COMP721 Web Development

Week 12: Misc - JQuery and XML Rendering with XSLT

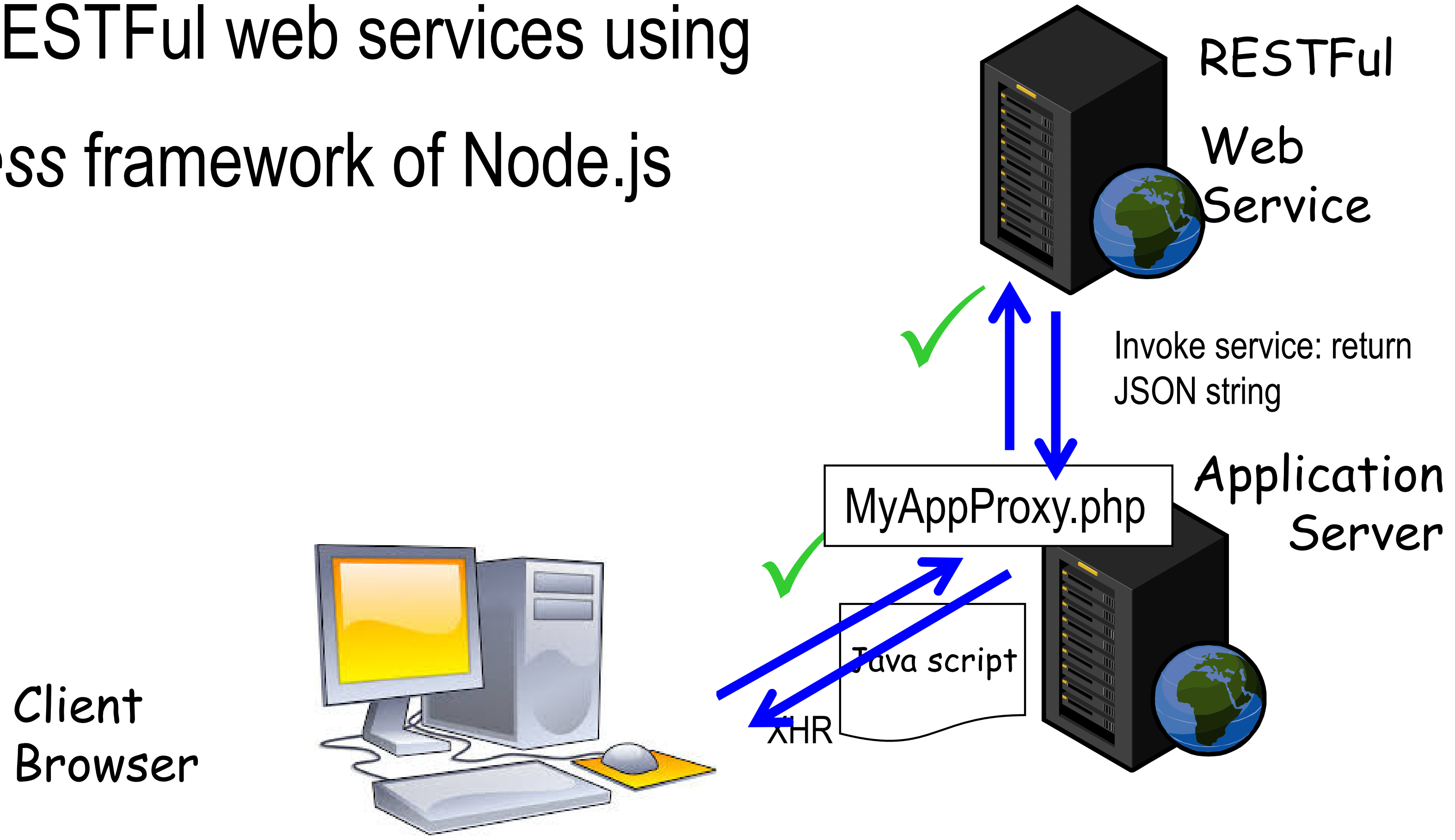
Agenda

- Week 11 review
- Vue.js framework
- Jquery
- XML rendering

Week 11 review: Web services

- Idea: use WWW as the middleware to invoke remote procedures/services
- Server side: invoking RESTFul web services using curl
- Building RESTFul web services using

