React.js is open source JS library and it is developed by facebook.

it can be used on client side and as well as on server

side with node.

Flexible,efficient and declarative JS library.

why React? -

MVC - React is used to develop view part from this

MVC. it is used develop UI part of application,

and specifically used for SPA(single page application)

it allows to create reusable set of components.

e.g Home

|

-- Header

-- Main

-- Footer

Header and Footer are the reusable set of components

1. react allows developer to create large web applications that can

change data without reloading of page. it is faster and scalable.

Features of React.js -

1. One way data flow - it is always from source to target. Whenever

changes in source are there it will reflect on target but directly from

target source can not be changed for this event handling or callbacks are

required.

2.Virtual DOM - In memory representation of actual DOM.

Whenever data changes due to some actions , changes will be done first

to virtual dom and it will be compared with actual dom and after that

only affected area will be updated on actual dom.

it is faster to load and better in performance.

3.JSX - instead of using regular JS, react uses JSX which allows developer

to write simple js with HTML quoting and render subcomponents using

HTML tags.

ECMA script - it is standard or specification used by javascript.

react uses simple JS - we should be aware of ECMA.

ES5 and ES6 differences

1. ES5 uses function keyword to implements class whereas ES6 uses class keyword

2. ES5 implements inheritance in 3 ways

i. prototype chaining

ii. classical inheritance

iii. psuedoclassical inheritance

ES6 implements inheritance using extends keyword.

Modern browsers support ES6 and also no browser is aware about JSX

but all browsers supports ES5, so we need transpiler for

ES6 to ES5 and JSX to ES5 i.e babel.

https://babeljs.io/

try it out

node.exe - runtime

node.js - sdk/collection of classes

npm - node package manager that is used to manage

dependencies

visual studio code.

CDN dependencies to generate react components

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.6.1/react.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/15.6.1/react-dom.min.js"></script>

Es6

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/16.10.2/umd/react.development.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react-dom/16.10.2/umd/react-dom.development.js"></script>

jsx

<script src="https://cdnjs.cloudflare.com/ajax/libs/react/16.10.2/umd/react.development.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/react-dom/16.10.2/umd/react-dom.development.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/babel-standalone/6.26.0/babel.min.js"></script>

all above scripts will be downloaded at client side whenever the page

is requested. This is going to affect performance of your application

so dependencies need to be managed on server side.

To manage babel on server side will need babel cli but this is

one case, we may have to manage other dependencies as well on

server side so instead of managing different dependencies

individually on server side we can make use of module bundler

that will help to manage all this.

module - reusable component, that encapsulates implementation and

exposes public interface

To deal with module we have different module formats/pattern.

Module Formats - it is basically a syntax which is used to export

and import modules.

AMD - Asynchronous module definition

it is used on client side, you can export/create module using

define() function. To import we need require.js

Common.js - it is used on server side using node - export is

taken care by module.export() and import is taken care using

require() function

UMD - universal module definition - which takes care of both i.e

AMD and commonJs

ES6 - export keyword to export the module and import keyword to

import the module

To use all these modules we have to make use of Module loader

or module bundler.

Module loader - inteprets module and return it in a certain

format. problem with this is it works at runtime.

Module Bundler - it works at build time.it loads the dependencies

and generate bundle file at build time.

2 module bundler - 1.browserify 2.webpack

browserify - deals with commonjs format only.

webpack - deals with commonjs,AMD,UMD,ES6