EmberJS Requisites

People matter, results count.

Table of Contents

- **Following topics will be covered**
  - ES6 features
  - NodeJS
  - NPM
  - Typescript
  - Promises
  - Babel
  - SaSS
  - Polyfill
  - Ajax
  - Jquery

ES6 Features

- Arrows Functions
- classes
- enhanced object literals
- template strings
- destructuring
- default + rest + spread Operators
- let + const
- iterators + for
- generators
- unicode
- modules
- module loaders
- map + set + weakmap + weakset
- proxies
- symbols
- subclassable built-ins
- binary and octal literals
- reflect api
- tail calls
- math + number + string + array
- Object APIs
- Promises

NodeJs

- Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine).
- It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications.
- Node.js is open source, completely free, and used by thousands of developers around the world.

Node.js = Runtime Environment + JavaScript Library

NPM

- **npm** is the default package manager for the JavaScript runtime environment Node.js.
- npm is automatically included when Node.js is installed.
- npm consists of a command line client that interacts with a remote registry.

Node Package Manager provides two main functionalities:

- It provides online repositories for node.js packages/modules which are searchable on search.nodejs.org
- It also provides command line utility to install Node.js packages, do version management and dependency management of Node.js packages.
- As an alternative, we can also use bower as a package manager.

Below is the command to install bower.

- **npm install -g bower**

TypeScript

- TypeScript is an extension (a “superset”) of the JavaScript language.
- It differentiates itself from competitors like Coffee Script and Dart in that plain JavaScript code can be intermixed with Typescript.
- Typescript provides following additional features to ECMA5:
  - Type annotations and compile-time type checking
  - Type inference
  - Type erasure
  - Interfaces
  - Enumerated type
  - Mixin
  - Generic
  - Namespaces
  - Tuple
  - Await

Promises

- The XMLHttpRequest API is async but does not use the Promises API.
- There are a few native APIs that now use promises, however:
- Battery API
- fetch API (XHR's replacement)
- ServiceWorker API

- Promises will only become more prevalent so it's important that all front-end developers get used to them. It's also worth noting that Node.js is another platform for Promises (obviously, as Promise is a core language feature).

Promises(contd…)

- A Promise object represents a value that may not be available yet, but will be resolved at some point in the future.
- It allows you to write asynchronous code in a more synchronous fashion.
- For example, if you use the promise API to make an asynchronous call to a remote web service you will create a Promise object which represents the data that will be returned by the web service in future.
- The caveat being that the actual data is not available yet. It will become available when the request completes and a response comes back from the web service.
- In the meantime the Promise object acts like a proxy to the actual data. Furthermore, you can attach callbacks to the Promise object which will be called once the actual data is available.

Promises-Browser Support

```
if (window.Promise)
{

 var promise = new
Promise(function(resolve, reject)
  {
 //asynchronous code goes here
   }
                                  );
 }
```

- We start by instantiating a new Promise object and passing it a callback function.
- The callback takes two arguments, resolve and reject, which are both functions.
- All your asynchronous code goes inside that callback.
- If everything is successful, the promise is fulfilled by calling resolve().
- In case of an error, reject() is called with an Error object.
- This indicates that the promise is rejected.

Promises-Browser Support

```
if (window.Promise)
{
console.log('Promise found');
 var promise = new Promise(function(resolve, reject)
{ var request = new XMLHttpRequest();
request.open('GET', 'http://api.icndb.com/jokes/random');
request.onload = function()
{ if (request.status == 200)
{ resolve(request.response); // we got data here, so resolve the Promise
 } else { reject(Error(request.statusText)); // status is not 200 OK, so reject
} };
 request.onerror = function()
{ reject(Error('Error fetching data.')); // error occurred, reject the Promise
 };
request.send(); //send the request });
console.log('Asynchronous request made.');
promise.then(function(data) {
console.log('Got data! Promise fulfilled.');
document.getElementsByTagName('body')[0].textContent = JSON.parse(data).value.joke; },
function(error) {
console.log('Promise rejected.');
console.log(error.message); });
 } else
{ console.log('Promise not available'); }
```

ES6 Modules

- An amazing omission in Javascript's design is the lack of a built-in module system.
- As more projects used Javascript and shared more code, the need for a robust module system became necessary.
- Two contenders sprung up, **Asynchronous Module Definition (AMD)** and **CommonJS (CJS).**
- The former is much more popular with browser applications and the latter is much more popular with server applications written in node.js.
- With the evolution of ES6,now can create modules for the project requirements.