  The *express* framework of Node.js



A RESTFul web service

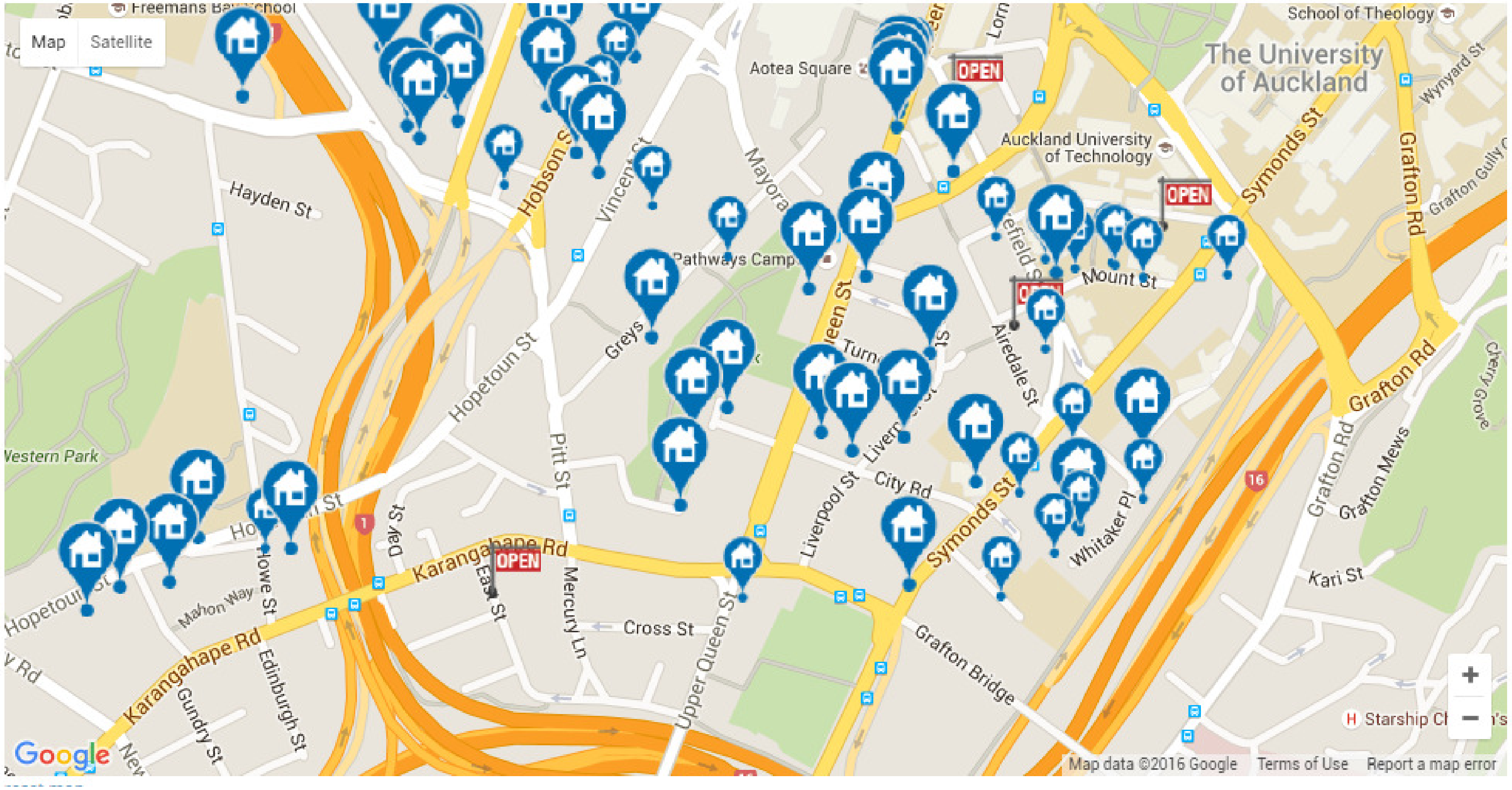
- http://api.openweathermap.org/data/2.5/weather?q=Auckland&APPID=2f4d83e3d50672cf2009fc34611903f3
- Name: api.openweathermap.org/data/2.5/weather
- Parameters: q, APPID
- The return is JSON data

Building RESTFul web services

- The key is to map/route HTTP requests to methods/functions implemented
- Routing example
  - HTTP GET Root request ‘/’ goes to index.html
  - HTTP POST ‘/users’ request goes to user.createUsers()
  - HTTP GET ‘/users’ request goes to user.seeResults()
  - HTTP DELETE ‘users/<id>’ goes to user.delete()

