



</talentlabs>

Express Lecture 4

Server-side Rendering



</talentlabs>

Agenda

- Server-side vs Client-side rendering
- Server-side Rendering
- EJS Template Syntax
- Example Application

Server-side vs Client-side Rendering

</talentlabs>



Rendering (Rendering the UI)



Client-side Rendering

In the JSON Response section, we talked about a JSON Response is only returning the data. **We have some other frontend code to handle the rendering.**

Server-side Rendering

In the HTML Template section, you can also **define the UI with Express.**

Rendering (Rendering the UI)



Server-side rendering is easier, you only need backend developers to build a web application.

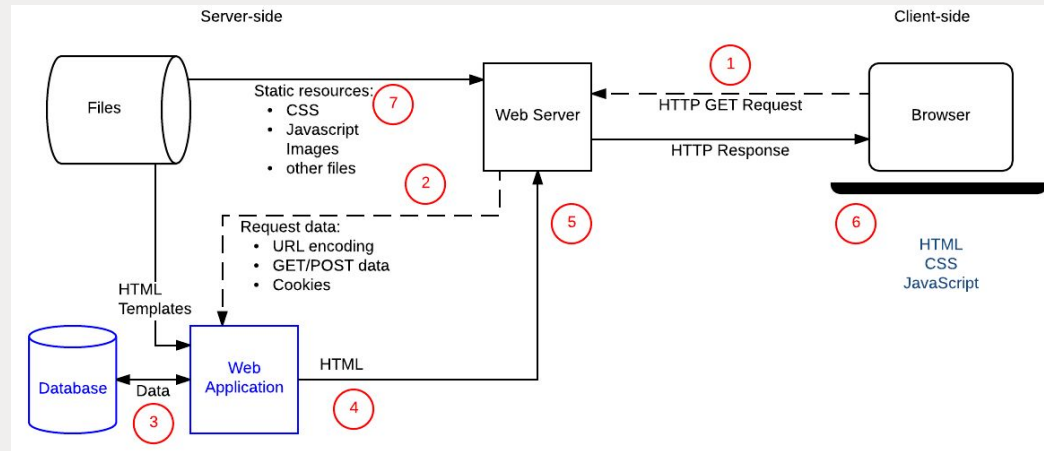
However, Client-side rendering is the trend nowadays. Because the clients are not just browser (reading HTML files). Many clients don't understand HTML file such as Mobile Apps.

```
/* GET home page. */  
router.get('/', function(req, res, next) {  
  res.render('index', {  
    title: 'Express App',  
    message: "I am here!!!!!!",  
  });  
});
```

