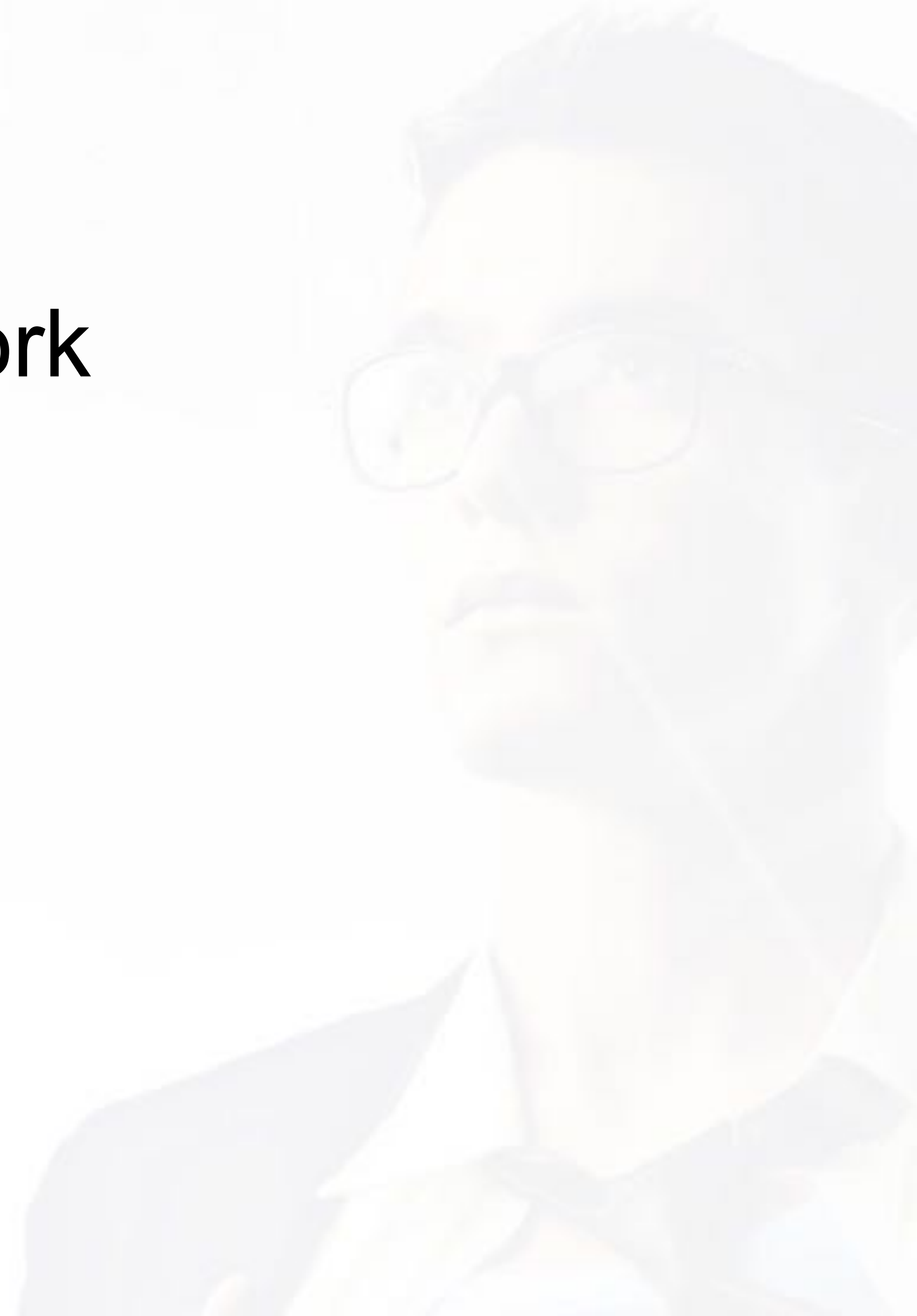
What is a single page App?

- Web app that consists of an HTML page
- The page will never be left
- Data- or user-driven content is exchanged
- Based on AJAX, JSON, REST and HTML templates
- Routings define what is shown when, including history back



Single page Apps with AngularJS

- Template based
- Enhanced HTML for component-based work
- Data Binding and Dependency Injection
- Independent of backend technology
- And all with just a single framework

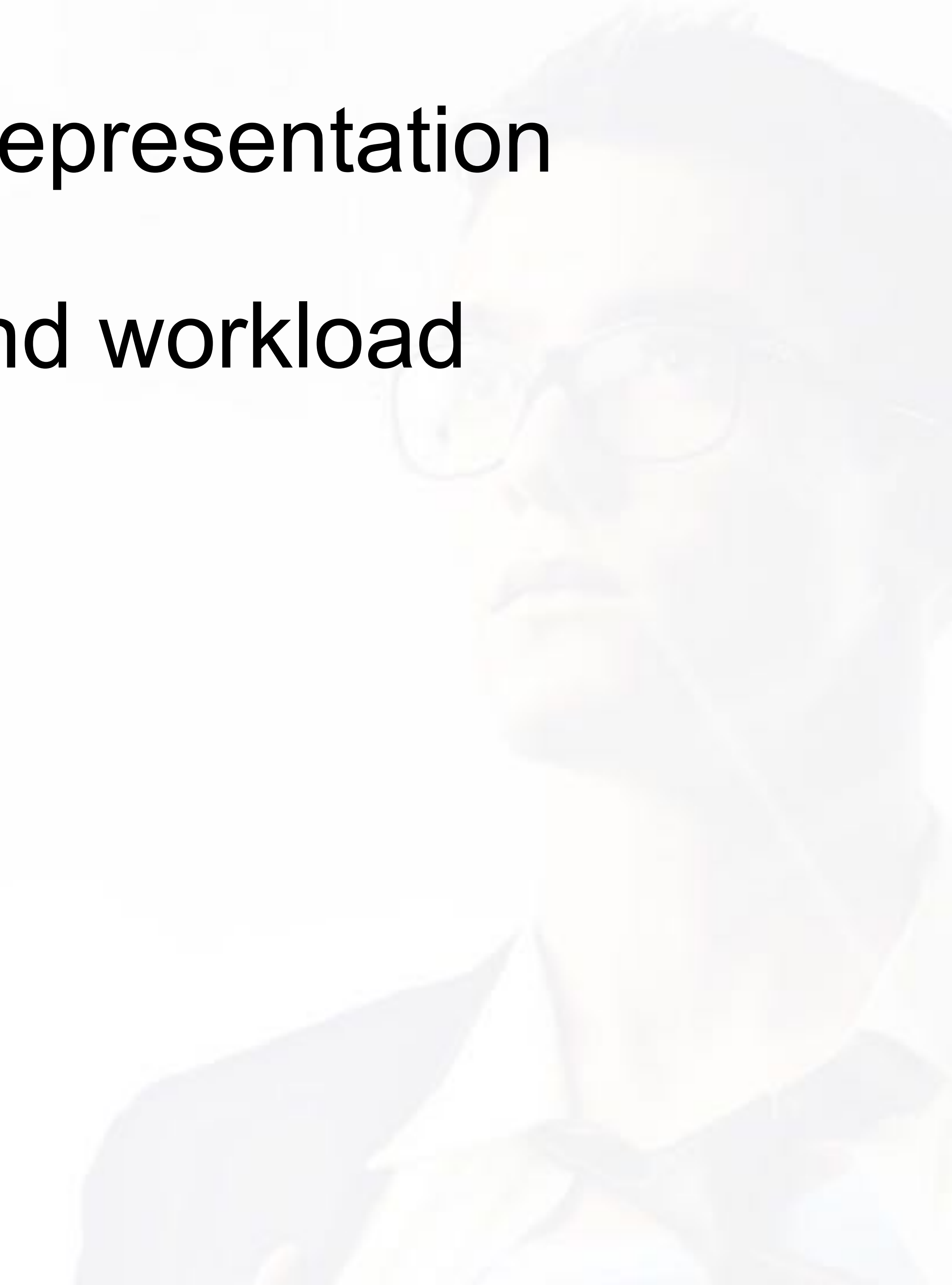


Core feature

What AngularJS can do

- Simple binding of model data in a view
- Inject or repeat DOM fragments
- Define Controller for DOM Elements
- Grouping DOM fragments as components
- Processing and validating forms

What AngularJS can do



- Isolation of the app logic from the DOM representation
- Fat client development to relieve back-end workload
- model view whatever
- Unit and End-2-End Testing

AngularJS key words

AngularJS key words

- **Template**

  HTML-Frakments as presentation template

- **Directive**

  Extends HTML with new attributes

- **Componenten**

  Extends HTML with new nodes

- **Model**

  Data made accessible to the user

AngularJS key words

- **Scope**

  Context in which the model is stored and made available for controllers, directives and expressions