Client-side Web/JavaScript APIs

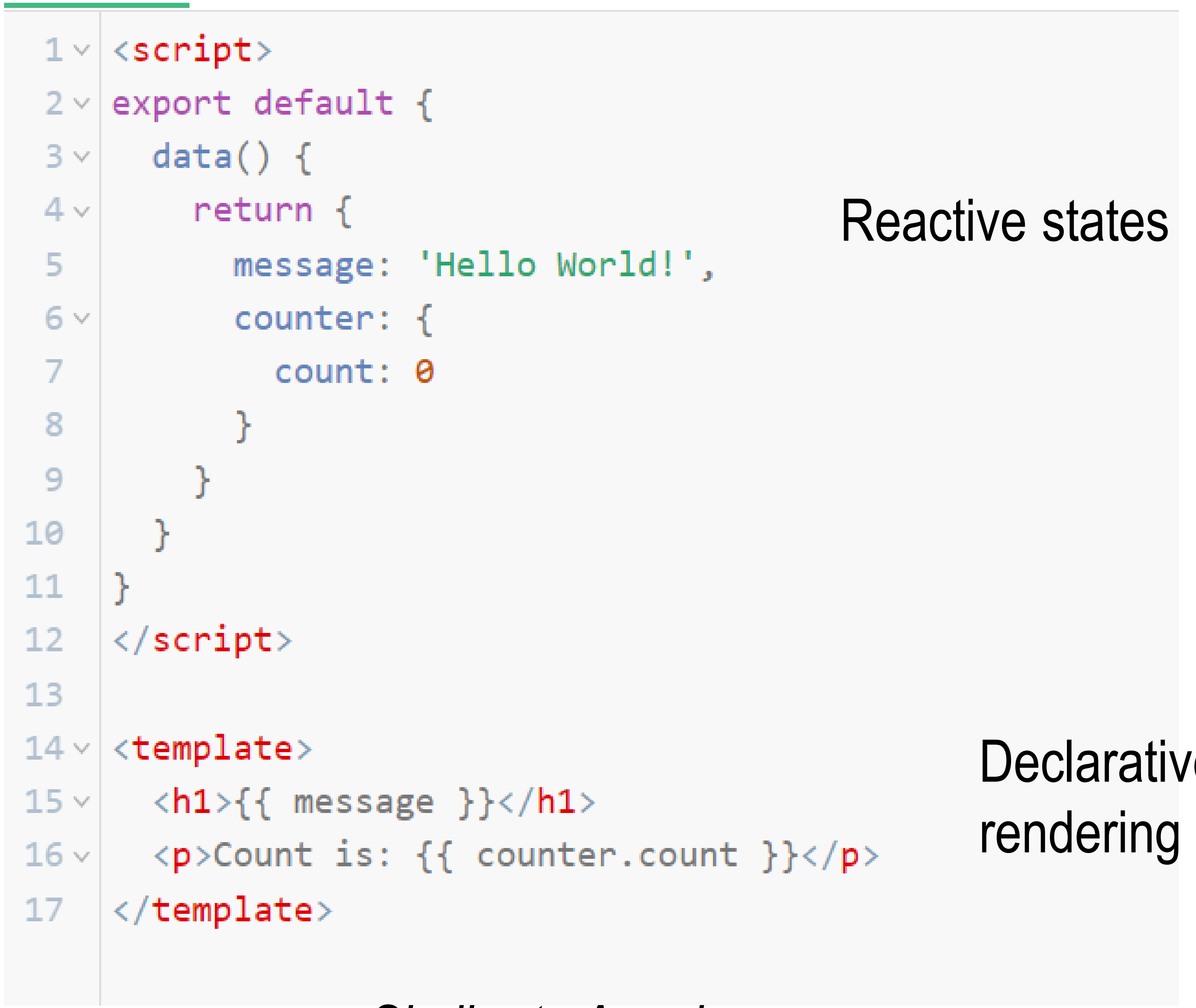
- Mashup: use the client to integrate data from different sources (including your own data)



Vue.js: JS framework

- Use single-file components (SFC) as extension to common web pages
- compiler-optimized rendering system
- Two core features of Vue:
  - Declarative Rendering: Vue extends standard HTML with a template syntax that allows us to declaratively describe HTML output based on JavaScript state
  - Reactivity: Vue automatically tracks JavaScript state changes and efficiently updates the DOM when changes happen

Declarative rendering and reactive states



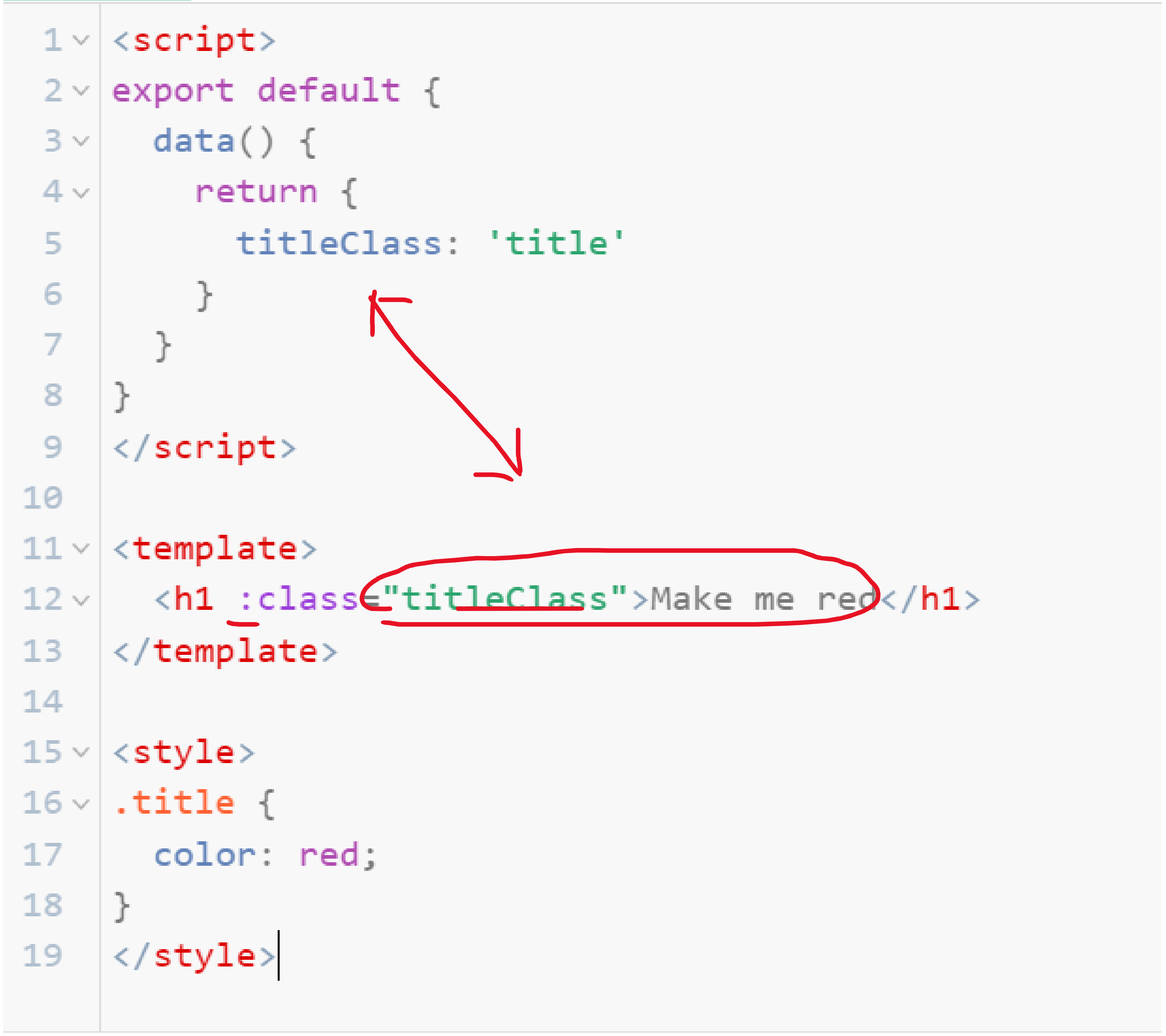
```
<script>
export default {
  data() {
    return {
      message: 'Hello World!',
      counter: {
        count: 0
      }
    }
  }
}
</script>

<template>
  <h1>{{ message }}</h1>
  <p>Count is: {{ counter.count }}</p>
</template>
```