ES6 Modules(contd…)

Use.js

```
var localVariable = 123;
// not visible outside this file
export default function User(age)
 {
this.age = age;
 };
```

Use-details.js

```
import User from 'user';
var evilTrout = new User(35);
```

Babeljs

- Babeljs is a configurable transpiler, an ECMAScript 6 to ECMAScript 5 compiler.
- It allows you to use ES6 features in your projects and then compiles ES5 for you to use in production.
- Below is the command to install Babel using Node package manager.

  npm install babel

- You need Babel because browser vendors are slow to adopt new language features, thus browser support for ES6 is quite poor.

Babeljs(contd…)

Code Snippet:

- **let** myFunc = () => {
  console.log("ES6 is awesome!");
  };

The function above would get compiled to:

var myFunc = **function**() {
  console.log("ES6 is awesome!");
  };

SASS (Systematically Awesome Style Sheets)

- It is a CSS pre-processor.
- It is an extension of CSS that is used to add power and elegance to the basic language.
- It facilitates you to add variables, nested rules, mixins, inline imports, inheritance and more, all with fully CSS-compatible syntax.

SASS (Systematically Awesome Style Sheets)

Features of Sass:

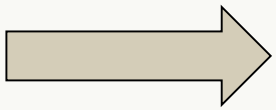
- Sass is fully CSS-compatible.
- It is more stable, powerful and elegant than CSS.
- It is based on JavaScript and is superset of CSS.
- It has its own syntax and compiles to readable CSS.
- It is an open-source pre processor that is interpreted into CSS.
- It supports language extensions such as variables, nesting, and mixins.
- It provides many useful functions for manipulating colors and other values.
- It provides many advanced features like control directives for libraries.
- It provides well-formatted, customizable output.

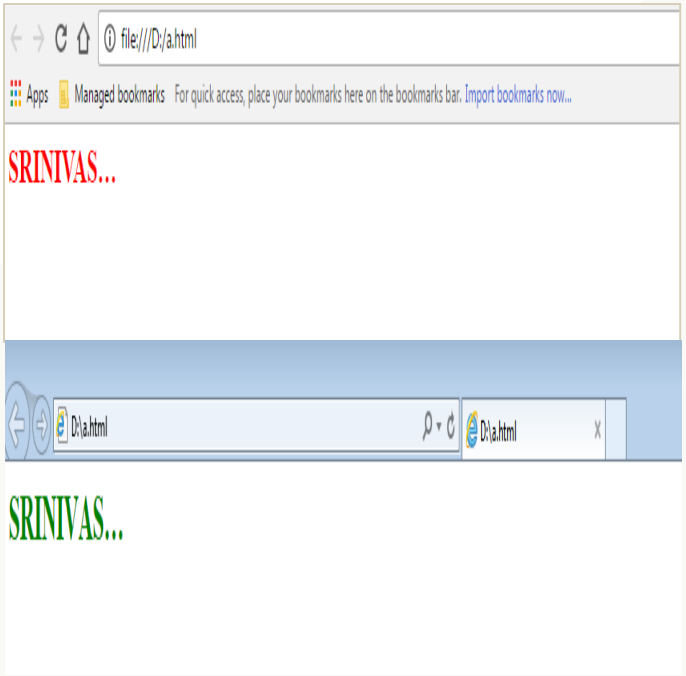
Disadvantages:

- Using Sass may cause of losing benefits of browser's built-in element inspector.

SASS (Systematically Awesome Style Sheets)

```
!DOCTYPE html>
<html>
<head>
  <title>Sass if() condition example
</title>
  <link rel="stylesheet" type="text/
css" href="simple.css"/>
</head>
<body>
  <h2>SRINIVAS…</h2>
</body>
</html>
```





