Telerik Academy

Node.js View Engines

End-to-end JavaScript Applications

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

- View Engines
  - Overview
- Client-side view engines
  - KendoUI, Handlebars.js, AngularJS
- Server-side view engines
  - Jade

Telerik Academy

View Engines

Telerik Academy

View Engines

- View engine (template engine) is a framework/library that generates views
- Using a programming language
- Web view engines are a mix-up of HTML and JavaScript
  - Given a template/view JavaScript generates a valid HTML code

Telerik Academy

JavaScript View Engines

- There are lots of JavaScript view engines, and they can be separated into client and server
  - Client: KendoUI, Handlebars.js, jQuery, AngularJS, etc.
  - Server: Jade, HAML, EJS, Vash, etc.
- Why use view engines?
  - Speed-up developer performance and easify the writing of HTML code
  - Auto generate DOM elements and make manual DOM manipulation almost useless

Telerik Academy

Client View Engines

KendoUI, AngularJS, Handlebars.js

Telerik Academy

Client View Engines

- Client view engines (templates) are used to parse data
  - The data is fetched from some place
    - i.e. with AJAX
  - The data is either raw JSON, XML or plain text
- Server view engines parse the data on the server
  - The client (browser) receives the read-to-use HTML

Telerik Academy

Templates with Handlebars.js

- Handlebars.js is a library for creating client-side templates
- Handlebars supports:
  - One-time value-binding to JavaScript objects
  - Iteration over a collection of elements
  - Conditional templates

Telerik Academy

Handlebars: Example

- Generates a list with all mustache types

Telerik Academy

Handlebars.js: Example

- Generates a list with all mustache types

```
<h1>{{title}}</h1>
<ul class="mustaches-list">
{{#types}}
  <li>
    <input name="mustaches[]"
           id="mustache-{{.}}"
           type="radio"
           value="{{.}}" />
    <label for="mustache-{{.}}">
      {{.}}
    </label>
  </li>
{{/types}}
```

<- Template

Telerik Academy

Handlebars.js: Example

- Generates a list with all mustache types

```
<h1>{{title}}</h1>
<ul class="mustaches-list">
{{#types}}
  <li>
    <input name="mustaches[]"
           id="mustache-{{.}}"
           type="radio"
           value="{{.}}" />
    <label for="mustache-{{.}}">
      {{.}}
    </label>
  </li>
{{/types}}
```

<- Template

JavaScript

```
var model = {
  title: 'Hello mustache!',
  types: ['Hungarian', 'Dali',
'Imperial', …]
}
```

Telerik Academy

Handlebars.js : Example

- Generates a list with all mustache types

```
<h1>{{title}}</h1>
<ul class="mustaches-list">
{{#types}}
  <li>
    <input name="mustaches[]"
           id="mustache-{{.}}"
           type="radio"
           value="{{.}}" />
    <label for="mustache-{{.}}">
      {{.}}
    </label>
  </li>
{{/types}}
```

<- Template

JavaScript

```
var model = {
  title: 'Hello mustache!',
  types: ['Hungarian', 'Dali',
'Imperial', …]
}
```

- All binding is done inside {{ }}

Telerik Academy

Handlebars.js Templates

Live Demo

Telerik Academy

KendoUI Templates

- KendoUI templates are part of the KendoUI framework
  - Can be found in `kendo.core.js`
  - Can be used stand-alone
- KendoUI templates supports:
  - One-time value-binding to JavaScript objects
  - Iteration over a collection of elements
  - Conditional templates

Telerik Academy

KendoUI Templates: Example

- Generates a table of technologies

```
<h1>#: title #</h1>
<table>
  # for (var i = 0; i < technologies.length; i+=1) { #
    <tr>
      <td><input type='checkbox' id="technology-#: i #" /></td>
      <td><label for="technology-#: i #">
            #: technologies[i].name #
          </label>
      </td>
</table>
```

```
var model = {
  title: 'Technologies',
  technologies: [{ name: 'ASP.NET', field: 'web' },
                 { name: 'Node.js', field: 'web' },
                 { name: 'WPF',     field: 'windows desktop' },
                 { name: 'Android', field: 'mobile' }]
};
```

Telerik Academy

KendoUI Templates

Live Demo

Telerik Academy

AngularJS Templates

- AngularJS templates are a part of the Core AngularJS framework
  - They actually represent views for controllers
- AngularJS supports:
  - Two-way data and event binding to JS objects
  - Iteration over a collection of elements
  - Conditional templates