Reactive states

Declarative rendering

Similar to Angular

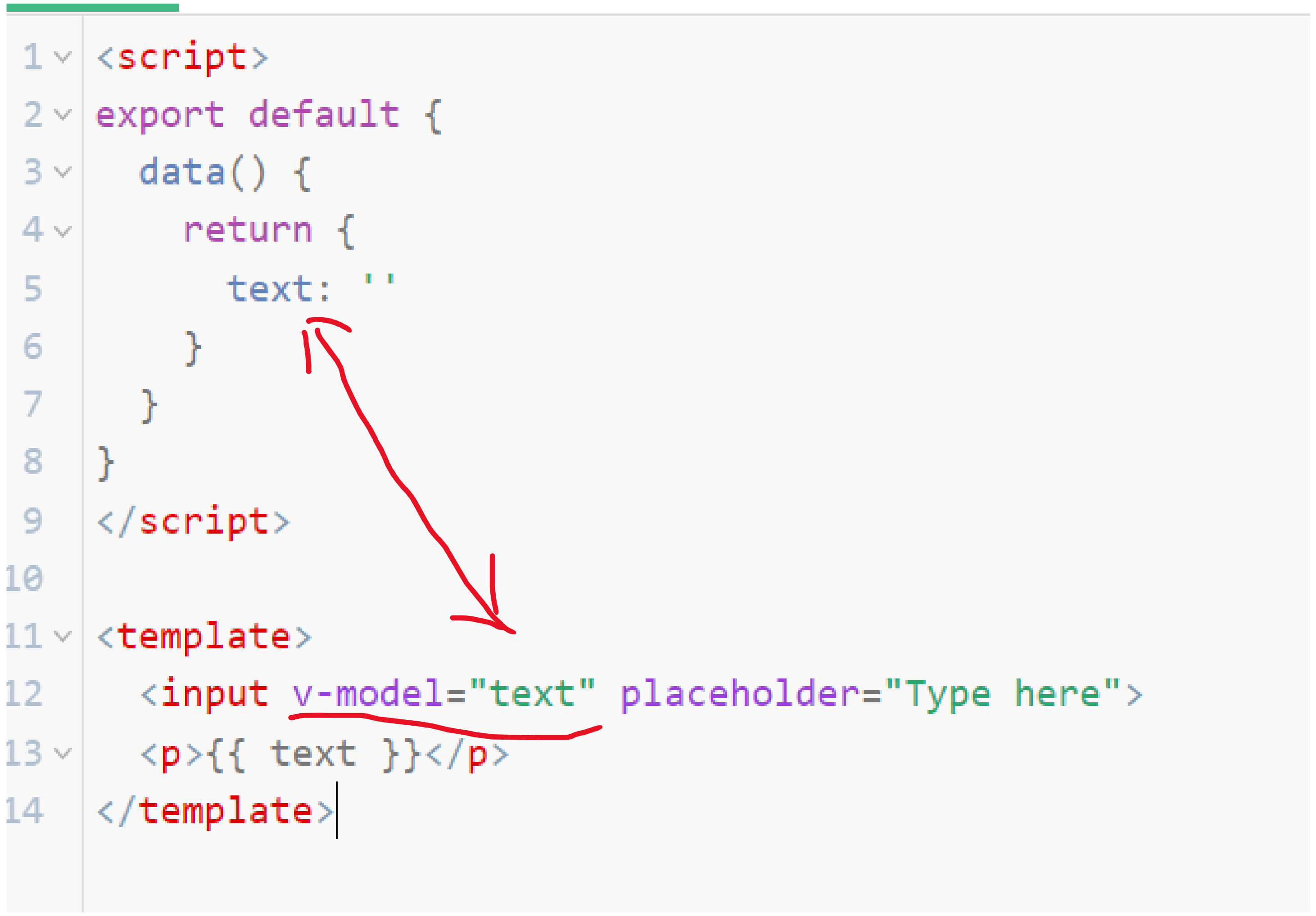
Dynamic attribute (put colon ":" before an attribute name)



```
<script>
export default {
  data() {
    return {
      titleClass: 'title'
    }
  }
}
</script>

<template>
  <h1 :class="titleClass">Make me red</h1>
</template>

<style>
.title {
  color: red;
}
</style>
```

