Templates

- Templates are used to create a connection between the project's interface and the project's JavaScript code.
  - When we place interface elements inside a template, such as a button or a form, we're able to reference those elements and build an interface that users can interact with.

```
<head>
  <title>meteorApp12</title>
</head>
 <body>
  <h1>Header</h1>
{{> myParagraph}}
</body>
<template name = "myParagraph">
  <p>hello world</p>
</template>
```

Spacebars syntax, and Spacebars is the syntax in our HTML when we want something dynamic to occur.
It's the syntax that bridges the gap between the interface and the application logic.

- Meteor takes care of automatically adds the html tags to either end of the file and it automatically includes any resources contained within the project's folder – such as the JavaScript and CSS files.

Templates

- Meteor templates are using three top level tags.
- The first two are **head** and **body**. These tags perform the same functions as in regular HTML.
- The third tag is **template**. This is the place, where we connect HTML to JavaScript.

- **The Template keyword** searches through all of the templates in the Meteor project.
- **The myParagraph keyword** is a reference to the name of the template we created earlier.

Simple template

- We are creating a template with **name ="myParagraph"** attribute.
- Our **template** tag is created below the **body** element, however, we need to include it before it is rendered on the screen.
- We can do it by using **{{> myParagraph}}** syntax.
- In our template, we are using double curly braces **({{text}})**. This is meteor template language called **Spacebars**.
- ***helper function***→ JavaScript functions that are attached to templates and allow us to execute code from inside the interface.
- In our JavaScript file, we are setting **Template.myParagraph.helpers({})** method that will be our connection to our template.
- We are only using **text** helper in this example.

Simple Template

meteorApp.html

```
<head> <title>meteorApp</title> </head>
<body> <h1>Header</h1> {{> myParagraph}} </body>
<template name = "myParagraph">
        <p>{{text}}</p> </template>
```

meteorApp.js

```
if (Meteor.isClient) {  / / This code only runs on the client
Template.myParagraph.helpers({ text: 'This is paragraph…' }); }
```

- Following example shows how this works. We are creating a template with **name = "myParagraph"** attribute. Our **template** tag is created below the **body** element, however, we need to include it before it is rendered on the screen. We can do it by using **{{> myParagraph}}** syntax. In our template, we are using double curly braces **({{text}})**. This is meteor template language called **Spacebars**.
- In our JavaScript file, we are setting **Template.myParagraph.helpers({})** method that will be our connection to our template. We are only using **text**helper in this example.
- After we save the changes, following will be the output.

Meteor separates code that will be run on the server vs the client

- if (Meteor.isClient) {

/ / code here will only be run on the client }

- if (Meteor.isServer) {

/ / code here will only be run on the server }

- While we could build our whole application like this, with client and server code in the same file.
- **Meteor will automatically restart the server, rebundle our files, and livereload our browser (while keeping all our data)**

Block Template

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body> <div>
{{#each paragraphs}} {{> paragraph}} {{/each}} </div>
</body> <template name = "paragraph"> <p>{{text}}</p>
</template>
```

meteorApp.js

```
if (Meteor.isClient) { / / This code only runs on the client
Template.body.helpers({ paragraphs: [
{ text: "This is paragraph 1…" }, { text: "This is paragraph 2…" },
{ text: "This is paragraph 3…" }, { text: "This is paragraph 4…" },
{ text: "This is paragraph 5…" } ] }); }
```


meteorApp
localhost:3000
This is paragraph 1...
This is paragraph 2...
This is paragraph 3...
This is paragraph 4...
This is paragraph 5...

C:\Users\admin\Marees_Meteor\FirstApp\client\main.html

```
<head>  <title>FirstApp</title></head>
<body>  <h1>Welcome to Meteor!</h1>  {{> hello}}
{{> info}}</body>
<template name="hello">  <button>Click Me</button>  <p>You've pressed the button {{counter}} times.</p></template>
<template name="info"> <h2>Learn Meteor!</h2>  <ul>    <li><a href="https://www.meteor.com/try" target="_blank">Do the Tutorial</a></li>    <li><a href="http://guide.meteor.com" target="_blank">Follow the Guide</a></li>    <li><a href="https://docs.meteor.com" target="_blank">Read the Docs</a></li>    <li><a href="https://forums.meteor.com" target="_blank">Discussions</a></li>
</ul></template>
```

C:\Users\admin\Marees_Meteor\FirstApp\client\main.js