Telerik Academy

KendoUI Templates: Example

- Generates a slide of images

```
<div id="wrapper" ng-controller="ImagesController">
  <div class="slider">
    <strong>{{currentImage.title}}</strong>
    <img ng-src = "{{currentImage.src}}" width=800 />
    <ul class="slider-images-list">
      <li ng-repeat="image in images">
        <img ng-src="{{image.src}}"
             ng-click="changeCurrent(image)"/>
      </li>
    </ul>
  </div>
</div>
```

```
app.controller('ImagesController', function ($scope) {
  $scope.images = [{title: 'QA Academy 2012/2013 Graduation',
                    src: 'imgs/9511183282_cbe735bb73_c.jpg'} … ]
  $scope.currentImage = $scope.images[0];
  $scope.changeCurrent = function(image){
    $scope.currentImage = image;
  };
});
```

Telerik Academy

AngularJS Templates

Live Demo

Telerik Academy

Server View Engines

Telerik Academy

Server View Engines

- Server view engines return ready-to-use HTML to the client (the browser)
  - They parse the data to HTML on the server
  - *Web applications, created with server view engines are not real SPA apps
    - In most cases

Telerik Academy

Jade Template Engine

- Jade is a server view engine
  - Produces HTML as a result
  - Can be parsed:
    - Manually (using CMD/Terminal commands)
    - Automatically using a task runner
    - Automatically using framework like Express
- Jade is more expressive and dynamic than HTML
  - Jade template can be parsed based on JS models or conditionals

Using Jade

- Install Jade with Node.js:

```
$ npm install jade -g
```

- Create the `index.jade` file:

```
ul
  each val in [1, 2, 3, 4, 5]
    li= 'Item ' + val
```

- Run:

```
$ jade index.jade
```

- Generates `index.html` with content:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>
```

Telerik Academy

Using Jade

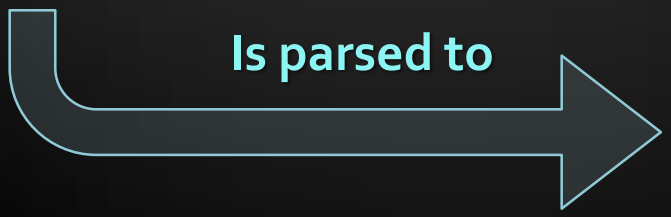
Live Demo

Telerik Academy

Jade Features: Tags

- Omit the opening and closing tags
  - Even their brackets
  - IDs and classes are set as in CSS selectors
    - #id and .class

```
#wrapper
  table.special
    tr
      th Header 1
      th Header 2
    tr
      td Data 1
      td Data 2
```



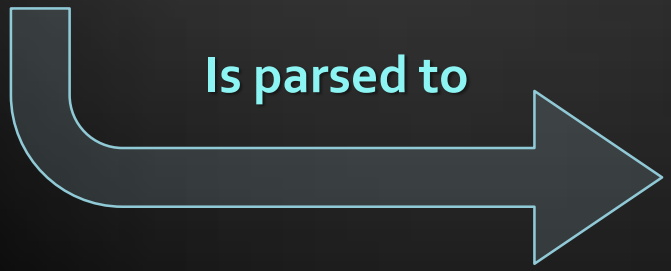

```
<div id="wrapper">
  <table class="special">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Data 1</td>
      <td>Data 2</td>
    </tr>
  </table>
</div>
```

Telerik Academy

Jade Features: Attributes

- Attribites are written inside '(' and ')'
  - And separated with commas ','

```
#wrapper
  header
    h1#logo
      a(href='...')
        img(src='…')
    nav#main-nav: ul
      li.nav-item
       a(href='…')
```




```
<div id="wrapper">
  <header>
    <h1 id="logo">
      <a href="...">
       <img src="..."/>
      </a>
    </h1>
    <nav id="main-nav">
      <ul>
        <li class="nav-item">
          <a href="...">...</a>
        </li>
      </ul>
    </nav>
  </header>
</div>
```

Telerik Academy

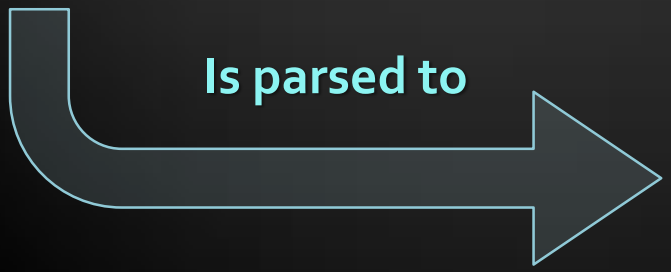
Jade Features

Live Demo

Jade Models

- Jade can generate markup, using data models
  - i.e. given an array of items, put them into a table

```
#wrapper
  header
    h1#logo
      a(href='...')
        = title
    nav#main-nav: ul
      each item in nav
        li.nav-item
          a(href= item.url)
            = item.title
```