- **Expressions**

  Terms that provide access to scope functions and variables

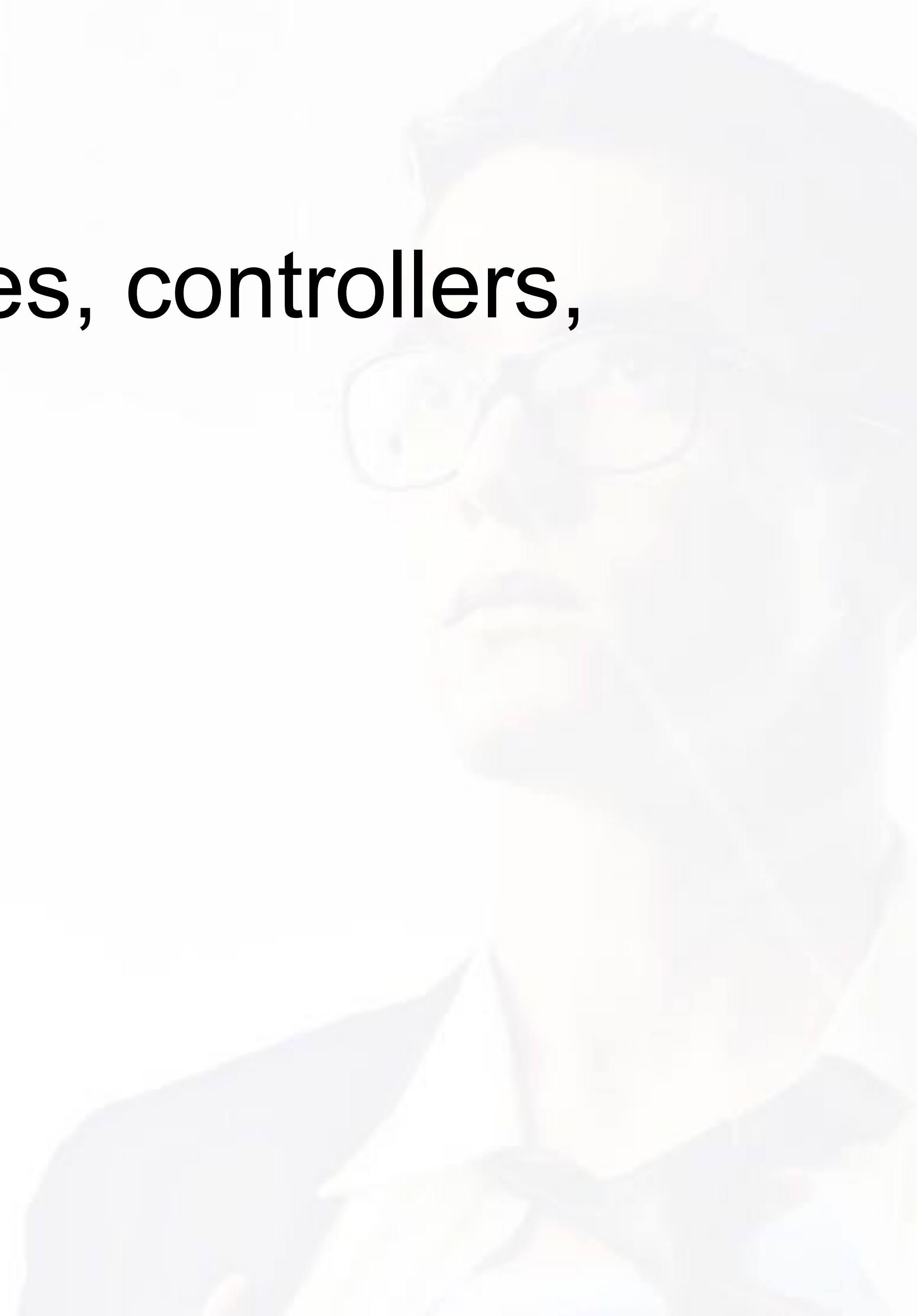
AngularJS key words

- **Compiler**

  Parst templates and instantiates directives, controllers, scope and expressions

- **Filter**

  Formats the output of a value



AngularJS key words

- **View**

  User interface

- **Data Binding**

  Synchronizes model data with the view

- **Controller**

  Business-Logik of a view

AngularJS key words

- **Dependency Injection**

  Dependent object instantiation and assignment

- **Injector**

  Container provides instances for dependency injections

AngularJS key words

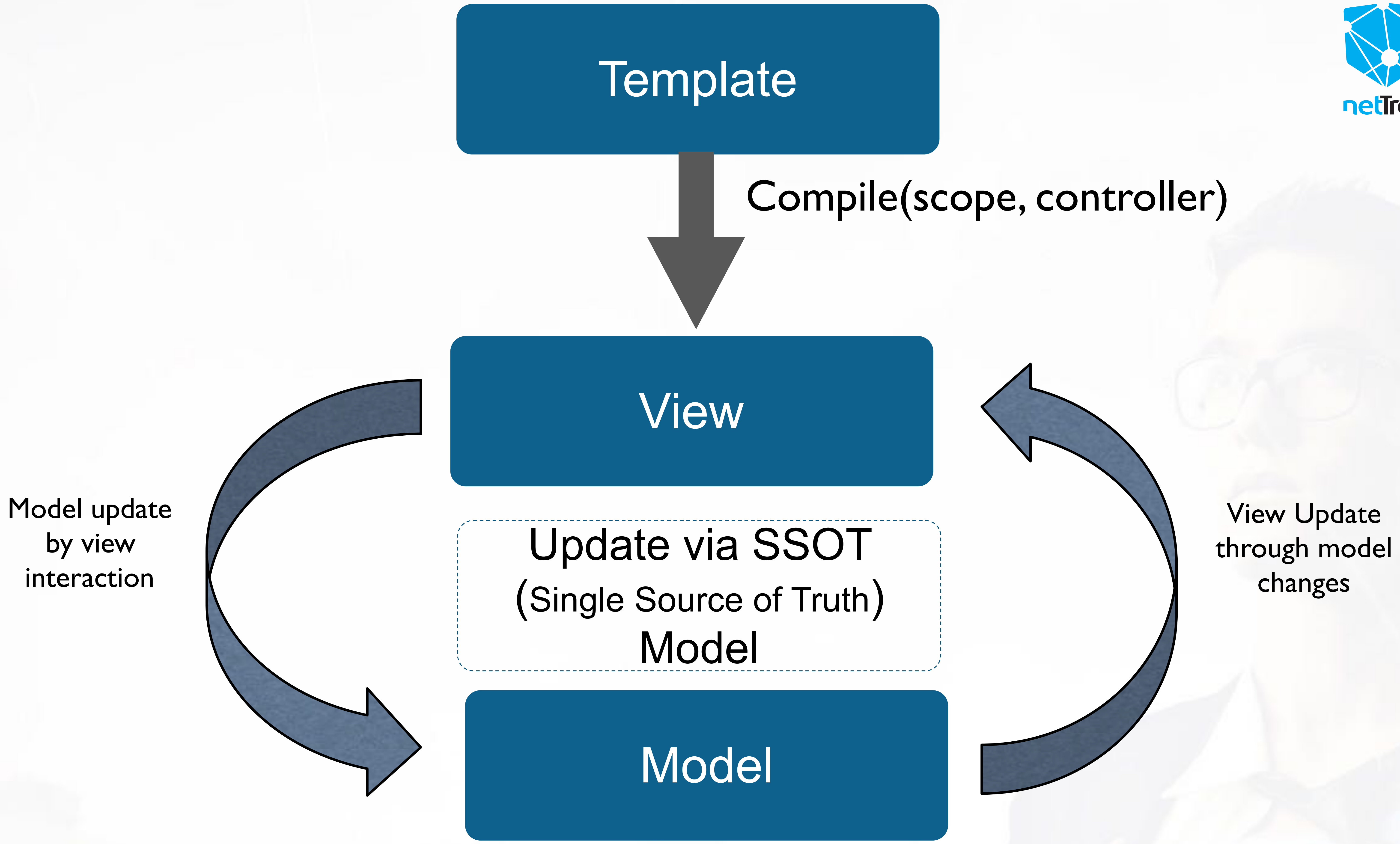
- **Modul**

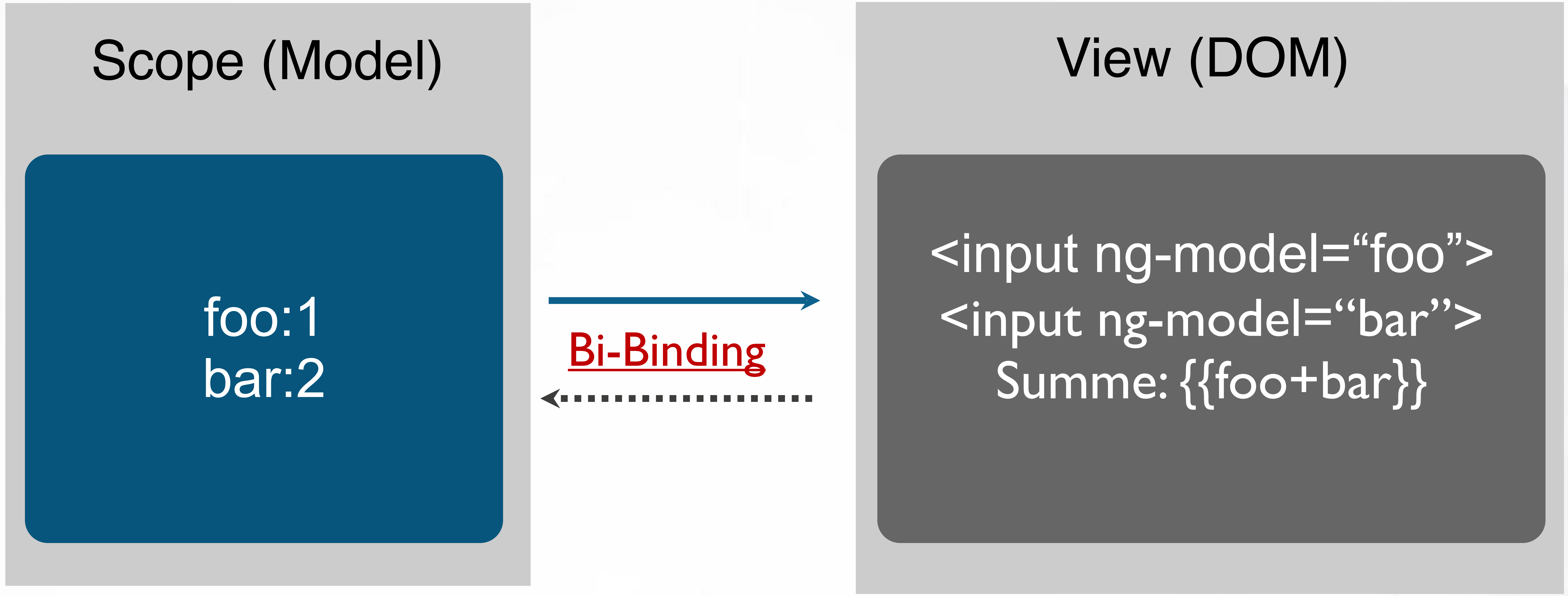
  Containers for application components, e. g. controllers, services, filters, directives

- **Services**

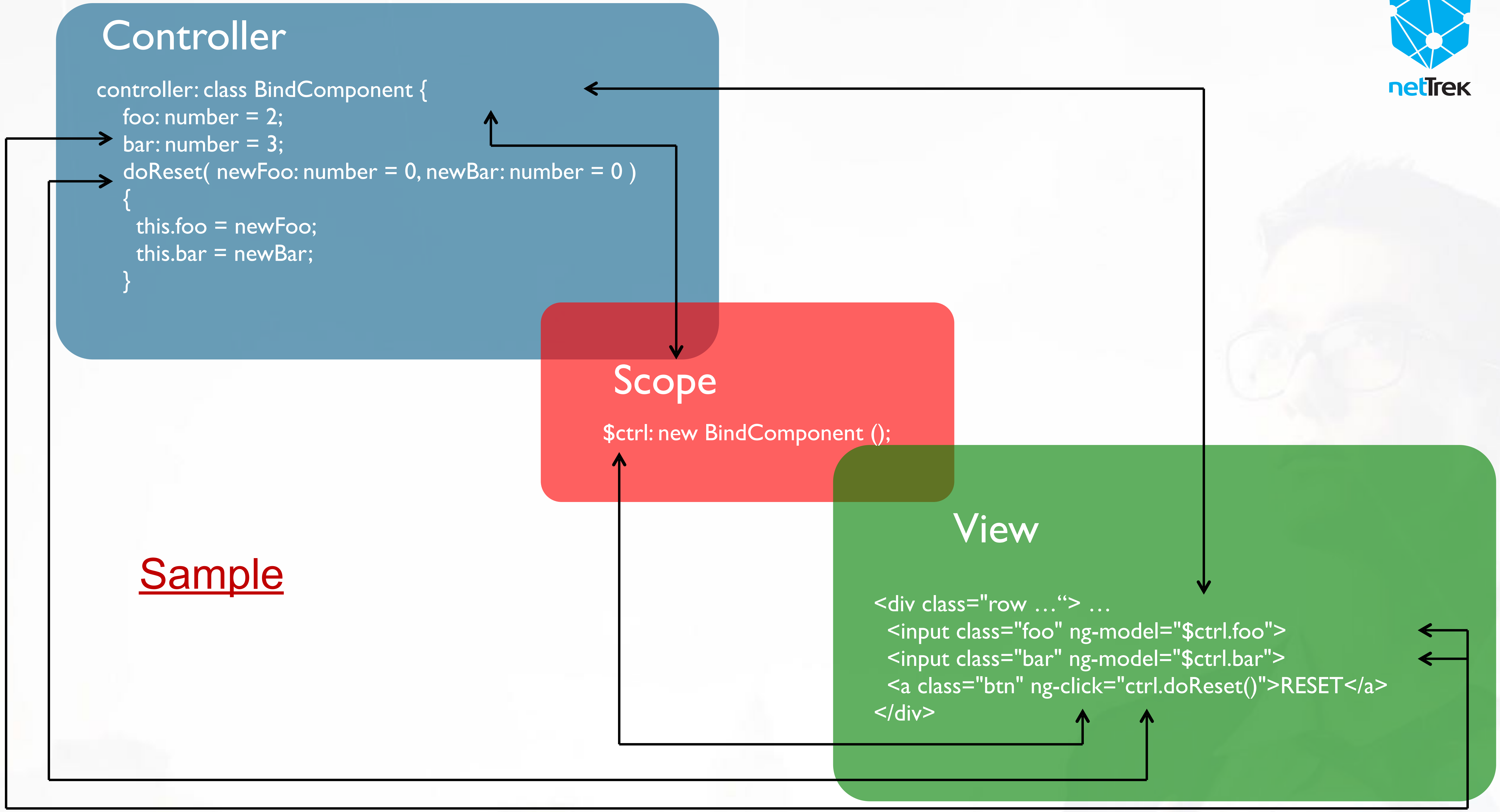
  Reusable business logic that is independent of a view

AngularJS Concepts


Template
Compile(scope, controller)
View
Model update by view interaction
Update via SSOT (Single Source of Truth) Model
View Update through model changes
Model



```
<input ng-model="foo">
<input ng-model="bar">
Summe: {{foo+bar}}
```



Controller

```
controller: class BindComponent {
   foo: number = 2;
   bar: number = 3;
   doReset( newFoo: number = 0, newBar: number = 0 )
   {
     this.foo = newFoo;
     this.bar = newBar;
   }
```

