Chapter-7

Common client-side web technologies

ASP.NET Core applications are web applications and they typically rely on client-side web technologies like **HTML**, **CSS**, and **JavaScript**

HTML is the standard markup language used to create web pages and web applications. Its elements form the building blocks of pages, representing formatted text, images, form inputs, and other structures. When a browser makes a request to a URL, whether fetching a page or an application, the first thing that is returned is an HTML document. This HTML document may reference or include additional information about its look and layout in the form of CSS, or behavior in the form of JavaScript.

Example:

```
<html>
  <head>
    <title>My web page</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <p>This is my first web page.</p>
    <p>It contains a
      <strong>main heading</strong> and <em> paragraph </em>.
    </p>
  </body>
</html>
```

CSS (Cascading Style Sheets) is used to control the look and layout of HTML elements. CSS styles can be applied directly to an HTML element, defined separately on the same page, or defined in a separate file and referenced by the page. Styles cascade based on how they are used to select a given HTML element. For instance, a style might apply to an entire document, but would be overridden by a style that applied to a particular element. Likewise, an element-specific style would be overridden by a style that applied to a CSS class that was applied to the element, which in turn would be overridden by a style targeting a specific instance of that element (via its ID). There are three ways to insert CSS in HTML Documents.

1. **Inline CSS:** It is used to apply CSS on a Single line or element.
   Eg: Welcome CSS
2. **Internal CSS :** It is used to apply CSS on a single document or page. It can affect all the elements of the page. It is written inside the style tag within head section of html.
   Eg: body{ background-color:yellow;}
   H1{color:pink;}

3. **External CSS:** It is used to apply CSS on multiple pages or all pages. It can be written in any text editor. The file should not contain any html tags. The file must save with .css extension.

Example:

```
<html>
  <head>
    <style>
      p {
        color: #FF7A59;
      }
    </style>
  </head>
  <body>

    <h2>Internal CSS Example</h2>

      <p>The default text color for the page is black. However I can change the color of every paragraph element on the page using internal CSS.</p>

      <p>Using internal CSS, I only need to write one rule set to change the color of every paragraph elemnet.</p>

      <p>With inline CSS, I'd have to add a style attribute to every single paragraph in my HTML.</p>

  </body>

</html>
```

Another example:

```
<html>
    <head>
      <link rel="stylesheet" type="text/css"  href="mystyle.css">
    </head>
    <body>
      <div>
        <h1>External CSS Example</h1>
        <p>In the external stylesheet, the div is styled to have a background color, text color, border, and padding.</p>
      </div>
    </body>
</html>
```

mystyle.css

```
body {
```

```
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

JavaScript is a dynamic, interpreted programming language that has been standardized in the ECMAScript language specification. It is the programming language of the web. Like CSS, JavaScript can be defined as attributes within HTML elements, as blocks of script within a page, or in separate files. Just like CSS, it's recommended to organize JavaScript into separate files, keeping it separated as much as possible from the HTML found on individual web pages or application views.

When working with JavaScript in your web application, there are a few tasks that you'll commonly need to perform:

1. Selecting an HTML element and retrieving and/or updating its value.
2. Querying a Web API for data.
3. Sending a command to a Web API (and responding to a callback with its result).
4. Performing validation.

You can perform all of these tasks with JavaScript alone, but many libraries exist to make these tasks easier. One of the first and most successful of these libraries are jQuery, which continues to be a popular choice for simplifying these tasks on web pages. For Single Page Applications (SPAs), jQuery doesn't provide many of the desired features that Angular and React offer.

Although ancient by JavaScript framework standards, jQuery continues to be a commonly used library for working with HTML/CSS and building applications that make AJAX calls to web APIs. However, jQuery operates at the level of the browser document object model (DOM), and by default offers only an imperative, rather than declarative, model.

For example, imagine that if a textbox's value exceeds 10, an element on the page should be made visible. In jQuery, this functionality would typically be implemented by writing an event handler with code that would inspect the textbox's value and set the visibility of the target element based on that value. This process is an imperative, code-based approach. Another framework might instead use databinding to bind the visibility of the element to the value of the textbox declaratively. This approach would not require writing any code, but instead only requires decorating the elements involved with data binding attributes.

As client-side behaviors grow more complex, data binding approaches frequently result in simpler solutions with less code and conditional complexity.

Example:

```
<html>
<title>valideting</title>
<head>
<script language="Javascript">
function valid()
{
        if(document.f1.t1.value=="")
        {
                alert("Please Enter the Name");
                document.f1.t3.focus();
                return false;
        }
        if(document.f1.t2.value=="")
        {
                alert("Please Enter the Address");
                document.f1.t3.focus();
                return false;
        }
                return true;
}
</script>
</head>
<body>
        <form name="f1" action="validetion.html" method="post"
                onsubmit="return valid();">
                <table border="1" align="center">
                        <tr>
                                <td>Name:</td>
                                <td><input type="text" name="name">
                                </td>
                        </tr>
                        <tr>
                                <td>Address:</td>
                                <td><input type="text" name="address">
                                </td>
                        <tr>
                                <td><input type="Submit" value="Send" onclick="valid()">
                                </td>
                        </tr>
                </table>
        </form>
</body>
</thtml>
```

JQuery:

- jQuery is a small and lightweight JavaScript library.
- jQuery is cross-platform.
- jQuery is easy to learn.

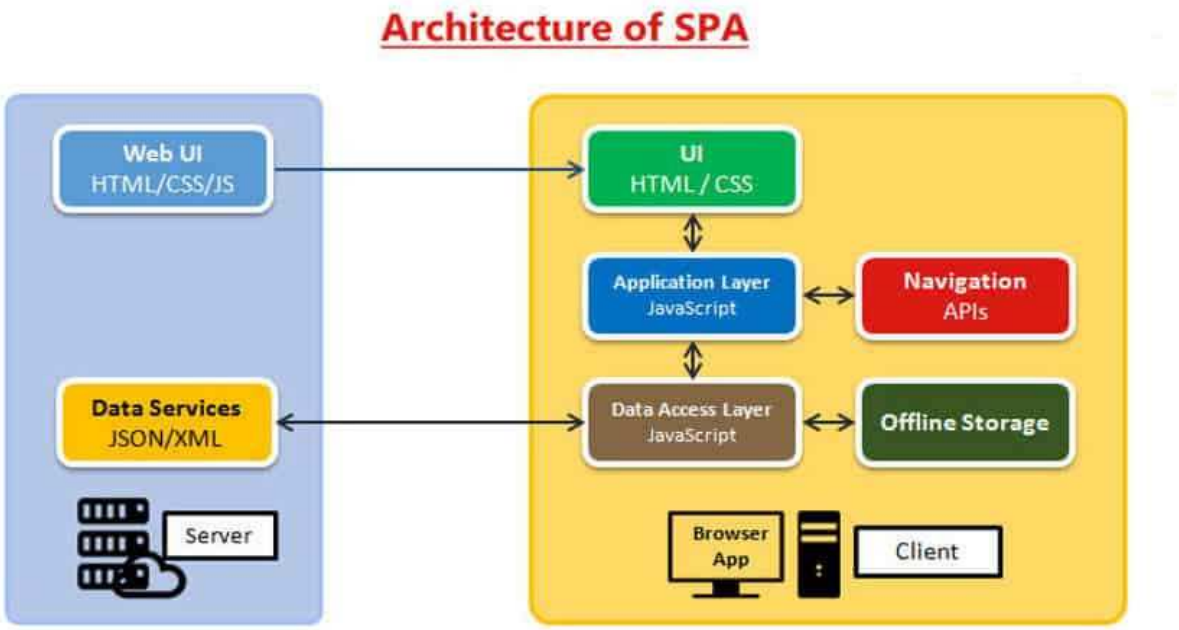
Example:

```
@{
    Layout = null;
}
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<style>
    form label {
        display: inline-block;
        width: 100px;
    }

    form div {
        margin-bottom: 10px;
    }

    .error {
        color: red;
        margin-left: 5px;
    }

    label.error {
        display: inline;
    }
</style>
<h2>Example Of Jquery Form Validation</h2>
<form id="first_form" method="post" action="">
    <div>
        <label for="first_name">First Name:</label>
        <input type="text" id="first_name" name="first_name"/>
    </div>
    <div>
        <label for="last_name">Last Name:</label>
        <input type="text" id="last_name" name="last_name"/>
    </div>
    <div>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email"/>
    </div>
    <div>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password"/>
    </div>
    <div>
```

```
        <input type="submit" value="Submit" />
    </div>
</form>

<script>
    $(document).ready(function () {

        $('#first_form').submit(function (e) {
            e.preventDefault();
            var first_name = $('#first_name').val();
            var last_name = $('#last_name').val();
            var email = $('#email').val();
            var password = $('#password').val();

            $(".error").remove();

            if (first_name.length < 1) {
                $('#first_name').after('<span class="error">This field is required</span>');
            }
            if (last_name.length < 1) {
                $('#last_name').after('<span class="error">This field is required</span>');
            }
            if (email.length < 1) {
                $('#email').after('<span class="error">This field is required</span>');
            }
            if (password.length < 8) {
                $('#password').after('<span class="error">Password must be at least 8 characters long</span>');
            }
        });