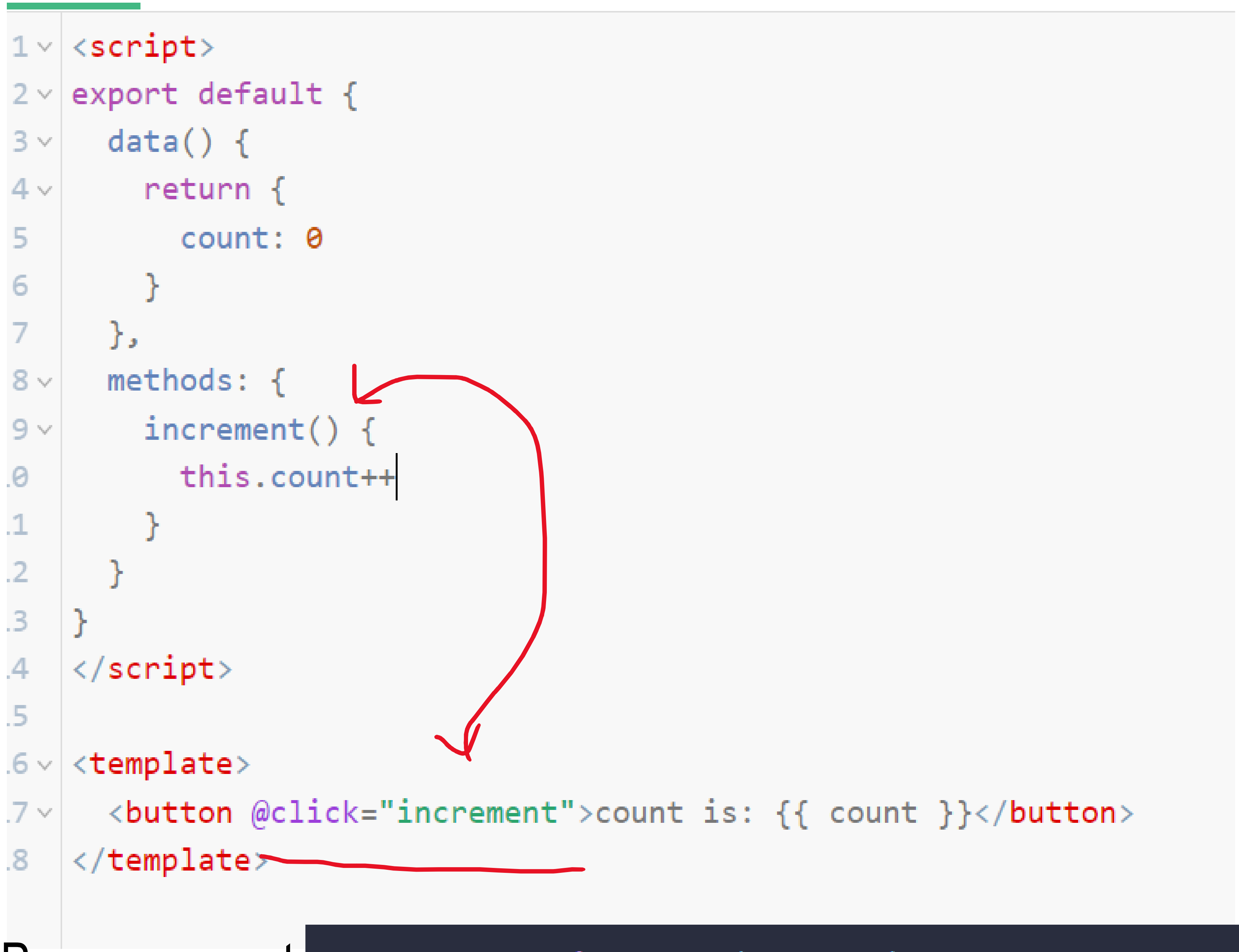
https://vuejs.org/tutorial/#step-3

Form binding (two-way binding in Angular)



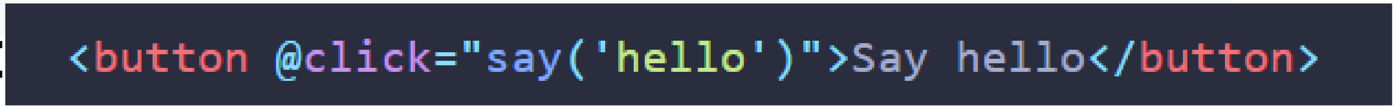
```
<script>
export default {
  data() {
    return {
      text: ''
    }
  }
}
</script>

<template>
  <input v-model="text" placeholder="Type here">
  <p>{{ text }}</p>
</template>
```