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById('root')  
);
```

Server-side Rendering Overview

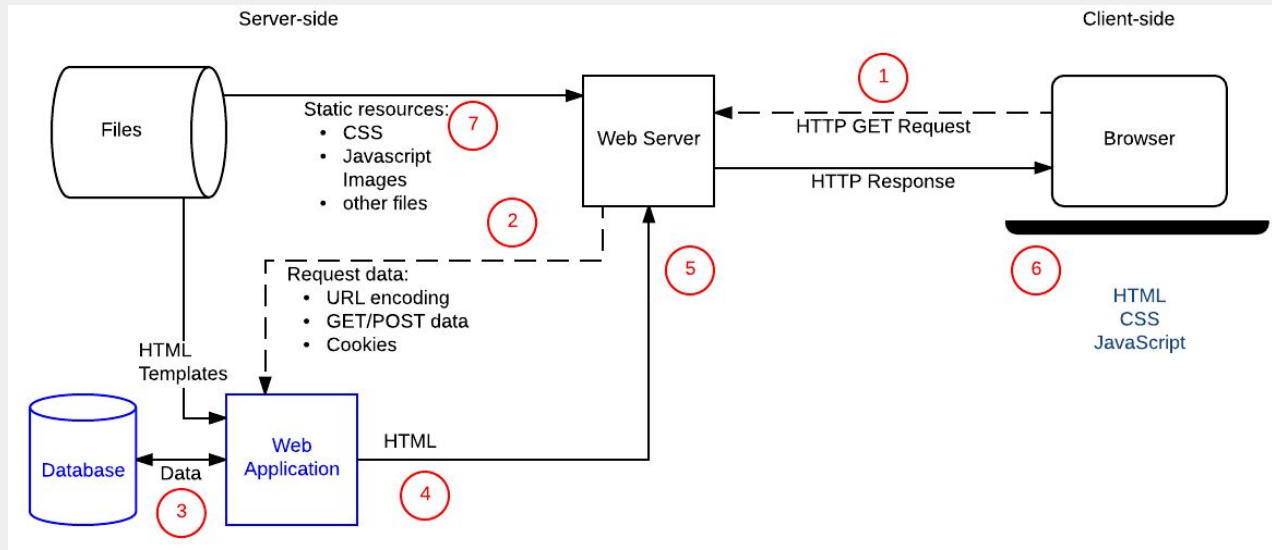
We will talk about this in the coming lectures



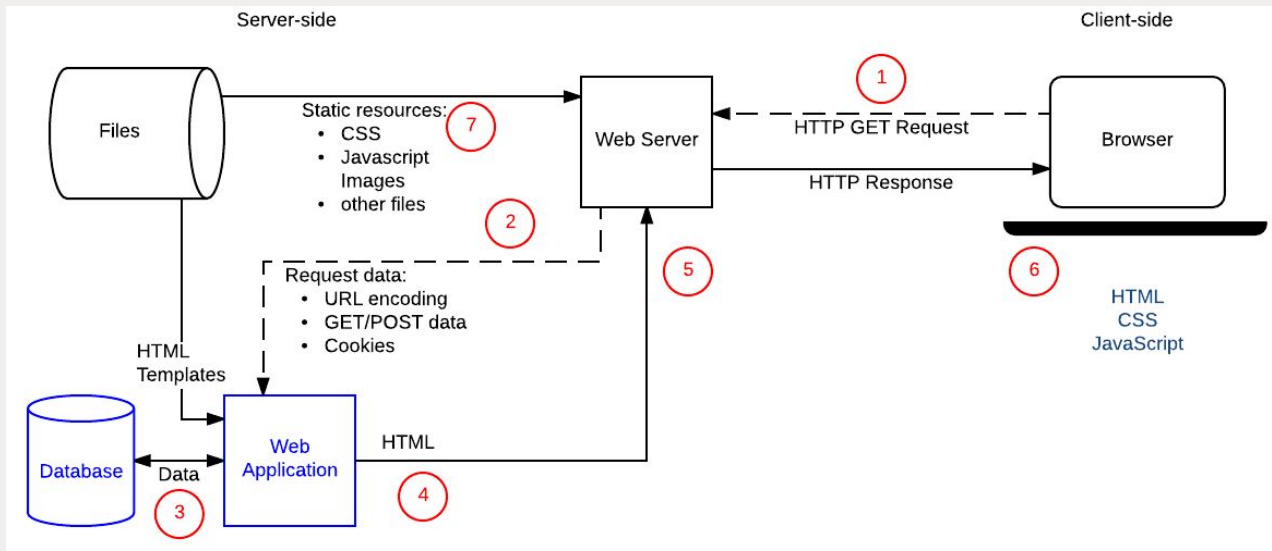
Server-side Rendering

</talentlabs>





1. Client sends a HTTP request to a Server.
2. The HTTP request is then handled by a Web Application (Express) in the server.
3. Now we need to prepare the HTML for the client!



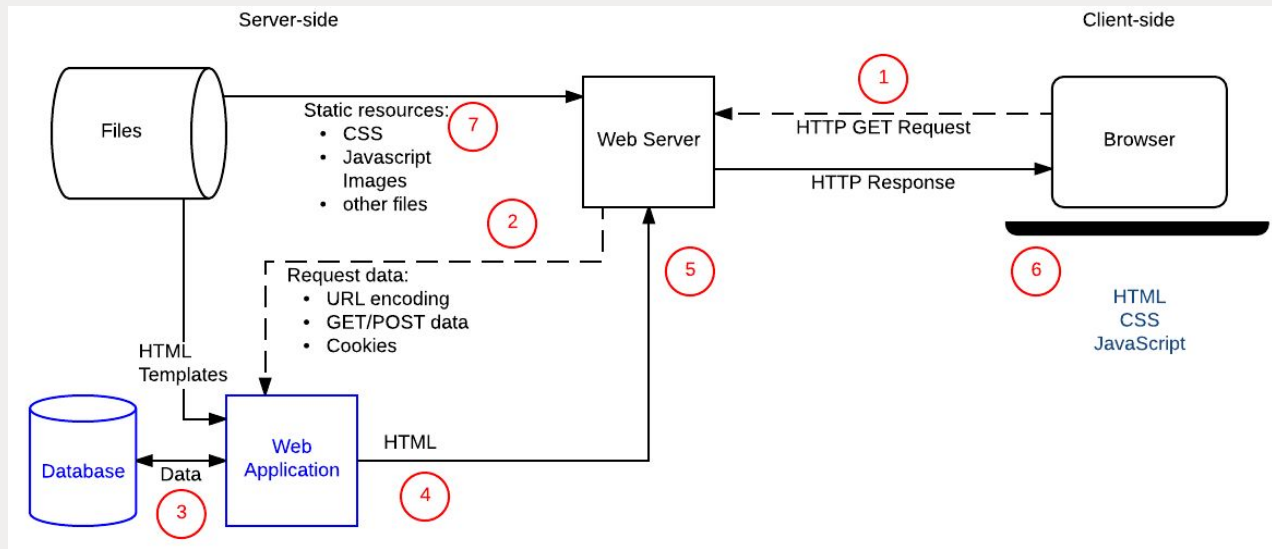
3. Now we need to prepare the HTML for the client!

3.1 Firstly, the web application find the corresponding **HTML template** for the request.

3.2 Optionally, the web application fetches data from a Database.
(we will talk about database later, for now imagine the data are just variables.)

3.3 Finally, the web application **fills in the data into the HTML templates!**

