

Hands-on course , 3
day(s)
Ref : FAN

Participants

Architects, Web developers
and project managers who
want to optimize their Web
applications with JavaScript.

Pre-requisites

Good knowledge of HTML
and Web technologies.
Knowledge of the JavaScript
programming language.

Next sessions

AngularJS: Mastering Google's JavaScript Framework

OBJECTIVES

Developed by Google, AngularJS is a structuring framework that simplifies the development of rich client-end applications. This course will give you proficiency in the framework's key features: Filters, controllers, templates, etc. You'll also see how it integrates into a REST architecture.

[1\) JavaScript refresher](#)

[2\) Overview of the AngularJS framework](#)

[3\) Controllers and scope management](#)

[4\) Module and dependency injection](#)

[5\) Defining routes](#)

[6\) Data feature, Server Exchange](#)

[7\) Integrating automated tests](#)

[8\) Best practice and tools](#)

1) JavaScript refresher

- Components of a Web application. HTML5 and CSS3
- Web/JavaScript development tools.
- HTML and JavaScript rendering engines. DOM, BOM.
- JavaScript refresher: Prototypes, closures, and callbacks.
- Refresher on object concepts in JavaScript.
- REST-oriented Ajax application.

Exercise

Configuring the environment.

2) Overview of the AngularJS framework

- AngularJS, positioning: jQuery, ExtJS, etc.
- Integration/ Compatibility, versions, documentation.
- Features and general principles.
- AngularJS directives, HTML compiler.
- Expressions. Two-way data-binding. Filters.

Exercise

Preparing an HTML model for development with AngularJS.

3) Controllers and scope management

- Creating and using controllers.
- Context management, the variable \$scope.
- Event propagation. API. Dirty checking.
- Processing and approving forms.

Exercise

Integrating controllers.

4) Module and dependency injection

- Features of the object angular.
- Principle of dependency injection.
- Concept of a module. Configuration.
- Dividing your application.

Exercise

Modularizing the application.

5) Defining routes

- Routing. API (\$routeProvider). Deep linking.
- History and access to URL parameters.
- Use of \$location and \$routeParams. Hashbang and HTML5 modes.
- View pre-processing. Use of "fragments".

Exercise

Creating a Single Page Application.

6) Data feature, Server Exchange

- API (\$provide, \$injector). Creating a provider.
- Methods: Service, Factory, Provider, Value.
- Ajax query with the \$http service. Integrating REST with the service \$resource. WebSockets. Promise API.
- Integration with Node.js.

Exercise

Integrating REST with Node.js.

7) Integrating automated tests

- Test utilities: Jasmine. Angular-scenario. Test: controllers, services, etc. Use with Karma.
- End to End Testing: User interface.

Exercise

Creating unit tests.

8) Best practice and tools

- Yeoman, optimizing development.
- Internationalization (I18N), implementation.

Workshop

Implementation.