Event Listeners



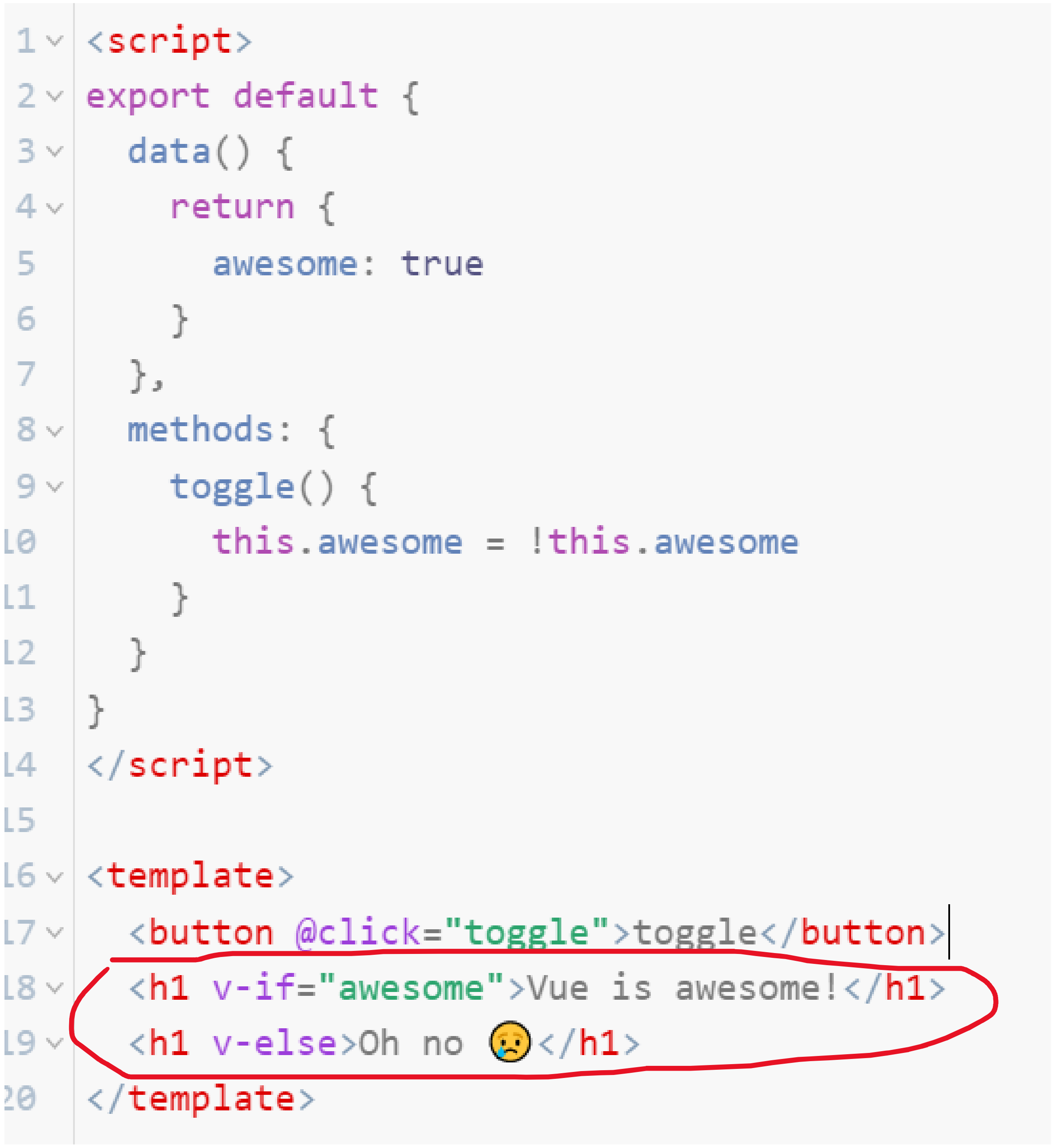
```
<script>
export default {
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  }
}
</script>

<template>
  <button @click="increment">count is: {{ count }}</button>
</template>
```

Pass argument:



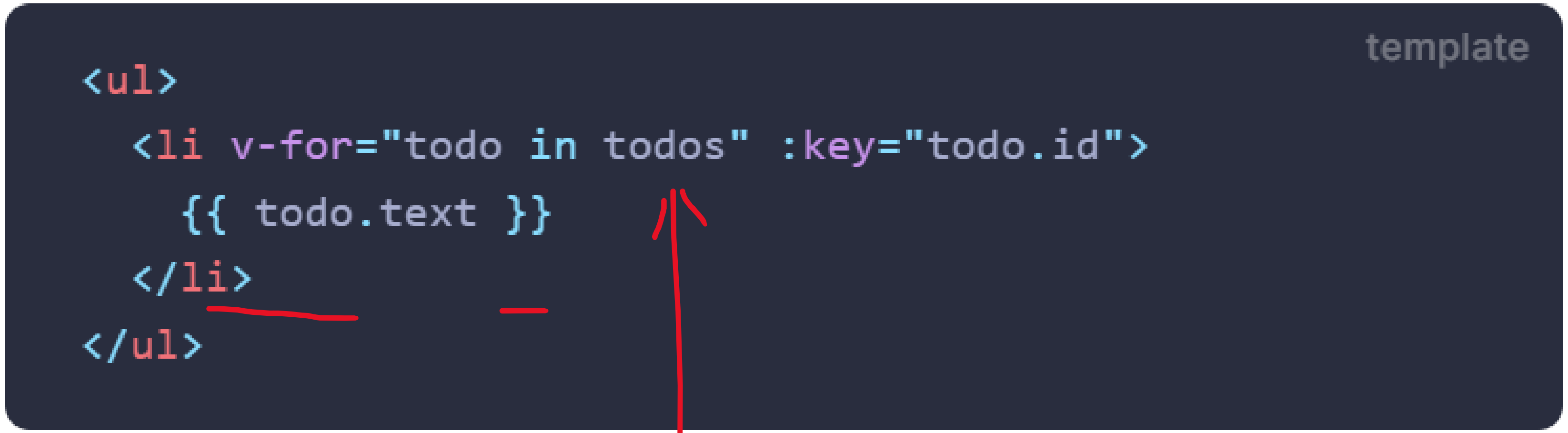
```
<button @click="say('hello')">Say hello</button>
```