```
router.get("/", function (req, res, next) {  
  res.render("index",  
    { title: "Express" });  
});
```



4,5,6: the rendered HTML is then sent back the the client for display.

1, 7, 6: The client requests for additional static resources based on the HTML, such as images and CSS files.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Express</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Express</h1>
    <p>Welcome to Express</p>
    <p><%= message %></p>
  </body>
</html>
```

EJS Template Syntax

`</talentlabs>`




Render a Placeholder

Given this rendering context in a route:

We can replace a value into a EJS placeholder like this:

```
router.get('/placeholder', function(req,
res, next) {
  res.render('placeholder', {
    title: "My Title",
  });
});
```

```
<html>
  <h1>
    <%= title %>
  </h1>
</html>
```



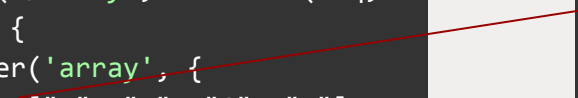
Render an Array

Given this rendering context:

We can render each item in the myArray like this:

```
router.get('/array', function(req,
res, next) {
  res.render('array', {
    myArray: ["A", "B", "C", "D"],
  });
});
```

```
<% myArray.forEach(function(item){ %>
  <div>
    <%= item %>
  </div>
<% }); %>
```



Render an Array of Objects

Given this rendering context:

We can render each item in the myArray like this:

```
router.get('/arrayofobjects',  
function(req, res, next) {  
  res.render('arrayofobjects', {  
    myArray: [  
      {name: "A"},  
      {name: "B"},  
      {name: "C"},  
      {name: "D"},  
    ],  
  });  
});
```

```
<% myArray.forEach(function(item){ %>  
  <div>  
    <%= item.name %>  
  </div>  
<% }); %>
```

Render Conditionally

Given this rendering context:

We can render each item **conditionally** in the myArray like this:

```
router.get('/condition', function(req, res,
next) {
  res.render('condition', {
    myArray: [
      {name: "A", show: true},
      {name: "B", show: true},
      {name: "C", show: false},
      {name: "D", show: true},
    ],
  });
});
```

```
<% myArray.forEach(function(item){ %>
  <% if (item.show) { %>
    <div>
      <%= item.name %>
    </div>
  <% } %>
<% }); %>
```

Summary:

`<% %>`: Wrapping an **JavaScript Operation** like if-statement, for-loop.