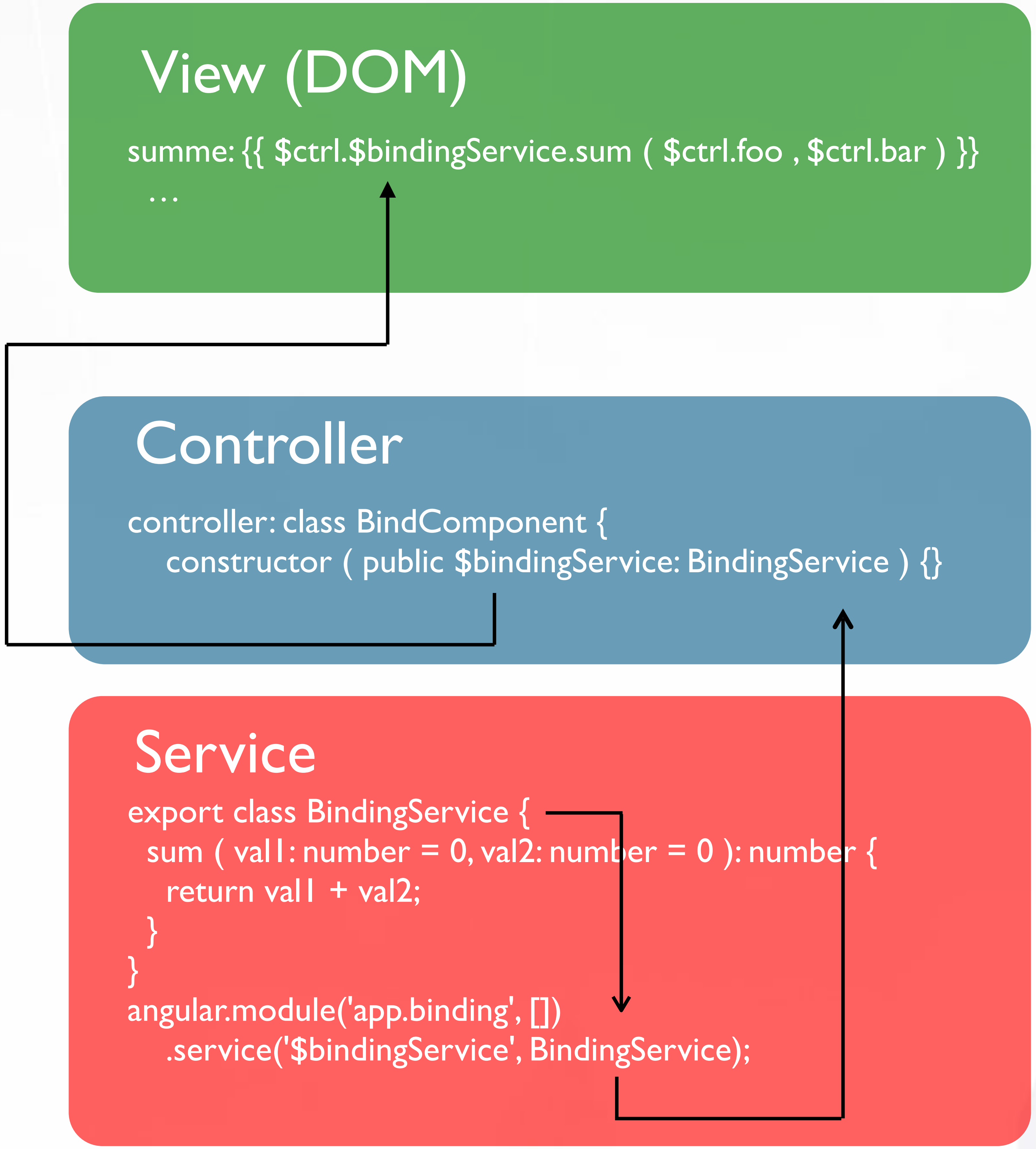
Scope

```
$ctrl: new BindComponent ();
```

View

```
<div class="row …“> …
  <input class="foo" ng-model="$ctrl.foo">
  <input class="bar" ng-model="$ctrl.bar">
  <a class="btn" ng-click="ctrl.doReset()">RESET</a>
</div>
```

Sample



```
summe: {{ $ctrl.$bindingService.sum ( $ctrl.foo , $ctrl.bar ) }}
 …
```

```
controller: class BindComponent {
    constructor ( public $bindingService: BindingService ) {}
```

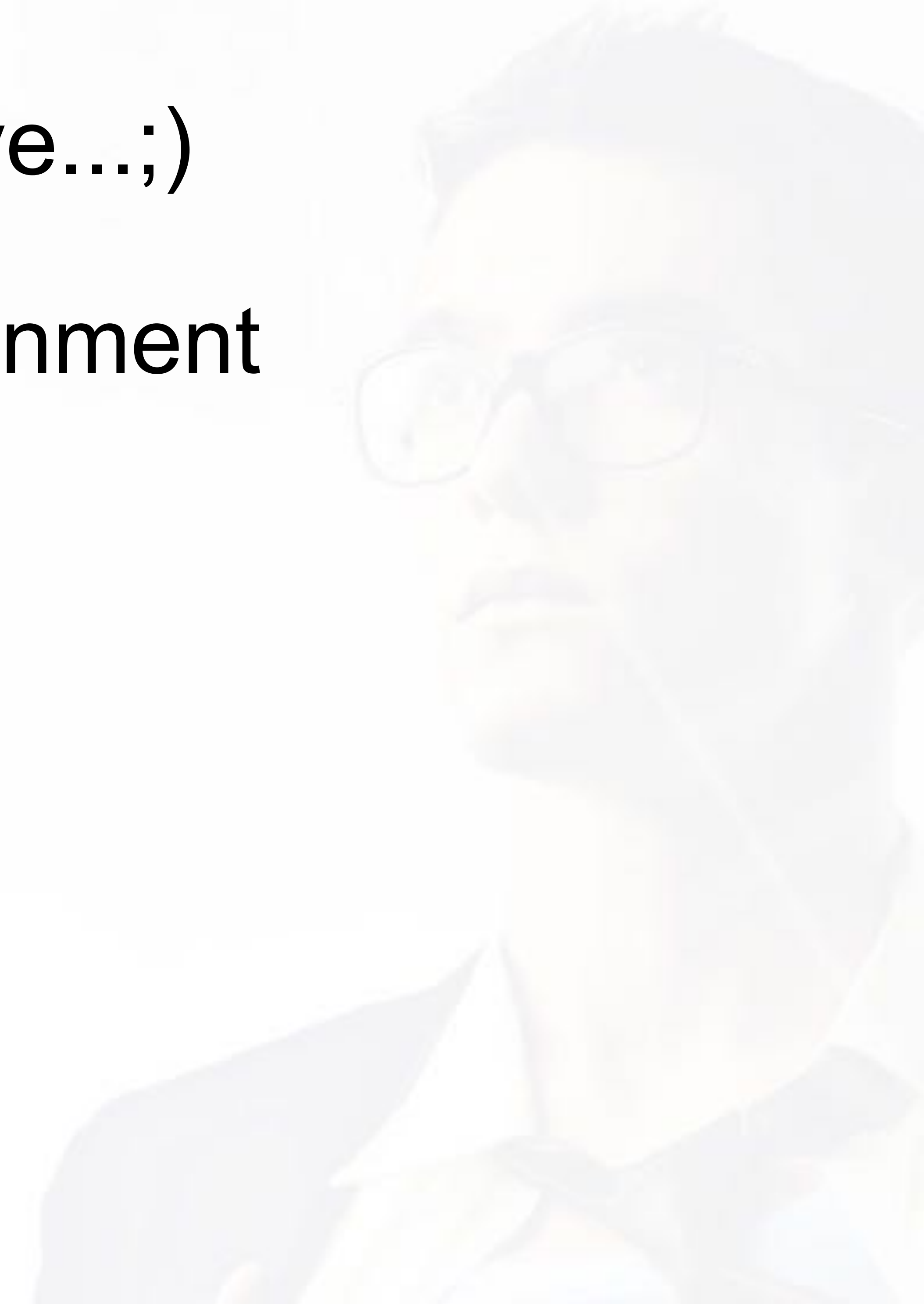
```
export class BindingService {
  sum ( val1: number = 0, val2: number = 0 ): number {
    return val1 + val2;
  }
}
angular.module('app.binding', [])
    .service('$bindingService', BindingService);
```

Sample

Project setup

Project setup methods

- Manual - via download and with lots of love...;)
- Angular-seed - Git clone setup and environment installation via NodeJS
- msg Setup



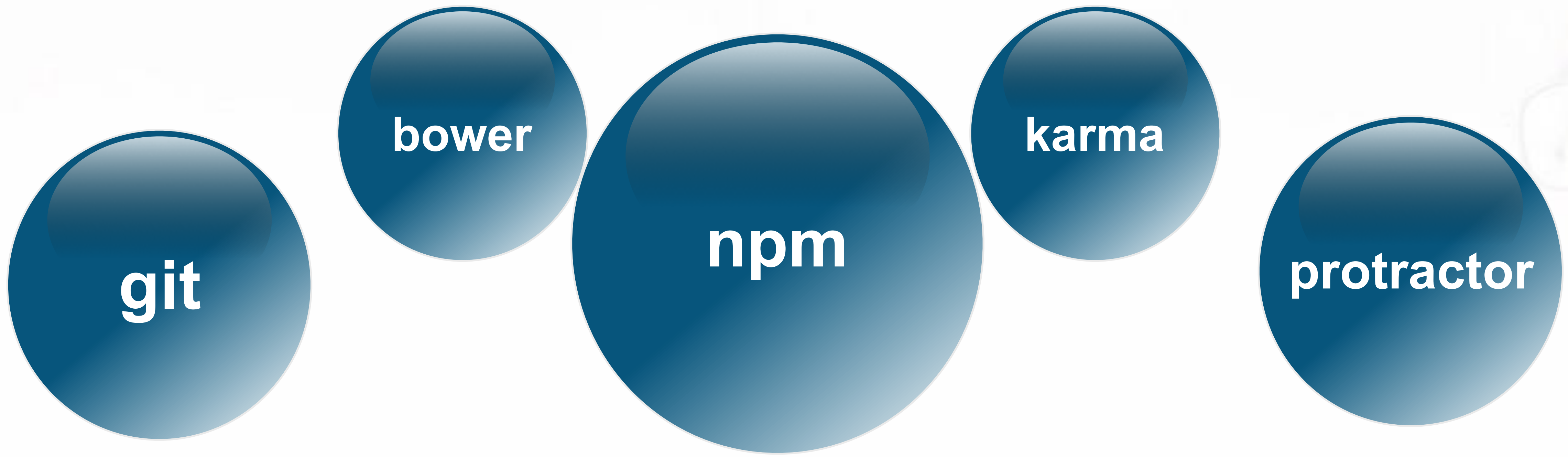
Setup manual

- Download angularJS
- Create HTML-Site
- Load angularJS
- add ng-app directive

Projektaufbau

```
├── image
│   └── myImage.png
├── index.html
├── scripts
│   ├── app.js
│   └── users
│       ├── _users.scss
│       ├── user-input-directive.js
│       ├── user-input-directive_test.js
│       ├── user-input.html
│       ├── users-controller.js
│       ├── users-controller_test.js
│       ├── users-module.js
│       ├── users-service.js
│       ├── users-service_test.js
│       └── users.html
└── styles
    └── main.scss
```

Angular-seed Setup



Technology

- JavaScript runtime environment for various OS
- Versioning system for software (GitHub - Filehoster)
- Package manager for JavaScript
- Karma - Test runner for Unit-Tests
- Protractor - Framework for E2E Tests

Env-Installtions for Angular-seed (Win)

- NodeJS and GIT must be installed first
- http://nodejs.org/download/
- http://git-scm.com

Env-Installtions for Angular-seed (MAc)

- NodeJS and GIT (via XCODE) must be installed first
- XCODE (4 Git and more)
    - XCODE - install and start oder install and:
        - xcode-select --install
- http://nodejs.org/download/