```
import { Template } from 'meteor/templating';
import { ReactiveVar } from 'meteor/reactive-var';
import './main.html';
Template.hello.onCreated(function helloOnCreated() {
 // counter starts at 0
this.counter = new ReactiveVar(0);});
Template.hello.helpers({  counter() {
return Template.instance().counter.get();  },});
Template.hello.events({  'click button'(event, instance) {
// increment the counter when button is clicked
instance.counter.set(instance.counter.get() + 1);  },});
```

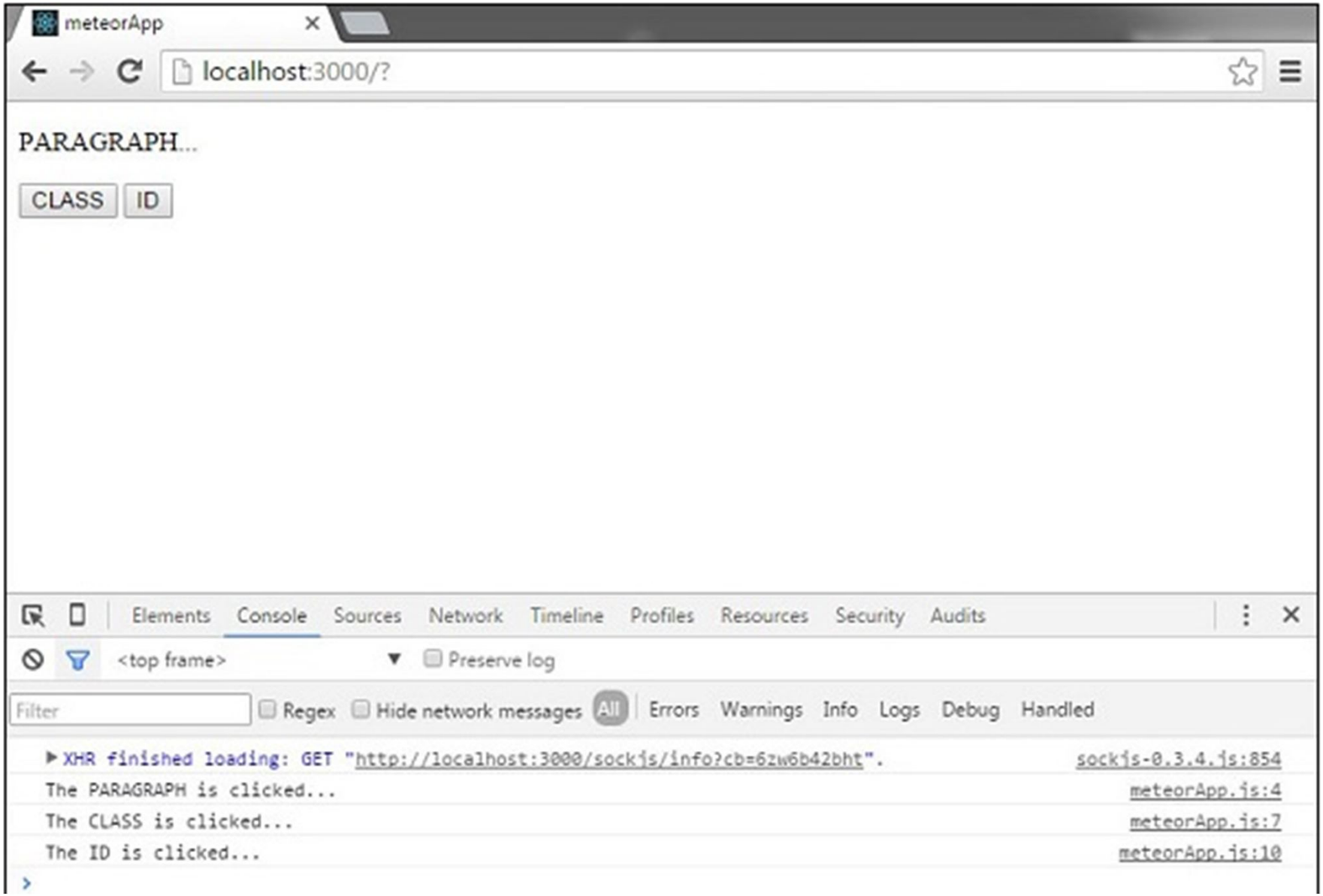
Event Handling

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body>
<div> {{> myTemplate}} </div> </body>
<template name = "myTemplate"> <p>PARAGRAPH…</p>
<button class = "myClass">CLASS</button>
<button id = "myId">ID</button> </template>
```