`<%= %>`: Wrapping a **JavaScript Value for display**.

```
router.get('/condition', function(req, res,
next) {
  res.render('condition', {
    myArray: [
      {name: "A", show: true},
      {name: "B", show: true},
      {name: "C", show: false},
      {name: "D", show: true},
    ],
  });
});
```

```
<% myArray.forEach(function(item){ %>
  <% if (item.show) { %>
    <div>
      <%= item.name %>
    </div>
  <% } %>
<% }); %>
```


Example Application

</talentlabs>



Example Data Model

Field Name	Type	Description
id	string	An identifier that uniquely identifies a product.
productName	string	The title of the product.
price	number	The price of the product.
active	boolean	If this product is visible to the public.

```
[
  {
    id: "9aa711b367533c012cb110d7ebca844b",
    productName: "Preiser 30411 Horse Drawn Box Wagon",
    price: 137.16,
    active: true,
  },
  {
    id: "d160b9ad8d9c7c7ed1c207a77e9934cc",
    productName: "Roco 10908 Seuthe Smoke Oil",
    price: 6.5,
    active: true,
  },
  {
    id: "f02b2132fcb3460565b5de94c05be8a8",
    productName:
      "Subway train 500 form three-car basic set red gauge N 10-1134
      Marunouchi Line",
    price: 45.75,
    active: false,
  },
  {
    id: "bb2466d62f3fdc9510272ee5233694d3",
    productName: "Walthers Trainline 931-1676 TL 50' PD Boxcar ATSF",
    price: 15.0,
    active: true,
  },
];
```

Hard Code the values
in the code (route/index.js):

```
const products = [
  {
    id: "9aa711b367533c012cb110d7ebca844b",
    productName: "Preiser 30411 Horse Drawn Box Wagon",
    price: 137.16,
    active: true,
  },
  {
    id: "d160b9ad8d9c7c7ed1c207a77e9934cc",
    productName: "Roco 10908 Seuthe Smoke Oil",
    price: 6.5,
    active: true,
  },
  {
    id: "f02b2132fcb3460565b5de94c05be8a8",
    productName:
      "Subway train 500 form three-car basic set red gauge N 10-1134 Marunouchi Line",
    price: 45.75,
    active: false,
  },
  {
    id: "bb2466d62f3fdc9510272ee5233694d3",
    productName: "Walthers Trainline 931-1676 TL 50' PD Boxcar ATSF",
    price: 15.0,
    active: true,
  },
];

router.get("/products", function (req, res, next) {
  res.render("products", {
    products: products,
  });
});
```

Template

```
<html>
<header>
  <!-- paste bootstrap css here -->
</header>

<body>
</body>
<table class="table">
  <thead>
    <th>Id</th>
    <th>Name</th>
    <th>Price</th>
    <th>Active</th>
  </thead>
  <tbody>
    <% products.forEach(function(product){ %>
      <% if (product.active) { %>
        <tr>
          <td><%= product.id%></td>
          <td><%= product.productName%></td>
          <td><%= product.price %></td>
          <td><%= product.active %></td>
        </tr>
      <% } %>
    <% }>; %>
  </tbody>
</table>
</html>
```

```
const products = [
  {
    id: "9aa711b367533c012cb110d7ebca844b",
    productName: "Preiser 30411 Horse Drawn Box Wagon",
    price: 137.16,
    active: true,
  },
  {
    id: "d160b9ad8d9c7c7ed1c207a77e9934cc",
    productName: "Roco 10908 Seuthe Smoke Oil",
    price: 6.5,
    active: true,
  },
  {
    id: "f02b2132fcb3460565b5de94c05be8a8",
    productName:
      "Subway train 500 form three-car basic set red gauge N 10-1134 Marunouchi
      Line",
    price: 45.75,
    active: false,
  },
  {
    id: "bb2466d62f3fdc9510272ee5233694d3",
    productName: "Walthers Trainline 931-1676 TL 50' PD Boxcar ATSF",
    price: 15.0,
    active: true,
  },
];

router.get("/products", function (req, res, next) {
  res.render("products", {
    products: products,
  });
});
```