Angular-seed Setup

- git clone https://github.com/angular/angular-seed.git
- npm install
- npm start
- npm test
- npm run update-webdriver && npm run protractor

msg setup

- npm
- bootstrap
- git
- SystemJS
- karma
- gulp
- TypeScript

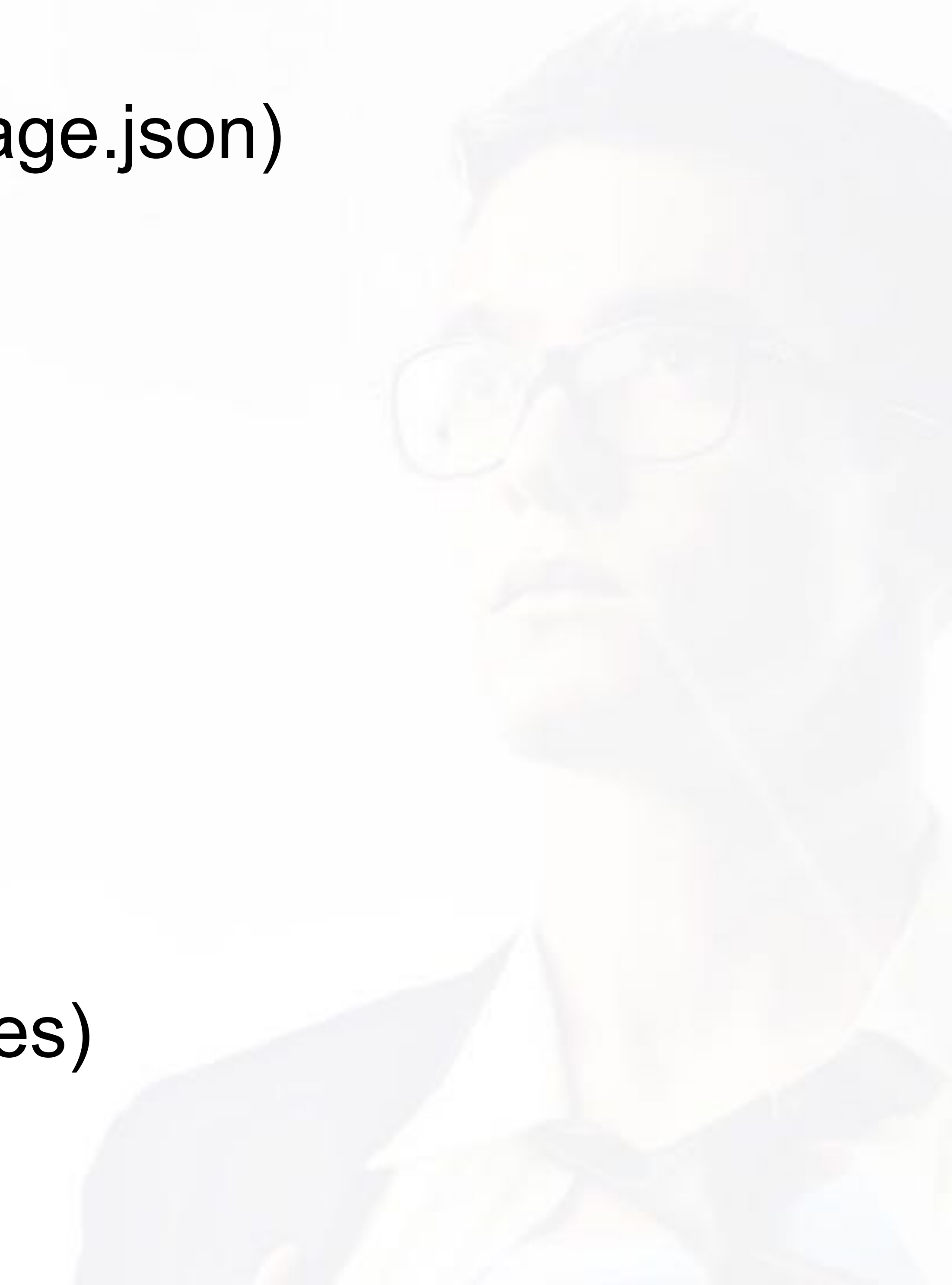
Technologie

SystemJS

- Dynamic ES module loader
- CSS library for responsive websites
- ES2015-based language compilable to ES3 +
- gulp is a toolkit for tasks in your development workflow

msg Setup light! - enviroment

- **npm init** - create project under npm control (package.json)
- **npm i typescript —save-dev**
    - install TypeScript
- **npm i lite-server —save-dev**
    - install Server
- **npm i @types/angular —save-dev**
    - install Type-Definitions (C - Header Files)



msg Setup light! - vendor

- npm i angular --save
- npm i bootstrap --save
- npm i jquery --save
- npm i systemjs —save

msg Setup light! - configure TypeScript

- **tsc init** - creates tsconfig-file
- "target": "es5"
- "module": "commonjs",
- "typeRoots": ["node_modules/@types"],
- "exclude": ["node_modules"]

msg Setup light! - configure SystemJS

```
packages: {
    'app': {
        main: './main.js',
        defaultExtension: 'js'
    }
},

map: {
    app: './',
    'angular': 'node_modules/angular/angular.js'
},
meta: {
    'angular': {
        format: 'global',
        exports: 'angular'
    }
}
```

Module

Angular Module

- **Container**
- Controller, Services, Directives & Filter
- **Setup**
- configuration
- initialisation

Benefits

- structuring
- Black Box
- teamwork
- reusability
- maintainability

Angular Module

- **Create**
- angular.module: IModule
    - Modulename: string
    - opt. Dependencies: string[]
- Getter (without dependencies list)
- (Beispiel 001)

Angular Module

Application module

- Initialization Module

- **automatic bootstrapping**
  - ng-app (001)
- **Run- and Config Method**
  - run - add callback, that runs when all dep. Modules initialized (002)
  - config - add callback, that runs in provider phase (002)

Angular Module

man. Bootstrapping

- angular.bootstrap (003)
  - element: string|Element|JQuery|Document
  - modules: string[]

Angular Module

- **Provide values in module**
  - value<string> -> IModule
    - name: string
    - value: T

  - Dependency Injection - (004)

Angular Module

  - constant - <string> -> IModule
    - name: string
    - value: T (005)
- **Provider-Objekt**
  - $provide to use in config-Block via DI (Beispiel 3.9)
    - to config or register Services
    - to define values to DI (006)

Controller

Controller

- Logic of a view, state or component
- Registration in a module
- simple connection via ng-controller
- Bind ctrl as Alias in $scope
    - use alias for binding in HTML

Controller

- **controller Method**
- obj {[name:string]: ConstructorFct} (007)
- name: string
- factory-method
  - class

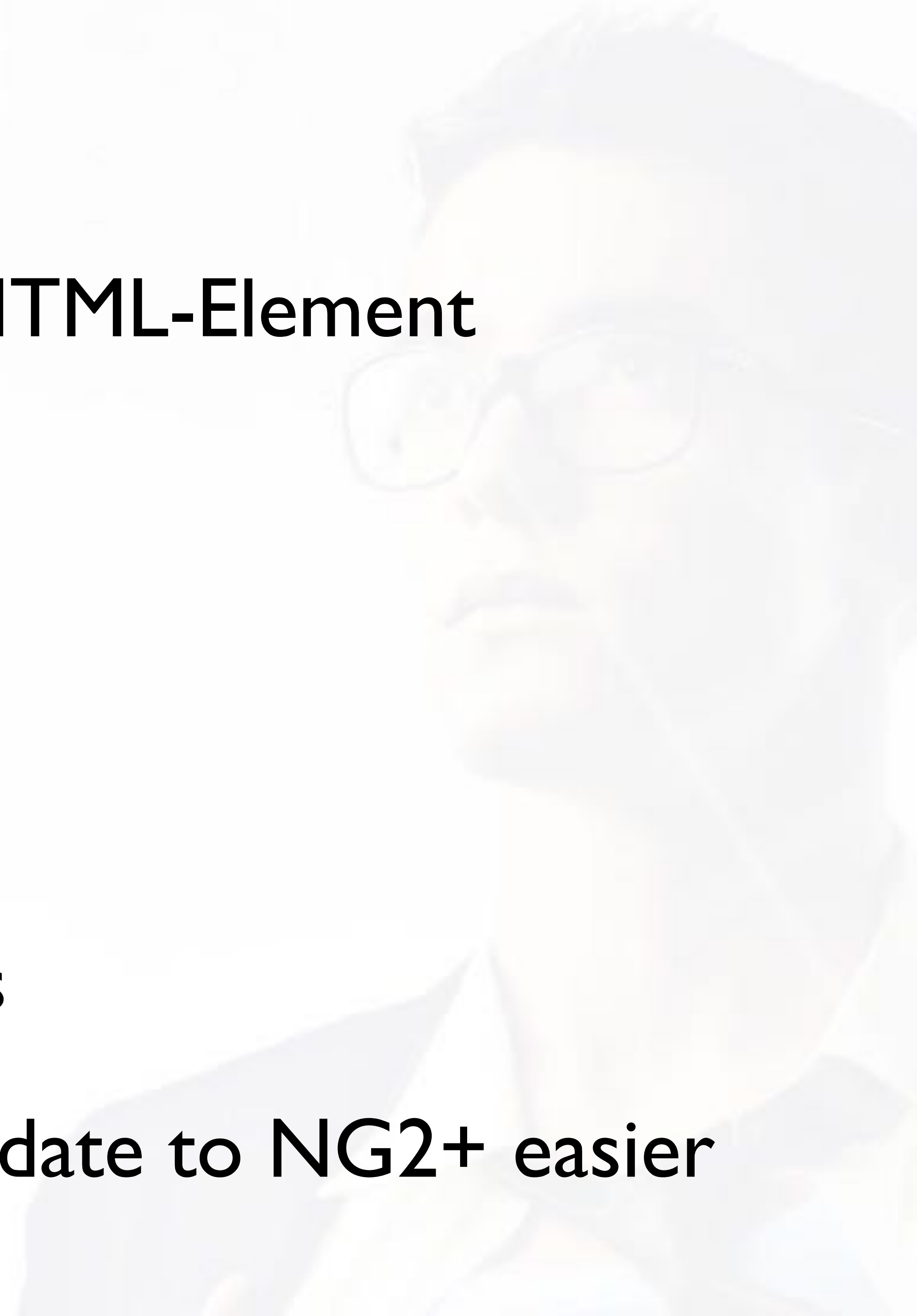
- **$controllerProvider**
- to use in Config-Block
- register (008)
  - name
  - factory-method
    - class

Controller Dependency Injection

- Dependency Injection
    - $scope: IScope
    - $element: IRootElementService

Scope and Binding

Scope & Bindungen



- Bridge between View and Logic
- **$scope** Automatically generated as singleton for HTML-Element
    - can be identified by **ng-scope** CSS class
- Model is made when binding
    - Properties and Methods
        - can be identified by **ng-binding** CSS class
- Best practice: Useing **Controller Alias** makes update to NG2+ easier

Binding

- **via Expression interpolation**
- Expression in curly brackets (009)
- Two colons before the expression -> one-time binding (011)
- Allowed expressions (if resolvable via $scope)
    - Properties, strings, operators
    - method return

Binding

- **via Directive**
- ng-bind
- ng-bind-html
    - only with ngSanitize module (010)
- events
    - e. g: ng-click
    - $event (011)

Binding

- ng-model (012)
    - Bidirectional binding
    - ng-model-options - Defines condition of model update
        - updateOn - Event-driven (013)
        - debounce - Time-controlled (013)
        - inherited form option

Scope

- **Hierarchical**
    - Object-oriented approach
    - automatic access to parents scope
    - Avoid: For easier upgrade to ng2+ (014)

Scope

- **root- and parent-Scope**
    - App-Scope injectable via $rootScope
    - $root: Scope property for accessing App-Scope
    - $parent: Scope property for accessing the parent scope (015)

Directives

Angular directives

- **Types**
- Logic and/or content
- **Usable over**
- HTML-Attributes
  ~~HTML-Nodes~~
- CSS-Classes

Existing directives

- event
- binding
- condition
- template
- style

Event Directives

- **Mouse events**
- ng-click, ng-mouseover, ng-mouseenter …
- **Change event**
- ng-change (ngModel required)
    - for input:text, checkbox, radiobutton, select (016)

Event Directives

- **Key event**
- ng-keydown, up, …
    - for input:text (018)
- **Clipboard**
- ng-copy
- ng-cut
- ng-paste (Beispiel 019)

Style Directives

- **Visual Directives**
- ng-cloak
    - prevents rendering of non def. Values in curly brackets (020)
- ng-class
    - Binds CSS classes to the HTML element of the directive (021)
        - expression, list, object and combined

Style Directives

- ng-style
  - sets style properties to HTML element (022)
- ng-show & ng-hide
  - sets the ng-hide class to make elements invisible (023)

Source and Link Directives

- **Defining Sources and Links**
- ng-src & ng-href
    - Binding source information via curly brackets (024)

DOM Directives

- **Remove or add HTML elements from the DOM**
- ng-if
    - add/remove HTML element in DOM depends from condition (025)
- ng-switch
    - ng-when
    - ng-default
    - switch HTML element in DOM depends from condition (026)

DOM Directives

- **iteration through list and add a HTML-Node for each item**
- ng-repeat
    - comparable to "for... of" all items of a list will be integrated and stored in a value variable.
    - During uses, AngularJS injects additional properties
        - $index, $first, $middle, $last, $even and $odd (027)

DOM Directives

- **iteration through list and add HTML fragments for each item**
- ng-repeat-start & ng-repeat-end
    - The fragment is defined between two directives.
    - The first directive (ng-repeat-start) controls the iteration
    - The last one (ng-repeat-end) defines the end of the fragment
        - (028)

Filter

Filter

- Usable within an expression
- A pipe Announces a filter
- Colons announces filter parameters
- {{ toFilter | filterName: parameter }}
- The filtered output will be rendered
- Combinations of filters are possible
- Internationalisation possible