Conditional rendering



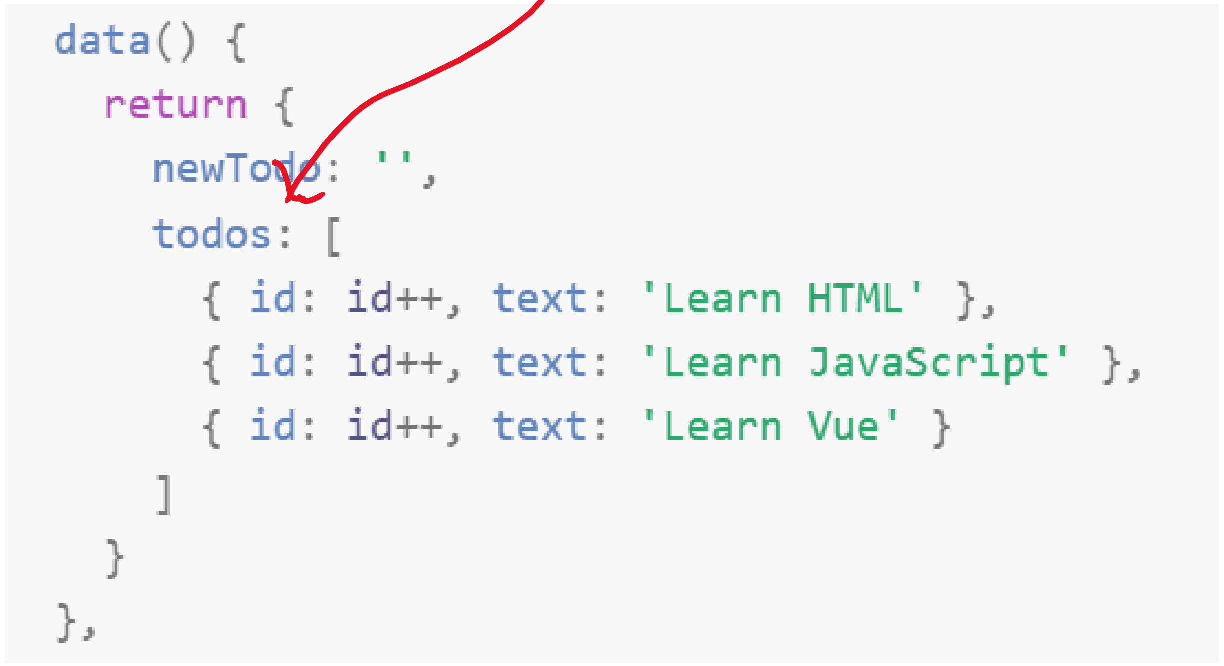
```
<script>
export default {
  data() {
    return {
      awesome: true
    }
  },
  methods: {
    toggle() {
      this.awesome = !this.awesome
    }
  }
}
</script>

<template>
  <button @click="toggle">toggle</button>
  <h1 v-if="awesome">Vue is awesome!</h1>
  <h1 v-else>Oh no 😢</h1>
</template>
```

Iterative rendering



template

```
<ul>
  <li v-for="todo in todos" :key="todo.id">
    {{ todo.text }}
  </li>
</ul>
```



```
data() {
  return {
    newTodo: '',
    todos: [
      { id: id++, text: 'Learn HTML' },
      { id: id++, text: 'Learn JavaScript' },
      { id: id++, text: 'Learn Vue' }
    ]
  }
},
```

Computed property

- Compute the property of an element:

The button label is updated using inline code instead of JS function

```
<button @click="hideCompleted = !hideCompleted">
  {{ hideCompleted ? 'Show all' : 'Hide completed' }}
</button>
```

Jquery

- **More like a library than a framework…**
- **Can simplify coding…**
- The jQuery code is stored as a single JavaScript file, containing all the jQuery methods.
- It can be added to a web page with the following mark-up:

```
<head>
  <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
</head>
```

Note: you can also download a version to your sever and include it as:

```
<script type="text/javascript" src="jquery.js"></script>
```

A simple example

```
<html>

  <head>
  <script type="text/javascript"
  src="jquery.js"></script>
  <script type="text/javascript">
  $(document).ready(function(){
    $("p").click(function(){
    $(this).hide();
    });
  });
  </script>
  </head>

   <body>

      <p>If you click on me, I will disappear.</p>

      <p><h1>This will not work</h1></p>

      <p>Me too</p>

   </body>

</html>
```

Use $ to select an element…

Another example: with Ajax

```
// file simpleajax.js
function getData(dataSource, aName, aPwd)  {
    $.ajax({
        type: "POST",
        url: dataSource,
        data: "name="+aName+"&pwd="+aPwd,
        success: function(msg){
            $("#response").html(msg);
        }
    });
}
```

'$' confusion: what is it?

- For 1st example, '$' is a function; it's a shorthand for the function 'jQuery()' defined in the lib
  - So `$("p").click(function(){`
    ```
    $(this).hide();
    })
    ```

  Stands for: invoke jQuery() function with argument `"p"` , the return of this function is an object, then we invoke the `click` function of this object with an argument, this argument is an anonymous function…

Actually we are attaching callback function hide() to all the "p" elements on the page…

‘$’ confusion: what is it?