meteorApp.js

```
if (Meteor.isClient) {
 Template.myTemplate.events({
'click p': function() { console.log("The PARAGRAPH is clicked…"); },
'click .myClass': function() { console.log("The CLASS is clicked…"); },
'click #myId': function() { console.log("The ID is clicked…"); }, }); }
```


meteorApp
localhost:3000/?
PARAGRAPH...
CLASS
ID
Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
<top frame>
Preserve log
Filter
Regex
Hide network messages
All
Errors
Warnings
Info
Logs
Debug
Handled
XHR finished loading: GET "http://localhost:3000/sockjs/info?cb=6zw6b42bht". sockjs-0.3.4.js:854
The PARAGRAPH is clicked... meteorApp.js:4
The CLASS is clicked... meteorApp.js:7
The ID is clicked... meteorApp.js:10

Form handling

meteorApp.html

```
<head>   <title>meteorApp</title></head>
 <body>
  <div>
    {{> myTemplate}}
  </div>
</body>

<template name = "myTemplate">
  <form>
    <input type = "text" name = "myForm">
    <input type = "submit" value = "SUBMIT">
  </form>
</template>
```

meteorApp.js

```
if (Meteor.isClient) {
Template.myTemplate.events({
'submit form': function(event) {
 event.preventDefault();
 var textValue = event.target.myForm.value;
console.log(textValue);
event.target.myForm.value = "";      }   });}
```