Filter

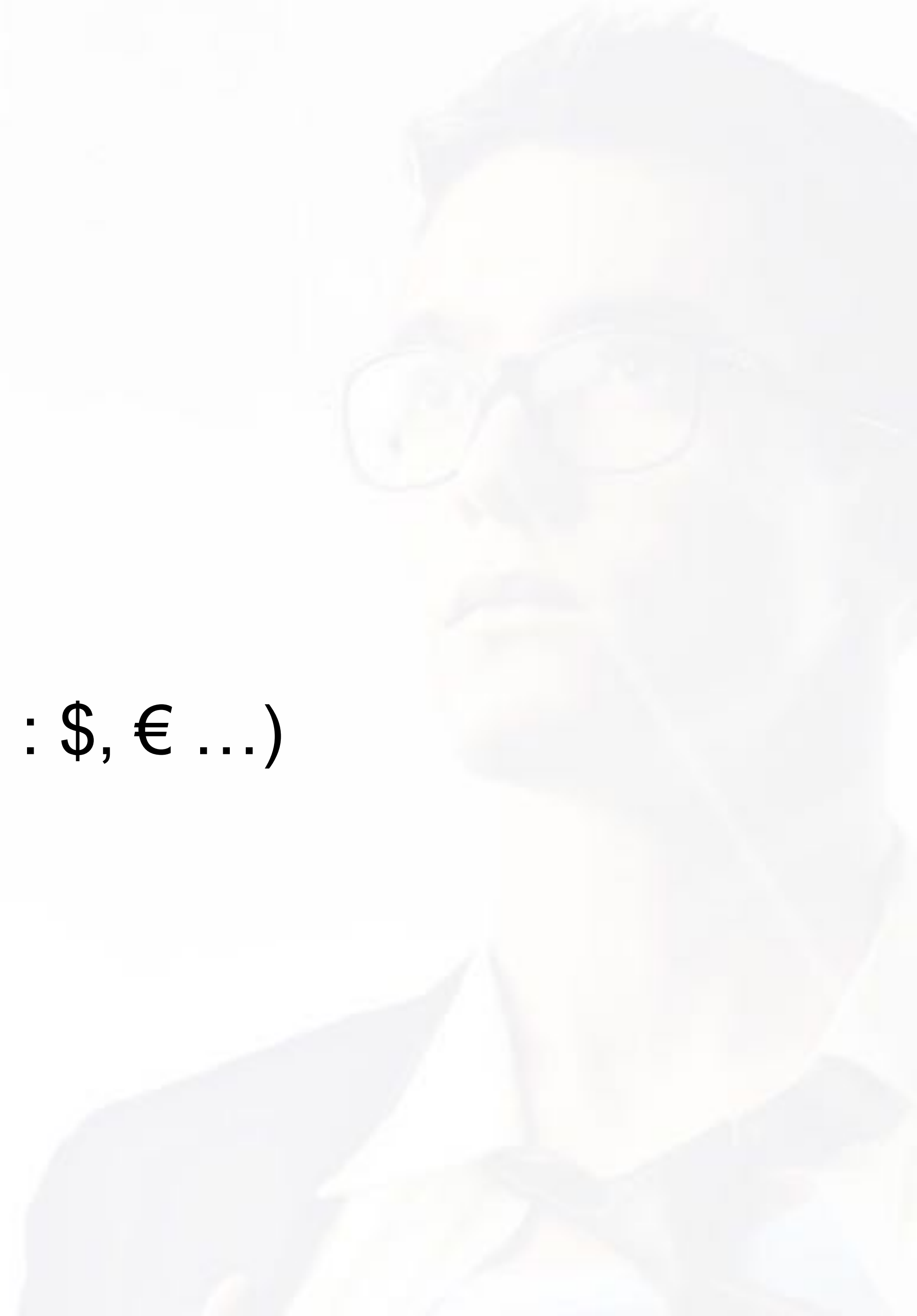
- **Existing**
  - uppercase & lowercase
    - Outputs the input in upper or lower case
  - limitTo
    - length [+/-number]
    - optional: start index [+/-number] (>=1.4)
    - Output limited entries [String, Array, Number]

Filter

- orderBy
    - Property Name (optional)[String]
    - Reverse (optional - default: false)[Boolean]
    - Sort a list (029)
- date
    - Pattern[string
    - timezone (>=1.4)
    - Returns a formatted date (030)

Filter

- number
  - Number of decimal places (optional - default: 3)
  - Returns a formatted number
- currency
  - Symbol (optional - default value depends from locale : $, € …)
  - Number of decimal places (optional - default 2)
  - Returns a formatted amount



Filter

- filter
    - search [string, object] (032)
    - searches for entries in a list (033)

Tips & Tricks

Tips & Tricks

- Disable Debug-information
  - Disable AngularJS debug attribute and class information
    - $compileProvider. debugInfoEnabled (false);
- jQuery can be included by initializing it before AngularJS
  - ng-jq directive since ng1.4
    - global variable referenced to: $.noConflict (true)

Tips & Tricks

- Minify JavaScript and Dependency Injection
    - Callbacks and factories defined as list
    - $inject

Services

Services

- Business logic that is independent of a view
- Existing: $log, $window, $document, $http and many more
- Singletons in the injector and therefor available via Dependency Injection
- provider with $get function (similar to getInstance Singletons Pattern)
- factory uses provider and generates $get using expected factory method
- service uses factory and $injector.instantiate (constructor)

Services

- $log
    - console.log for angularJS
    - $logProvider.debugEnabled(false); //disable all debug logs
- $window === window
- $document
    - jQuery Objekt of documents
- (035)

Services

- provider
    - Method expects a constructor for service generation
    - Property $get must contain a factory that returns the service object.
    - This Object will be handled as Singleton
    - (036)

Services

- factory
  - Method expects a factory method that returns the service object.
  - This Object will be handled as Singleton
- (037)

Services

- service
  - Method expects a class that returns the service object.
  - This Object will be handled as Singleton
- (038)

Services

- $cookies-service - >=1.4
    - get(key);
    - getObject(key);
    - getAll();
    - put(key, value, [options]);
    - putObject(key, value, [options]);
    - remove(key, [options]); (039)

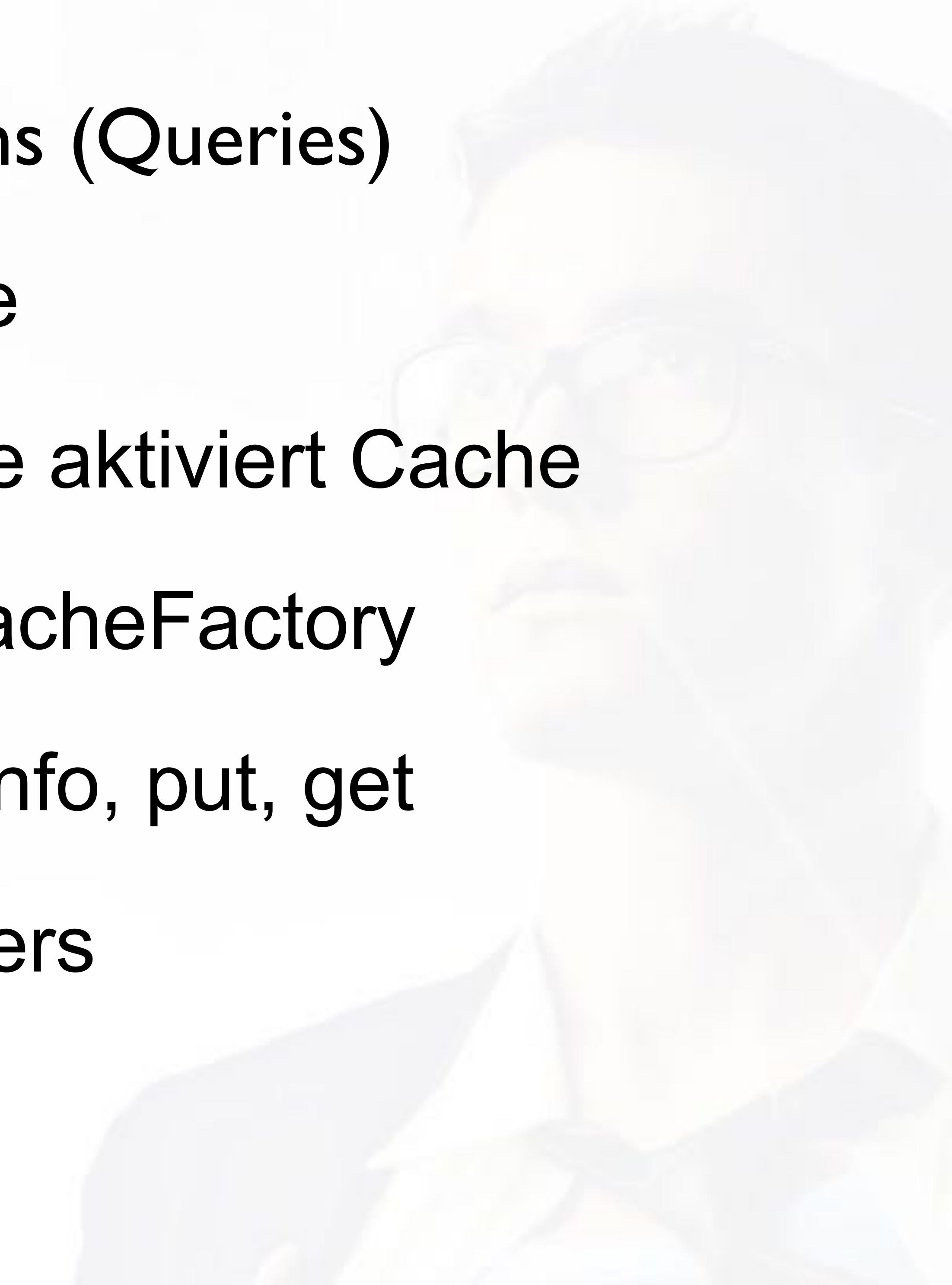
HTTP-Service

$http Service

Input

- config: IRequestConfig
    - method
        - POST, GET, PUT
    - url
    - data (Payload)

- params (Queries)
- cache
    - true aktiviert Cache
    - $cacheFactory
        - info, put, get
- headers



$http Service

Output

- promise: IPromise<IHttpPromise<T>>
- then(fnResult, fnErr, fnNotf)
  - IHttpResponse

$http Service

- IHttpResponse
  - data: T;
  - status: number;
  - headers: IHttpHeadersGetter;
  - config: IRequestConfig;
  - statusText: string;
  - xhrStatus: 'complete' | 'error' | 'timeout' | ‚abort'; (040)

$http Shortcut

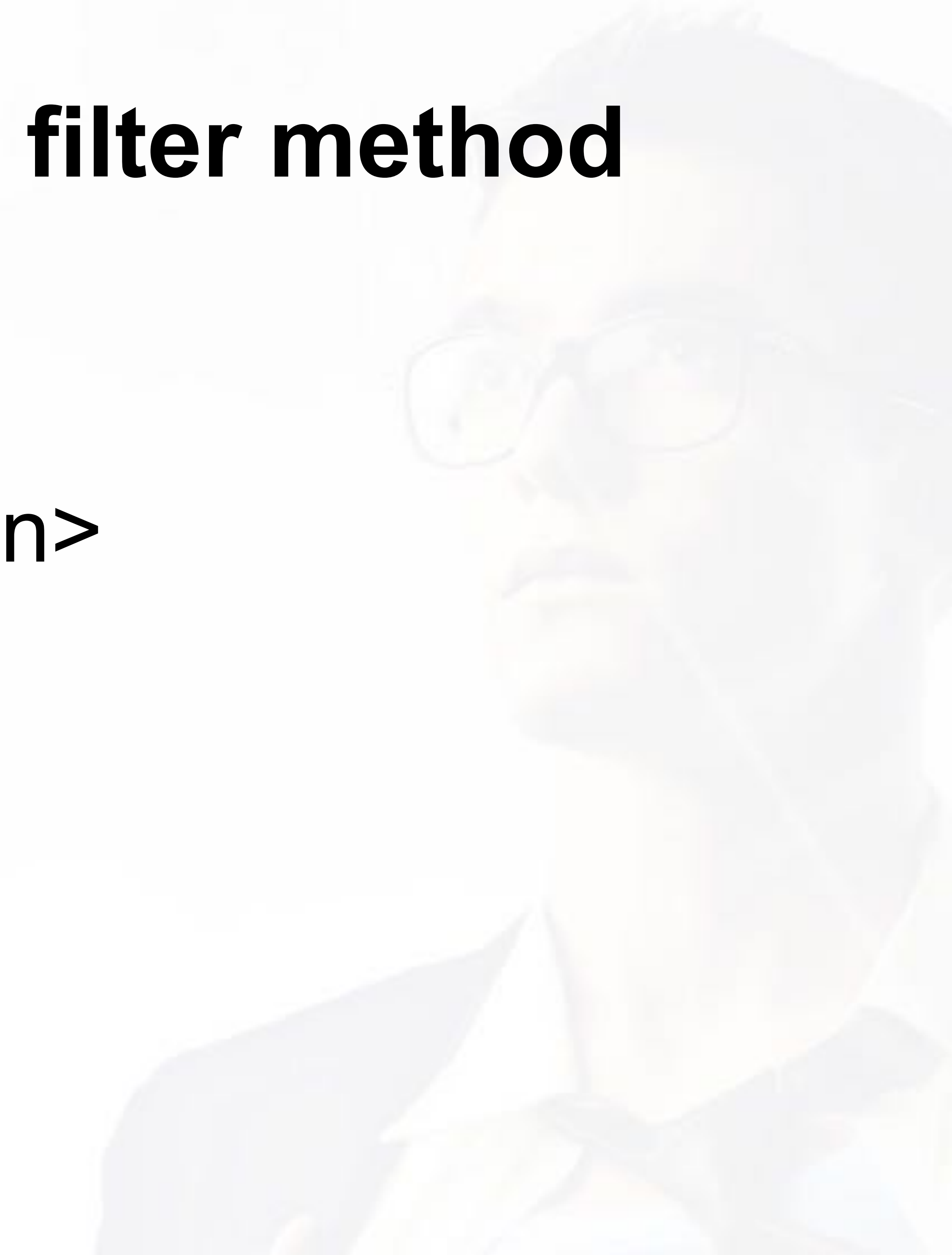
- get (url, [config]) (041)
- delete(url, [config])
- jsonp(url, [config]) JSON_CALLBACK
- post(url, data, [config]) (042)
- put(url, data, [config])

