React

**1.GETTING STARTED**

Introduction

Components 🡪 Custom HTML elements 🡪 Can be reused

Card in html 🡪 reuse each time for each person.

Babel 🡪 Next Gen JS 🡪 Brower JS

Function Name Starts with Capital

Syntax JSX 🡪 JavaScript XML

Code Start

Function 🡪

function Person () {

return(

<h1>XYZ</h1>

);

}

ReactDOM.render(<Person/>, document.querySelector(‘#p1’));

Code End

HTML vs React

class ---- className

Props

Pass variables to components

JSX 🡪 Only 1 root element

Code Start

Var app = (

<div>

<person/>

<person />

</div>

);

Code End

Why React?

UI State 🡪 Difficult to manage using normal JS

Traversing easy but dynamically add/delete becomes tough

Allows us to focus on Buisness Logic

Highly Efficient and Fast

Huge EcoSystem and Community

Alternatives

Angular vs React vs Vue

Types of Pages

Single:

1 HTML Page, Content (Re)Rendered

Typically only 1 ReactDom.render()call

Multiple:

Multiple Pages, Content is rendered on Server

Can have widgets of React unaware of each other

One 1 ReactDom.render()call per widget

**2.NEXT GEN JS**

Keywords for creating variables –

Var , const , let

Let – new var –

Const – constant value

var is function scoped and let is block scoped.

Arrow Functions

Code Start

Function myFunc() {  
}

Const myFunc = () => {

}

Code End

Arrow – no problems with this keyword

Code Start

function MyName (name) {

console.log(name);

}

MyName("Kartik");

const MyName = (name) => {

console.log(name);

}

MyName("Kartik");

Code End

Only 1 argument then we can omit parenthesis

Code Start

const MyName = name => {

console.log(name);

}

MyName("Kartik");

Code End

IF you only return then we can write it in one line and omit return

Code Start

const doubleIt = number => number \* 2;

console.log(doubleIt(2));

Code End

EXPORTS AND IMPORTS

A screenshot of a cell phone

Description automatically generated

Spread and Rest Operators

**A screenshot of a cell phone

Description automatically generated**

Code Start

const CheckIt = (...number) => number.filter(el => el === 1);

console.log(CheckIt(1,2,3,4));

Code End

DESTRUCTURING

Extract array elements or object properties and store in variables

Code Start

[a,b] = [‘Hello’,’Kartik’]

{name,age} = {name: “Kartik, age: ‘23’}

DATA TYPES

Primitive

Number, String , Boolean

Copy by Value

Reference types

Arrays, Objects

Copy by reference

For true copy --- use spread operator

Code Start

Const person = {

Name : ‘Max’

}

Const two = {

…person

}

Code End

REFRESHING ARRAYS

Code Start

const numbers = [1,2,3]

const doubleNum = numbers.map( num => num\*2 );

console.log(doubleNum)

Code End

**3.BASICS**

Workflow 🡪 Optimize + Next Gen JS + Be more Productive

How 🡪 Dependency Mgmt tool (npm or yarn) + Bundler(Webpack) + Compiler (Babel + Presets) + Development Server

Create-react-app appname

Npm start

USE PARENTHESIS (); for multiline code

RETURN 1 Root Element

Use as many Stateless Components as possible ( changes using params) (Functional or Presentational Components or Dumb Components)

Stateful 🡪 Using state or useState ( Smart or Stateful Components)

**4.WORKING WITH LISTS AND CONDITIONALS**

**5. STYLING**

Npm install –save radium

Add pseudo css and media queries

**DEBUGGING  
  
STYLING  
  
COMPONENTS DEEP DIVE**

**HTTP REQUESTS**

**ROUTING  
  
FORMS AND VALIDATION**

**REDUX**

**AUTHENTICATION**

**TESTING INTRODUCTION  
  
DEPLOYMENT  
  
BONUS**