```
<div id="wrapper">
  <header>
    <h1 id="logo">
      <a href="...">Lorem ipsum</a>
    </h1>
    <nav id="main-nav">
      <ul>
        <li class="nav-item">
          <a href="#home">Home</a>
        </li>
        <li class="nav-item">
          <a href="#about">About</a>
        </li>
      </ul>
    </nav>
  </header>
</div>
```

Telerik Academy

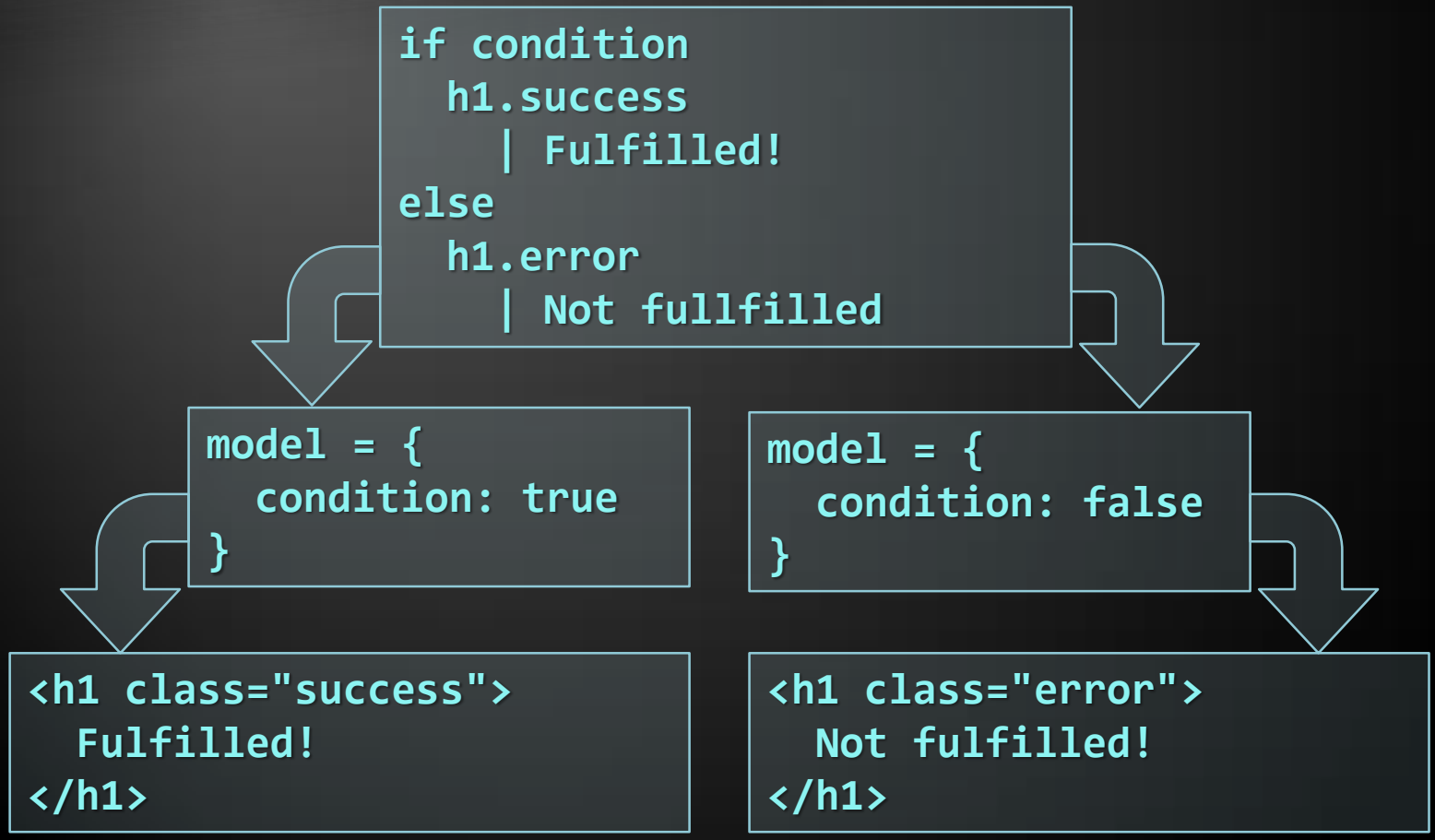
Jade Models

Live Demo

Telerik Academy

Running Script in Jade

- Jade can contain conditionals, loops, etc…
  - And the HTML is generated based on the model



Telerik Academy

Scripts in Jade

Live Demo

Telerik Academy

View Engines

http://academy.telerik.com

Telerik Academy

Homework

1. Create a simple web site for buying mobile devices
   - Create the following pages and put them into nav
     - Home -> contains greeting and site information
     - Smart phones -> contains a list of smartphones
     - Tablets -> contains a list of tablets
     - Wearables -> contains a list of wearables
   - All pages must have the same header, navigation and footer
   - Use Jade and Jade Layouts
   - Use the each directive to create the navigation