$http Params, Cache and Header

- Params (Queries) could be defined (043)
- Cache verwenden (044)
- $cacheFactory-Service (045)
    - defines a self controllable cache
- headers setzen (046)

Creating and Using Filters

Filter development



Register within the module with the filter method

- name: string,
- filterFactoryFunction: Injectable<Function>
    - returns the filter function
        - Input
        - ... Parameter [optional] (047)

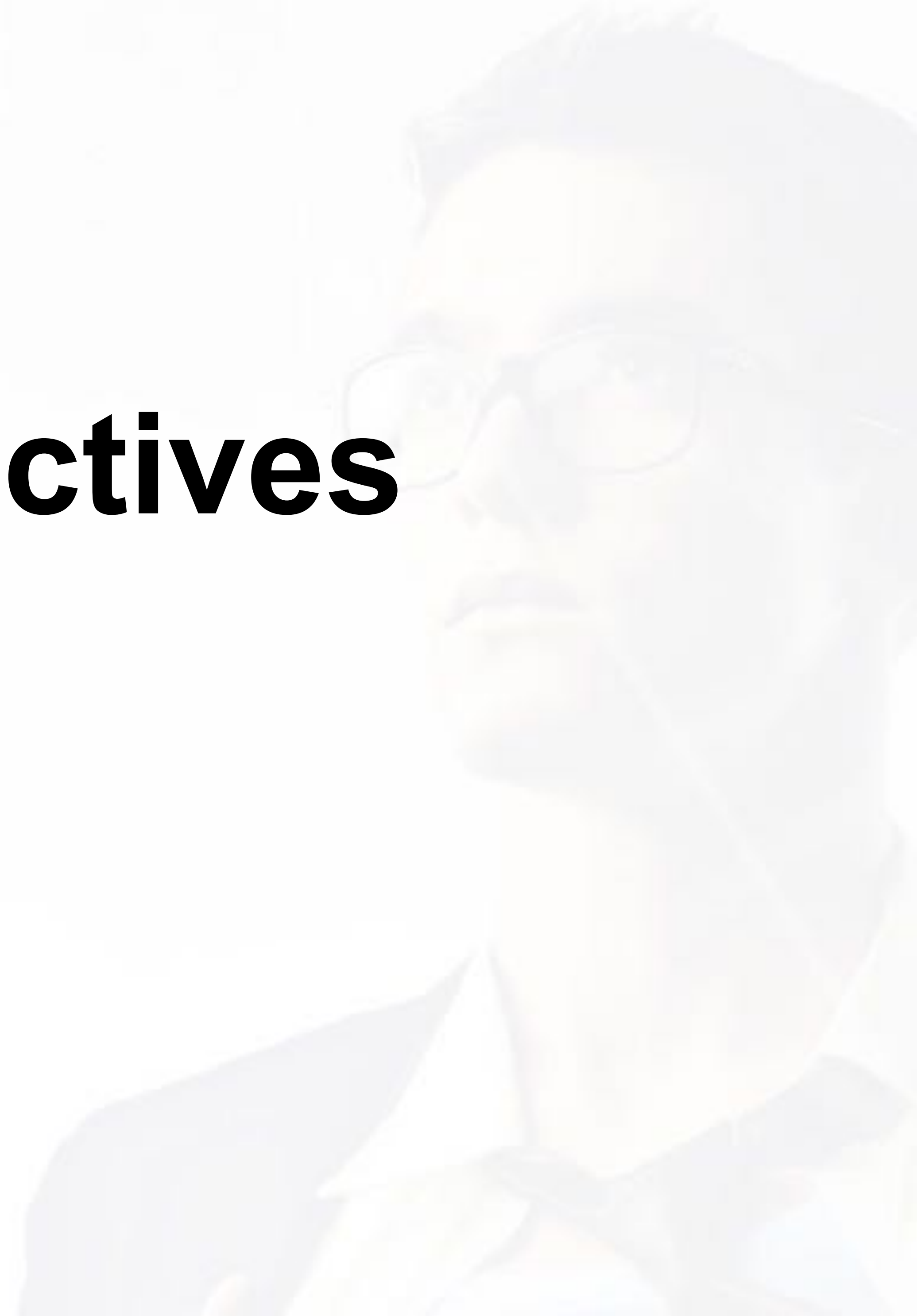
Filter usages

Filters can be used within an expression

- {{ input | filterName: parameter1: parameter2:.... }}}
- {{ 1234 | number: 2 }} //1.234,00

Filters are injectable

- inject. [filtername]Filter e. g. numberFilter
- console.log (numberFilter (1234,2)); (048)



Creating and using Directives

Directives Use

- <button **ng-click="…"**
- **<ion-list>** (use components instead since 1.5)
- <div class="**ion-list**" …

Directives Development

directive-Methode

- name: string
- config: IDirective

IDirective

- ~~template [string]~~
- ~~templateUrl [string]~~
- ~~templateUrl [function]~~
- ~~replace [bool]~~
- restrict [string] z.B. ‚A~~E~~C'

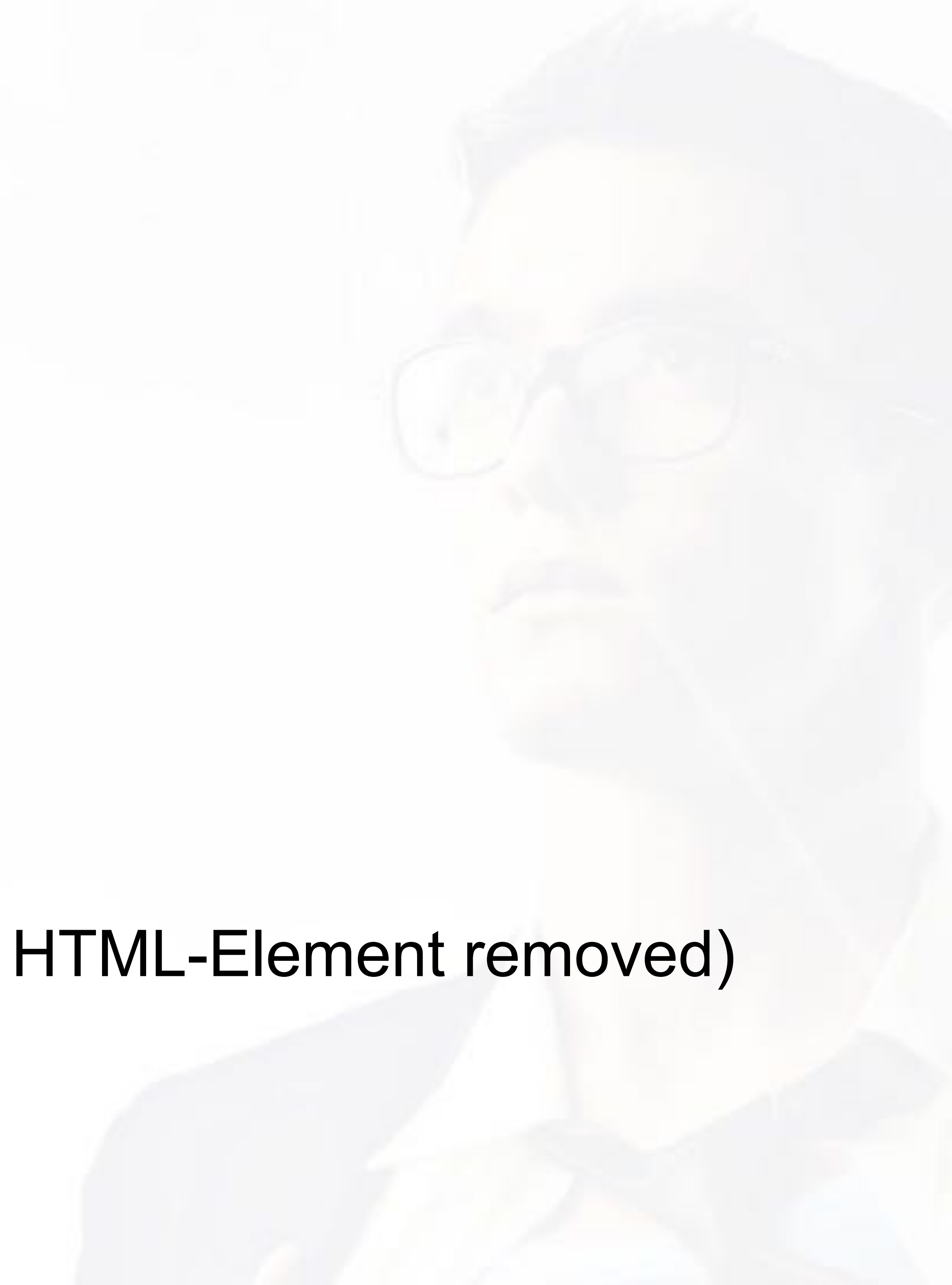
IDirective

- scope
  - Define to work isolate
  - true or properties
- define scope properties for isolated scope
- [scopePropertyName:string] : "[BindingType]Attribute-Name"
  - e.g. scope: { userName : =name } // Attribute "name" defines "$scope.userName"

- ~~controller [string or function]~~
- ~~controllerAs [string]~~
- ~~transclude [bool]~~

BindingType

- @ : Binding via expression in curly brackets e. g. name="{{pName}}"
- = : Regular Binding e. g. name= "pName".
- & : Delegate event