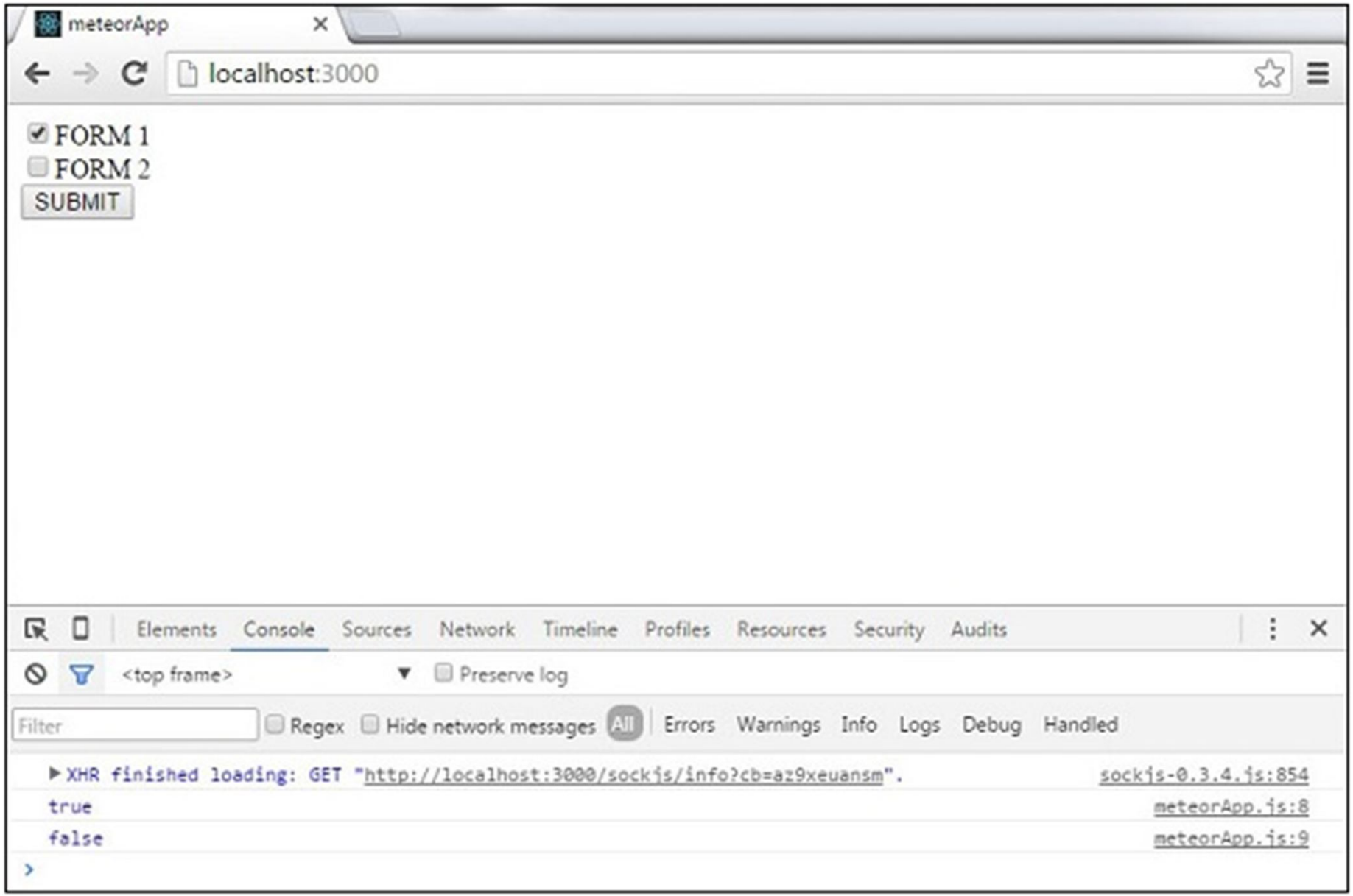
Form handling

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body>
<div> {{> myTemplate}} </div> </body>
<template name = "myTemplate"> <form>
<input type = "checkbox" name = "myForm" value = "form-1">FORM 1
<input type = "checkbox" name = "myForm" value = "form-2">FORM 2
<input type = "submit" value = "SUBMIT"> </form>
</template>
```

meteorApp.js

```
if (Meteor.isClient) {
Template.myTemplate.events({
'submit form': function(event) {
event.preventDefault();
var checkboxValue1 = event.target.myForm[0].checked;
var checkboxValue2 = event.target.myForm[1].checked;
console.log(checkboxValue1);
console.log(checkboxValue2); } }); }
```


meteorApp
localhost:3000
FORM 1
FORM 2
SUBMIT
Elements
Console
Sources
Network
Timeline
Profiles
Resources
Security
Audits
<top frame>
Preserve log
Filter
Regex
Hide network messages
All
Errors
Warnings
Info
Logs
Debug
Handled
XHR finished loading: GET "http://localhost:3000/sockjs/info?cb=az9xeuansm".
sockjs-0.3.4.js:854
true
meteorApp.js:8
false
meteorApp.js:9

Form handling

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body>
<div> {{> myTemplate}} </div> </body>
<template name = "myTemplate"> <form>
<template name = "myTemplate">
  <form>
    <input type = "radio" name = "myForm" value = "form-
1">FORM 1
    <input type = "radio" name = "myForm" value = "form-
2">FORM 2
    <input type = "submit" value = "SUBMIT">
  </form>
</template>
```

meteorApp.js

```
if (Meteor.isClient) {
Template.myTemplate.events({
'submit form': function(event) {
 event.preventDefault();
var radioValue = event.target.myForm.value;
console.log(radioValue);
  }   });
```

Form handling

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body>
<div> {{> myTemplate}} </div> </body>
<template name = "myTemplate"> <form>
<select>
    <option name = "myOption" value = "option-1">OPTION 1</option>
    <option name = "myOption" value = "option-2">OPTION 2</option>
    <option name = "myOption" value = "option-3">OPTION 3</option>
    <option name = "myOption" value = "option-4">OPTION 4</option>
  </select>
</template>
```

meteorApp.js

```
if (Meteor.isClient) {
Template.myTemplate.events({
'change select': function(event) {
event.preventDefault();
 var selectValue = event.target.value;
console.log(selectValue);
}
});
```

Sessions

- Sessions allow us to store small pieces of data that:
  - Are not saved to the database.
  - Will not be remembered on return visits.
- To begin using sessions, we need to first install a *package*, and in Meteor, a package is simply a plugin that can:

- > **meteor add session**

Session

- Sessions are used for saving data while the users are using the app. This data will be deleted when the user leaves the app.

meteorApp.html

```
<head> <title>meteorApp</title> </head> <body> <div>
{{> myTemplate}} </div> </body>
<template name = "myTemplate"> </template>
```

meteorApp.js

```
if (Meteor.isClient) {
var myData = { key1: "value1", key2: "value2" }
Session.set('mySession', myData);
var sessionDataToLog = Session.get('mySession');
console.log(sessionDataToLog); }
```