Simple.css

```
h2{
  color: if( 1 + 1 == 3 , green , red);
}
```

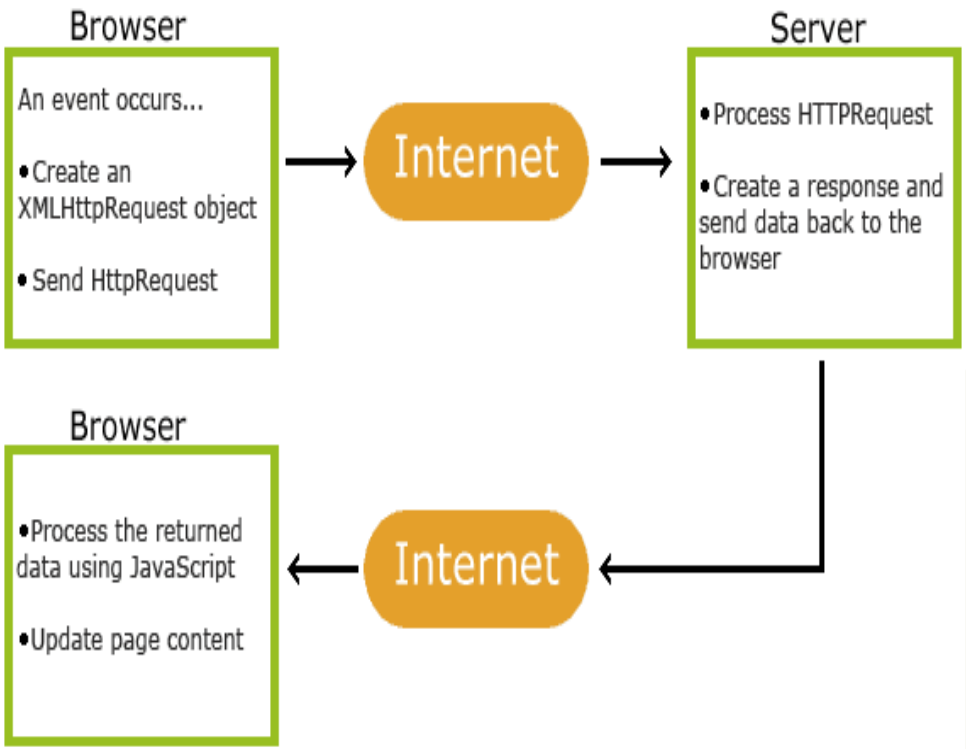
Polyfill

- A polyfill is a piece of code (usually JavaScript on the Web) used to provide modern functionality on older browsers that do not natively support it.
- Polyfills allow web developers to use an API regardless of whether it is supported by a browser or not, and usually with minimal overhead. Typically they first check if a browser supports an API, and use it if available, otherwise using their own implementation
- For example a polyfill could be used to mimic the functionality of an HTML Canvas element on Microsoft Internet Explorer 7 using a Silverlight plugin, or mimic support for CSS rem units, or text-shadow, or whatever you want.
- Some Popular Polyfills libraries:
  - HTML5 Shiv
  - CSS3PIE
  - Flexie
  - browserID

Ajax

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML
- In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-side scripts.
- It can send as well as receive information in a variety of formats, including JSON, XML, HTML, and even text files.
- AJAX's most appealing characteristic is its "asynchronous" nature which means it can communicate with the server, exchange data, and update the page all without having to refresh the browser.
- The two major features of AJAX allow you to do the following:
  - Make requests to the server without reloading the page
  - Receive and work with data from the server

Ajax(contd…)



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

Ajax(contd…)

- Code Snippet

```
function loadDoc() {
  var xhttp;
  if (window.XMLHttpRequest) {
    // code for modern browsers
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE6, IE5
    xhttp = new ctiveXObject("Microsoft.XMLHTTP");
          }
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

JQuery

- jQuery is a fast, small, and feature-rich JavaScript library.
- Current Version is 3.2.0.
- It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.
- With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Jquery(contd…)

jQuery includes the following features:

- DOM element selections using the multi-browser open source selector engine *Sizzle*, a spin-off of the jQuery project·
- DOM manipulation based on CSS selectors that uses elements' names and attributes, such as id and class, as criteria to select nodes in the DOM
- Events
- Effects and animations
- Ajax
- Deferred and Promise objects to control asynchronous processing
- JSON parsing
- Extensibility through plug-ins
- Utilities, such as feature detection
- Compatibility methods that are natively available in modern browsers, but need fall backs for older ones, such as inArray() and each()
- Multi-browser (not to be confused with cross-browser) support

Jquery(contd…)

Code Snippet

```
<script type = "text/javascript"
src = https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
 <script type = "text/javascript" language = "javascript">
   $(document).ready(function() {
  $("li").eq(2).addClass("selected");
  });
    <body>
    <div>
   <ul>
   <li>list item 1</li>        <li>list item 2</li>        <li>list item 3</li>
   <li>list item 4</li>        <li>list item 5</li>        <li>list item 6</li>
  </ul>
  </div>
  </body>
```

References

- https://babeljs.io/learn-es2015/
- http://sass-lang.com/
- https://www.npmjs.com/
- https://nodejs.org/en/docs/
- http://es6-features.org/#Constants
- https://jquery.com/
- https://www.tutorialspoint.com/jquery/jquery-traversing.htm
- http://www.javatpoint.com/ajax-tutorial

Recap

Es6

AMD

NPM

XMLHttpRequest

Jquery

Babel

Promises

AMD

Polyfill

callback

Thank You For Your Time

People matter, results count.