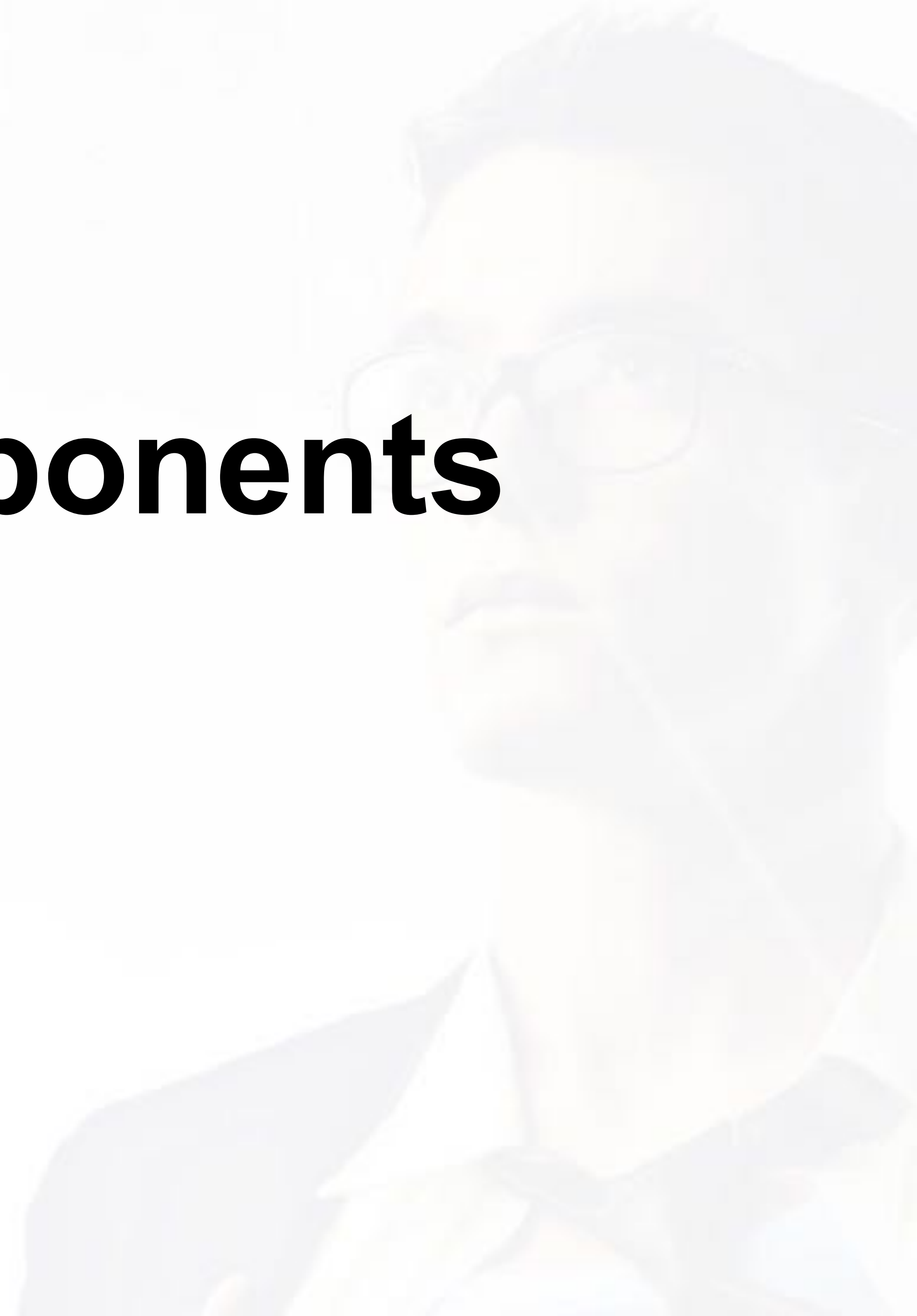
IDirective

- link: IDirectiveLinkFn

  Link-Attribute

  - scope: IScope
  - element: IAugmentedJQuery
  - attributes: IAttributes

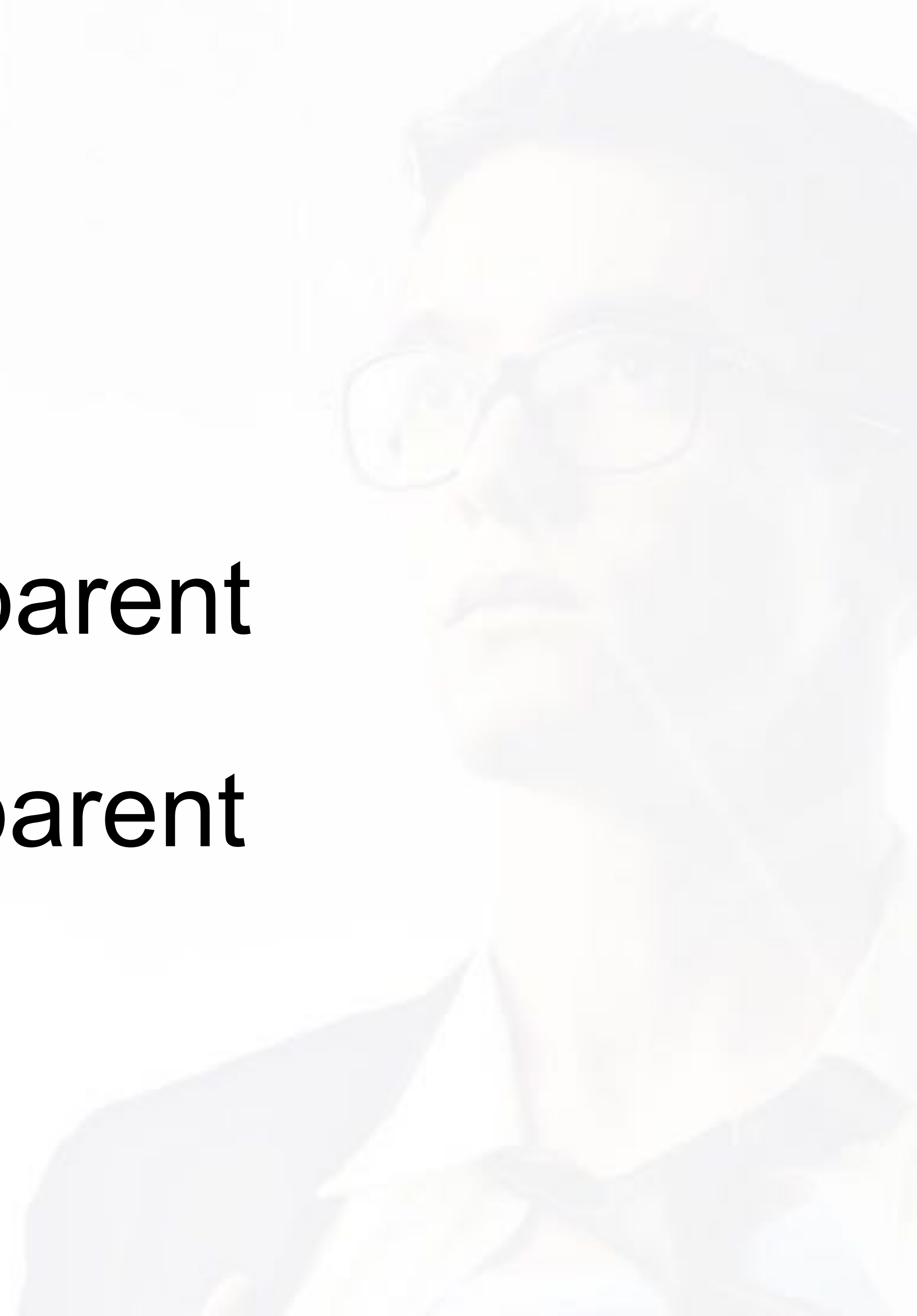
- Do not forget to destroy (jQuery ‚$destroy' event, when HTML-Element removed)
- (049)



Creating and Using Components

Component usage

-
  ```
  <ion-list
          titel=`$ctrl.myTitel`
          update=`$ctrl.doUpdate()`>
  ```
- attribute Title will be define bind from parent
- update will be triggered to inform the parent about updates



Component development

component-Methode

- name: string,
- options: IComponentOptions

IComponentOptions

- template [string|function] (050)
- controller [string oder function] (051)
- controllerAs [string - default $ctrl]
- transclude [bool - default false] (052)

Components Parent Child Communication

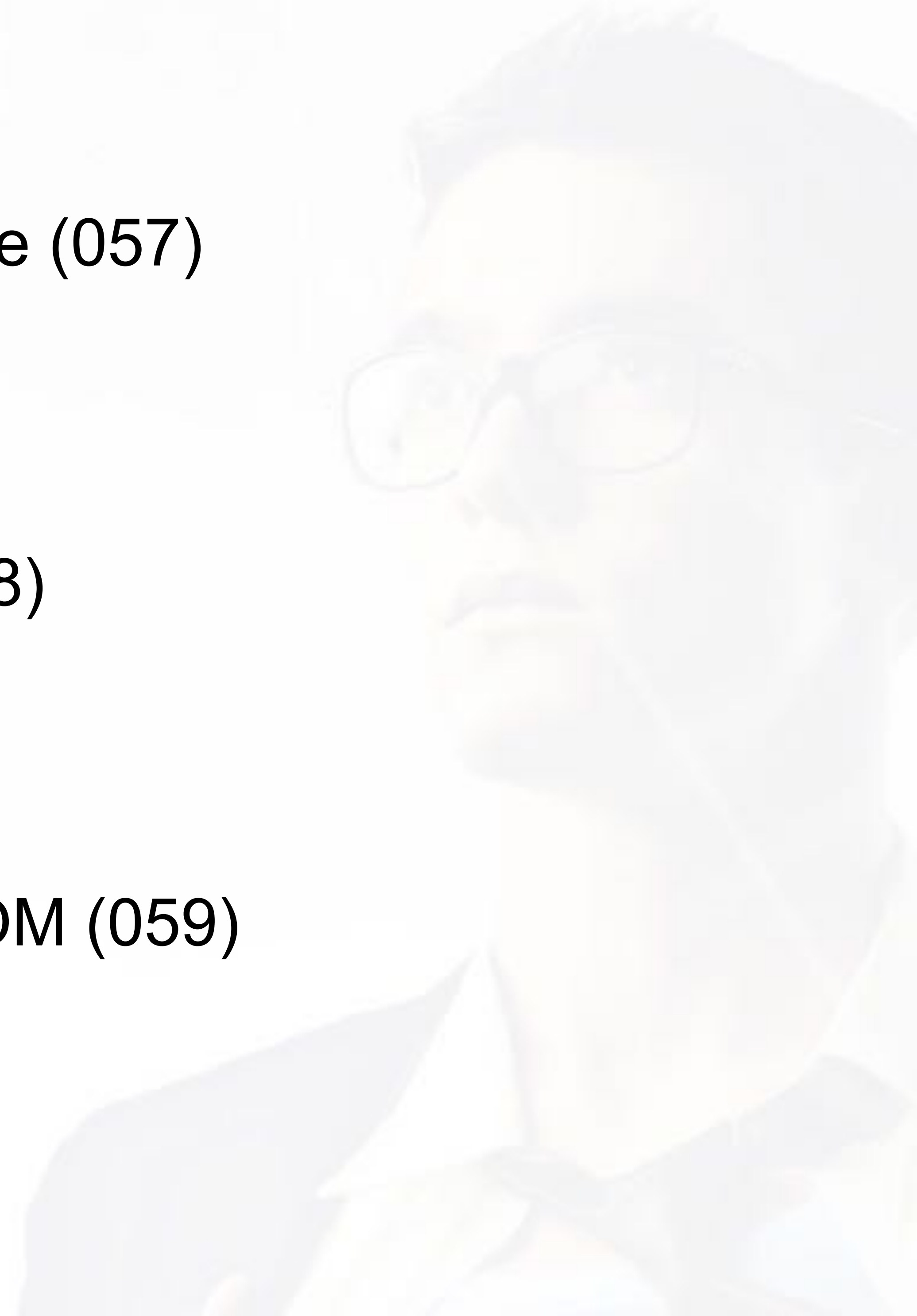
- **bindings works like isolated Scope in directive but defines all properties directly within the controller instance**
- @ : Binding via expression in curly brackets e. g. name=„{{pName}}" (053)
- = : Regular Binding e. g. name= „pName". (053)
- & : Delegate event (054)
- < : oneway binding for Components only - (053)

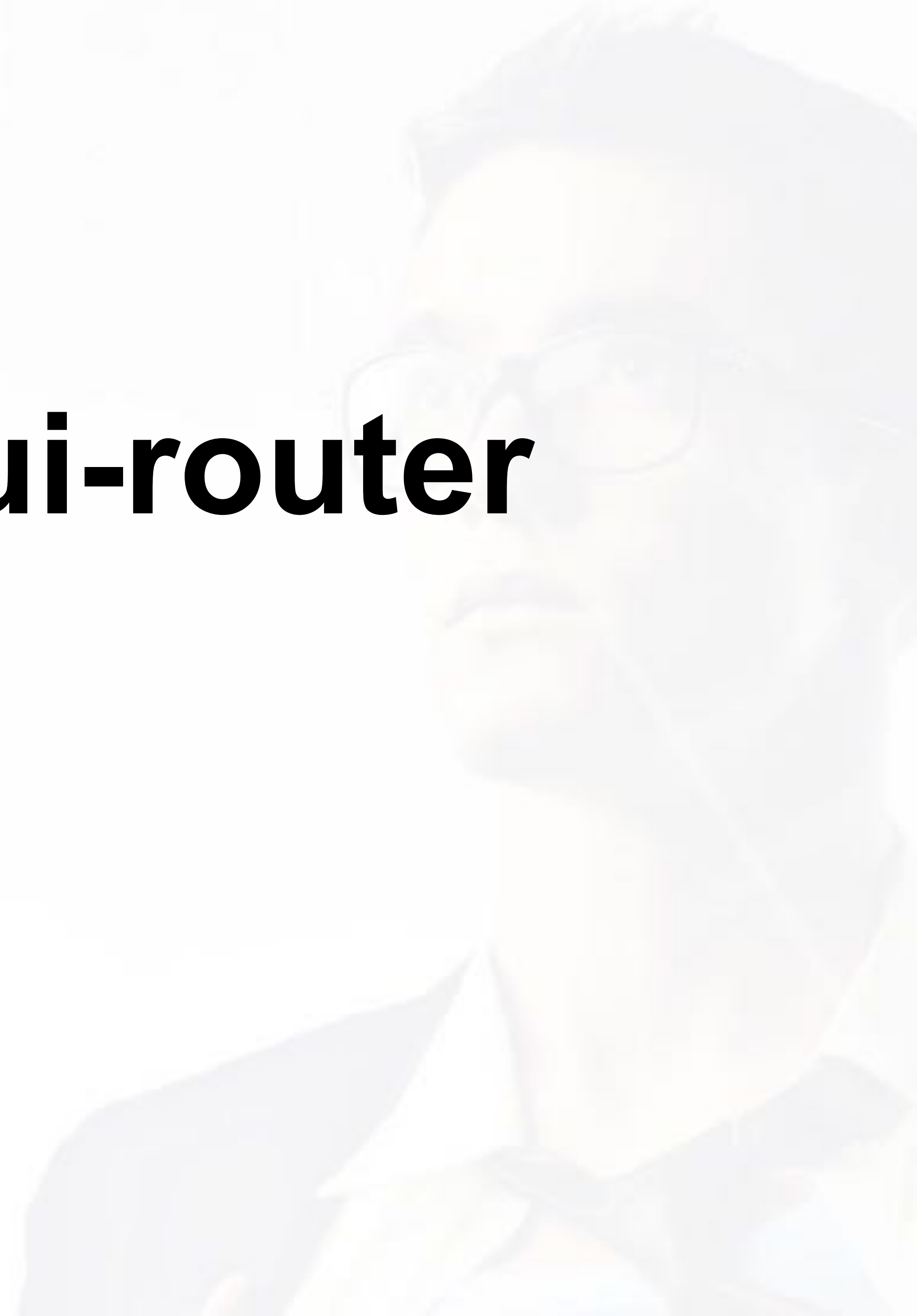
Required components

- **require: {[controller: string]: string}**
    - Component Options property to define, which component are required as parent component.
    - controller
        - key that will be used to define the parent component controller instance within a property in the child parent component controller
        - value is an expression that link zu the parent component Tag-Name
    - e.g. require: { userCtrl: '^user' }, (056)