- For 2nd example, ‘$’ becomes an object
  - So $.ajax(…)

  Stands for: invoke the ajax() function of the jQuery object
- In JavaScript, functions are a type of object. Specifically, functions are instances of the Function object which is derived from Object. jQuery takes advantage of that fact and hangs some "static" methods from the jQuery function object.

jQuery selector syntax

- **jQuery Syntax**
- The jQuery syntax is tailor made for **selecting** HTML elements and perform some **action** on the element(s).
- Basic syntax is: **`$(selector).action()`**
- A (selector) as parameter to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

- Examples:
  - `$(this).hide()`
  - `$("p").hide()`
  - `$("p.test").hide()` $\longrightarrow$ class selector
  - `$("#test").hide()` $\longrightarrow$ id selector

Note: selector uses CSS syntax.

jQuery document ready function

- **The Document Ready Function**
- Usually jQuery functions are inside a document.ready() function:

```
$(document).ready(function(){

   // jQuery functions go here...

});
```

- This is to prevent any jQuery code from running before the document is finished loading (is ready).

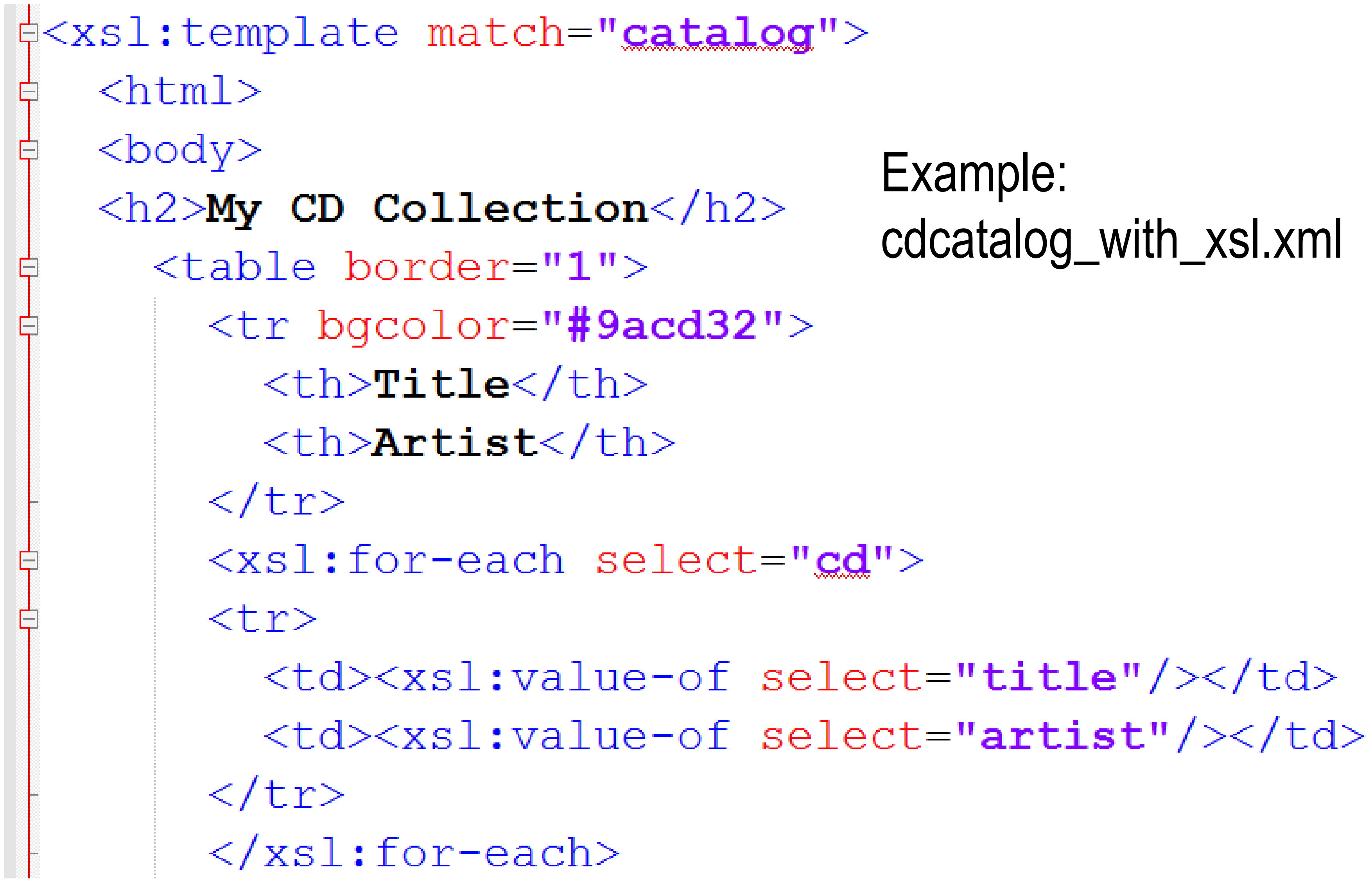
jQuery lib is rich…

- jQuery for animation
- jQuery for search box
- jQuery for RSS
- jQuery for drag and drop UI
- jQuery for form validation

XML rendering

- XSLT: data-oriented XML rending/transformation language
- XSLT: Extensible Stylesheet Langage Transformations
- We just use an example to demonstrate how to translate XML to HTML
- Angular template design borrowed a lot of ideas from XSLT…

The CD Catalog example



```
<xsl:template match="catalog">
  <html>
  <body>
  <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
```

Example:
cdcatalog_with_xsl.xml