    });
</script>
```

What is Single Page Application? Draw architecture of SPA and explain its features.

SPA is a web application taking a **single HTML page**. Delivering dynamic updates, it allows interacting with the **page without refreshing** it. Making use of a single page app you can significantly decrease the server load and enhance **loading speed for a better UX**. This is possible because SPA only boots data on demand while the entire page has previously been loaded.



Being quicker than traditional multi-page websites, a single page app may be the best solution for business. Let's consider SPA features to figure out what it is good for.

SPA FEATURES

Some of the core features most competitive single page apps have are as follows:

- **No server roundtrip.** SPA can easily reshape any part of UI to restore the full HTML page preventing server roundtrip. This can be achieved by using Separation of Concerns (SoC) design principle that sets a goal for dividing the software system into parts so that each part could tackle a separate problem.
- **Templating and routing on the client side.** Supported by JS-based routers, SPA can monitor the user's actual state and location within the whole navigation experience.
- **The capability to work offline.** A single page app can still work offline in the event of a lost internet connection, for example. Once the connection is restored, the local data is synched with the web server.
- **Cross-platform functionality.** SPAs work on different operating systems from any browser. This feature is more advantageous for developers, however, a regular user working with numerous devices can also greatly benefit from it.
- **A reduced throughput.** The connection with the server is carried out in small fragments. Data is packed into smaller objects (e.g. **JSON data format**) which is especially helpful in the case of slow or traffic-limited connection.

These and other features significantly influence single page application performance hence making it more attractive for startups and other online projects. Single page application architecture is impeccable for dynamic apps, SaaS platforms, social media, and more.

EXAMPLES OF SPAS

Angular and React.

jQuery vs a SPA Framework

Factor	jQuery	Angular
Abstracts the DOM	Yes	Yes
AJAX Support	Yes	Yes
Declarative Data Binding	No	Yes
MVC-style Routing	No	Yes
Templating	No	Yes
Deep-Link Routing	No	Yes

Most of the features jQuery lacks intrinsically can be added with the addition of other libraries. However, a SPA framework like Angular provides these features in a more integrated fashion, since it's been designed with all of them in mind from the start. Also, jQuery is an imperative library, meaning that you need to call jQuery functions in order to do anything with jQuery. Much of the work and functionality that SPA frameworks provide can be done declaratively, requiring no actual code to be written.

Data binding is a great example of this functionality. In jQuery, it usually only takes one line of code to get the value of a DOM element or to set an element's value. However, you have to write this code anytime you need to change the value of the element, and sometimes this will occur in multiple functions on a page. Another common example is element visibility. In jQuery, there might be many different places where you'd write code to control whether certain elements were visible. In each of these cases, when using data binding, no code would need to be written. You'd simply bind the value or visibility of the elements in question to a viewmodel on the page, and changes to that viewmodel would automatically be reflected in the bound elements.

Angular SPAs

Angular remains one of the world's most popular JavaScript frameworks. Since Angular 2, the team rebuilt the framework from the ground up (using TypeScript) and rebranded from the original AngularJS name to

angular. Now several years old, the redesigned Angular continues to be a robust framework for building Single Page Applications.

Angular applications are built from components. Components combine HTML templates with special objects and control a portion of the page. A simple component from Angular's docs is shown here:

```
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<h1>Hello {{name}}</h1>`
})

export class AppComponent { name = 'Angular'; }
```

Components are defined using the @Component decorator function, which takes in metadata about the component. The selector property identifies the ID of the element on the page where this component will be displayed. The template property is a simple HTML template that includes a placeholder that corresponds to the component's name property, defined on the last line.

You can develop Angular apps by using a CLI. Getting started with Angular development locally (assuming you already have git and npm installed) consists of simply cloning a repo from GitHub and running **npm install** and **npm start**. Beyond this, Angular ships its own CLI, which can create projects, add files, and assist with testing, bundling, and deployment tasks. This CLI friendliness makes Angular especially compatible with ASP.NET Core, which also features great CLI support.

React

Unlike Angular, which offers a full Model-View-Controller pattern implementation, React is only concerned with views. It's not a framework, just a library, so to build a SPA you'll need to leverage additional libraries. There are a number of libraries that are designed to be used with React to produce rich single page applications.

One of React's most important features is its use of a virtual DOM. The virtual DOM provides React with several advantages, including performance (the virtual DOM can optimize which parts of the actual DOM need to be updated) and testability (no need to have a browser to test React and its interactions with its virtual DOM).

React is also unusual in how it works with HTML. Rather than having a strict separation between code and markup (with references to JavaScript appearing in HTML attributes perhaps), React adds HTML directly

within its JavaScript code as JSX. JSX is HTML-like syntax that can compile down to pure JavaScript. For example:

```
<ul>
{ authors.map(author =>
    <li key={author.id}>{author.name}</li>
)}
</ul>
```

If you already know JavaScript, learning React should be easy. There isn't nearly as much learning curve or special syntax involved as with Angular or other popular libraries.

Because React isn't a full framework, you'll typically want other libraries to handle things like routing, web API calls, and dependency management. The nice thing is, you can pick the best library for each of these, but the disadvantage is that you need to make all of these decisions and verify all of your chosen libraries work well together when you're done. If you want a good starting point, you can use a starter kit like React Slingshot, which prepackages a set of compatible libraries together with React.

Choosing a SPA Framework

When considering which option will work best to support your SPA, keep in mind the following considerations:

1. Is your team familiar with the framework and its dependencies (including TypeScript in some cases)?
2. How opinionated is the framework, and do you agree with its default way of doing things?
3. Does it (or a companion library) include all of the features your app requires?
4. Is it well documented?
5. How active is its community? Are new projects being built with it?
6. How active is its core team? Are issues being resolved and new versions shipped regularly?

Frameworks continue to evolve with breakneck speed. Use the considerations listed above to help mitigate the risk of choosing a framework you'll later regret being dependent upon. If you're particularly risk-averse, consider a framework that offers commercial support and/or is being developed by a large enterprise.

Compare SPA Framework over jQuery.

jQuery is a library.	Angular is a framework.
If you want to build a web site, jQuery is enough without Angular.	If you want to build web application, need to go with Angular.
Doesn't supports two-way data binding.	Supports two-way data binding.
It doesn't uses Dependency Injection (DI)	Much use of Dependency Injection (DI)
It can be used on any website or web application.	It is mostly used for Single Page Applications (SPA)
Unit Test not possible	Unit testing is possible

jQuery vs a SPA Framework

Factor	jQuery	Angular
Abstracts the DOM	**Yes**	**Yes**
AJAX Support	**Yes**	**Yes**
Declarative Data Binding	**No**	**Yes**
MVC-style Routing	**No**	**Yes**
Templating	**No**	**Yes**
Deep-Link Routing	**No**	**Yes**

Compare angular and react

Parameters	**React**	**Angular**
Type	React is a JavaScript library, and it is much older compared with Angular.	Angular is a complete framework.
Use of libraries	React js can be packaged with other programming libraries.	Angular is a complete solution in itself.
Learning curve	It is easier to grasp compared Angular. However, it is difficult to learn when augmented with Redux.	Learning Angular is not easy for beginners. Thus, it requires lots of training.
Community support	When it comes to community support React doesn't offer much.	It has a viable and dependable community support system
Installation time	React takes longer to set up. But, it is really fast for delivering projects and building apps.	Angular is easy to set up but may lead to an increase in coding time which also results in delayed project deliveries.
Best feature	It gives you the freedom to choose the tools, architecture, and libraries, for developing an app.	It offers a limited amount of freedom and flexibility.
Data binding	React language uses one-way data binding, which means that the Ul elements can't be changed without updating the corresponding model state.	Angular, on the other hand, uses the two-way data binding method. It helps you to ensure that the model state

Parameters	React	Angular
		automatically changes when any change is made.
Testing & Debugging	It requires a set of tools to perform different types of testing.	The testing and debugging for a complete project is possible with a single tool.
Documentation	Although it is also undergoing regular updates, the documentation is relatively faster.	Due to the ongoing development process, the documentation is slower.
Updates	Updates in React are simple because scripts help in the migration.	It plans updates every six months, which gives some time to make needed changes for migration.
Application Types	Use this app if you want to develop Native apps, hybrid apps, or web apps	You should use this framework If you want to develop a SPA(Single Page Application) and mobile apps.
Ideal for	Ideal for modern web development and native- rendered apps for Android and iOS devices.	Ideal to use when you want to develop large-scale, feature-rich applications.
Model	It is based on Virtual DOM	Based on MVC (Model View Controller)
Written in	JavaScript	Typescript
Community Support	Facebook developers community	A large community of developers and supporters
Language preference	JSX - JavaScript XML	TypeScript
Companies Using	Facebook, Uber Technologies, Instagram, Netflix, Pinterest, etc.	Wepay, Beam, Auto Trader, Mesh, Streamline Social, etc.
Template	JSX + J% (ES5/ES6)	HTML + TypeScript
Git hub stars	180K	80.8 K
Fork	30.3 K	48.2 K