Components Events

- $onInit
    - It'll be triggered, when bindings-properties are available (057)
- $onChanges (onChangesObj: IOnChangesObject): void
    - It'll be triggered, when bindings-properties change (058)
- $onDestroy
    - It'll be triggered, wenn the component removes the DOM (059)





Component routing with ui-router

Prearrangement

- prearrangement
    - load and include the module ‚**ui.router**' as a dependency
- $locationProvider
    - user provider to enable the -> **html5Mode** (true)
- base: href
    - Define base: href because html5Mode (do not forget Mod_Rewrite)

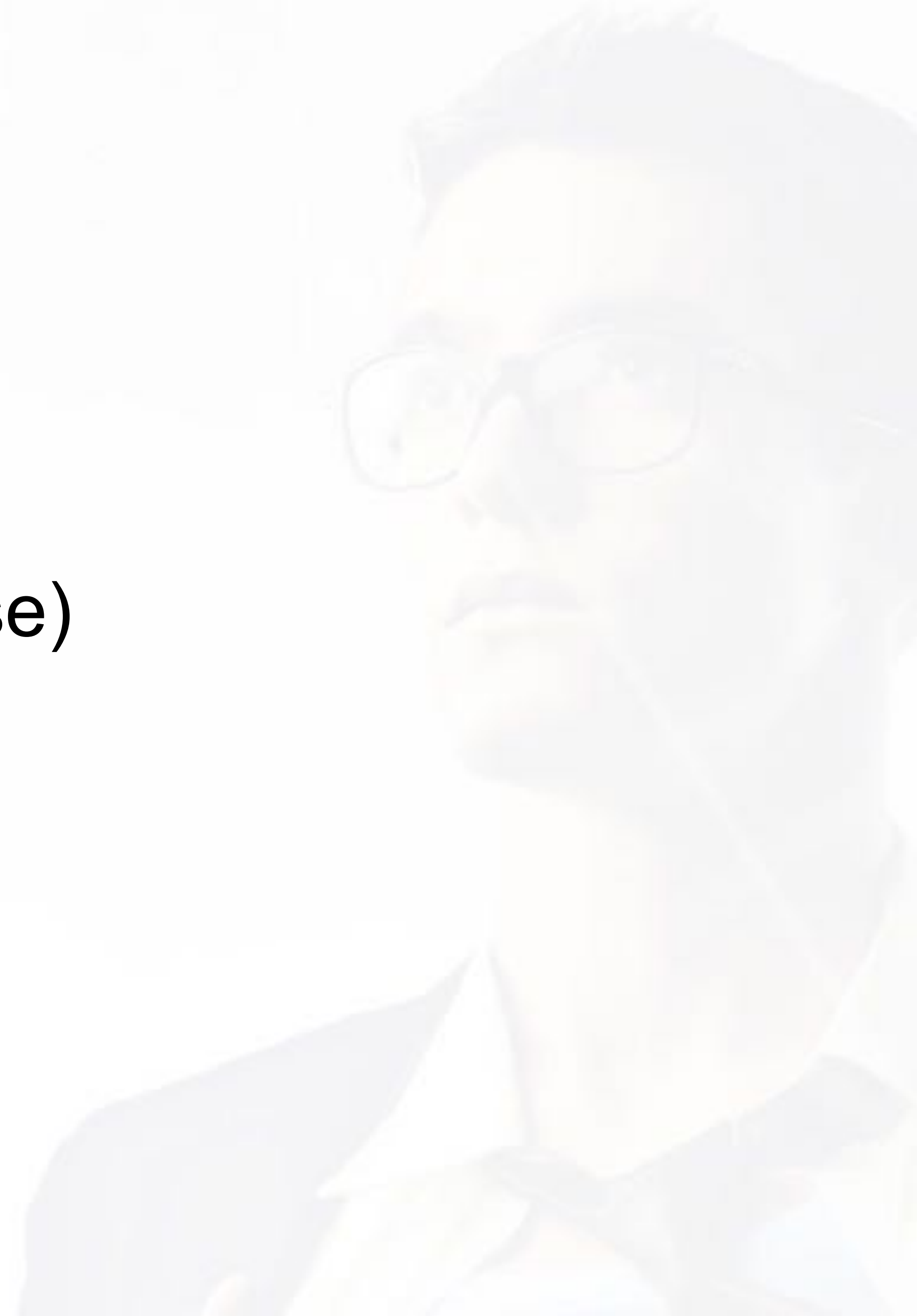
Prearrangement

- add ui-view Node as Route- Component container
    - <ui-view>

State configuration

Ng1StateDeclaration

- url: string - **route-path**
- name: string - **state-name**
- componente: string - **componente-name** (camelCase)



State registration

use $stateProvider: StateProvider Service

- state method
    - state: Ng1StateDeclaration

State not found

- do not forget to define otherwise rule for 404
    - use $urlServiceProvider: UrlService
        - otherwise method of rules
            - {state: ‚nameOf404State‘}

Route changing

Via ui-sref Directive

- name of state
- name of state as function
  - parameter { key: val }

helper

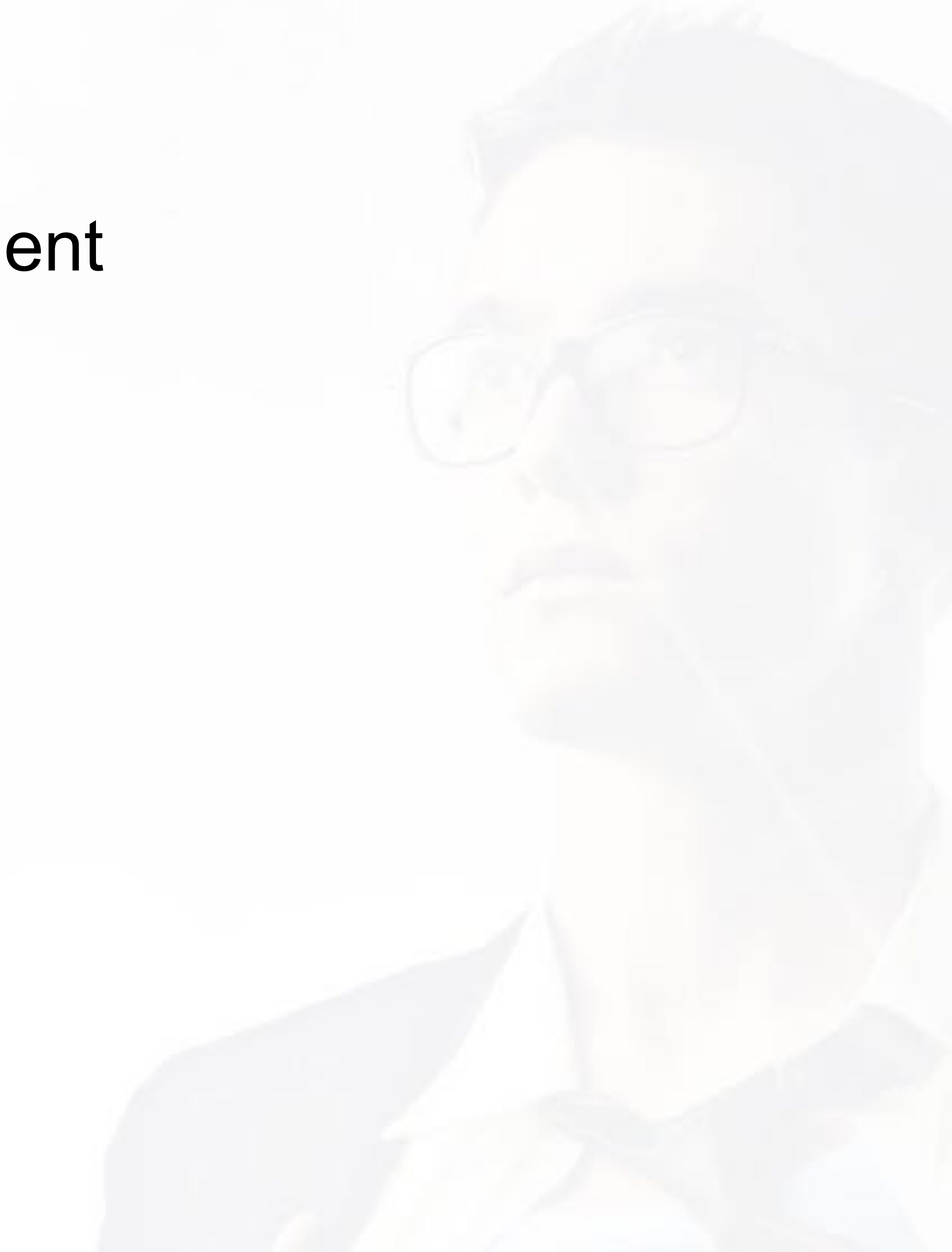
- ui-sref-active
  - set css-class if state is active

Via $state: StateService

- go - method
  - state name
  - params Object
    - [key: string]: any

Routing with resolve

- resolve object has key value pairs
  - key => data that will be bind at route-component
    - bindings: { key: ,<,}
  - value: any or promise



Routing with parameter

- define URL with Parameter Information
    - url: ’/user/{id}’
- get Params with
    - $transition$: Transition
        - .params().[param.name]

Routing with parameter

add resolve to state definition

- resolve: object - needed to send **data** to routeState before route-change
    - Key: Name where data should be send with
        - Value: Function that return value or promis
- key must be use as binding ‚<‛ in Component

Routing with childs

- child route state name must extend parent state name with dot syntax
    - parentStateName.childStateName
- child url must be relative
    - e.g. as Param
        - ‚/:to‘

Forms

Forms basics

- <form **novalid** ...> use novalid to avoid browser form validation

ngModel

- defines data in Model
    - depends of type -> data will stored if valid
- useable e.g.
    - Checkbox/Radio, Mail, Number

Form CSS validation

- ng-valid – class will be added if Model is valid
- ng-invalid – class will be added if Model is not valid
- ng-valid-[key] e.g.: ng-valid-minlength // ng-minlength
- ng-invalid-[key] e.g.: ng-invalid-required // required
- ng-pristine – class will be added if input element is pristine
- ng-dirty – class will be added if input element was edited
- (063)

Formular-CSS-Validierung

- ng-touch – Has been in focus before
- ng-untouch – Never been in focus before
- ng-pending – Validation not yet completed
- Example for an error display (red background)

```
.ng-touched.ng-invalid,  .ng-dirty.ng-invalid {
    background: red ;
  }
```

- Example of a correct display (green background)

```
.ng-touched.ng-valid,  .ng-dirty.ng-valid {
    background: lightgreen ;
  }
```

Form directives

- ng-model / ng-model-options
- ng-submit
- ng-readonly
- ng-selected / ng-checked
- ng-disabled
- ng-minlength / ng-maxlength
- ng-pattern

Form service

- scope.[form-name].$valid [bool] – true, when all form-data valid
- scope.[form-name].[input-control-name].$valid [bool] – true, when input valid
- Analogous to valid there is $error[object]. The object properties provide information about validation errors
- { "required": true, "number": false, "max": false, "min": false }
- <button ng-click="update(user)“
  ng-disabled="form.$invalid">SAVE</button>