Data Binding in React

- Data binding is a technique used in web applications to access the data and bind to UI, identify the changes in UI and update into data.
- Web Applications support
    - One Way Binding
    - Two Way Binding
- React Supports only “One Way Binding”
- Two Way Binding requires “Event Handling”.
- Server Technologies support Two Way Binding by using frameworks like
    - MVC (Model View Controller)
    - MVP (Model View Presenter)
    - MVVM (Model View – View Model)
- You can store data in “Variables” if you are using “Function” component.
- You can store data in “Properties” if you are using “Class” component.
- Variables and Properties are “Immutable”, they can’t change according to state and situation.
- Hence you have to use “State” for component.

Ex: Function Component without state

DataBindingComponent.js

```
export default function DataBindingComponent()
{
   var product = {
      Name: "JBL Speaker",
      Price: 4500.55,
      Stock: true,
      Photo: "images/speaker.jpg"
   };
   return(
      <div className="container-fluid">
         <h3>Product Details</h3>
         <dl>
            <dt>Edit Name</dt>
            <dd><input type="text" value={product.Name} /></dd>
```

```
            <dt>Name</dt>
            <dd>{product.Name}</dd>
            <dt>Price</dt>
            <dd>{product.Price}</dd>
            <dt>Stock</dt>
            <dd>{(product.Stock==true)?"Available":"Out of Stock"}</dd>
            <dt>Preview</dt>
            <dd>
              <img src={product.Photo} alt="speaker" width="100" height="100"/>
            </dd>
          </dl>
        </div>
    )
}
```

Ex: Properties in Class Component

DataBindingComponent.js

```
import React from "react";
export default class DataBindingComponent extends React.Component
{
    product = {
        Name: "Nike Casuals",
        Price: 6500.55,
        Stock: true,
        Photo: "images/shoe.jpg"
    };
    render(){
        return(
            <div className="container-fluid">
                <h3>Product Details</h3>
```

```
            <dl>
              <dt>Edit Name</dt>
              <dd><input type="text" value={this.product.Name} /></dd>
              <dt>Name</dt>
              <dd>{this.product.Name}</dd>
              <dt>Price</dt>
              <dd>{this.product.Price}</dd>
              <dt>Stock</dt>
              <dd>{(this.product.Stock==true)?"Available":"Out of Stock"}</dd>
              <dt>Preview</dt>
              <dd>
                <img src={this.product.Photo} alt="speaker" width="100" height="100"/>
              </dd>
            </dl>
          </div>
      )
    }
}
```

State in Function Component

- State allows to store data and handle between requests in a component.
- React 16.8 and higher versions are introduced with “useState()” hook in function component.
- It can maintain state for component, where you can store value and use across requests.
- “useState()” configure a state object.

Syntax:

```
import  {useState} from ‘react’;

const [getterRef, setter] = useState();

setter(value);              - set value into state
```

{ getterRef } - get value from state

FAQ: Why state is defined as “const”?

A: State requires initialization of memory. “const” defines initialization.

Ex: **State in Function Component**

DataBindingComponent.js

```
import {useState} from "react";

export default function DataBindingComponent()
{
   const [product, setProduct] = useState({Name:'', Price: 0, Stock: false});
   const [title, setTitle] = useState('State in Function Components');
   function handleNameChange(e){
       setProduct({
          Name: e.target.value,
          Price: product.Price,
          Stock: product.Stock
       })
   }
   function handlePriceChange(e){
      setProduct({
         Name: product.Name,
         Price: e.target.value,
         Stock: product.Stock
      })
  }
  function handleStockChange(e){
   setProduct({
```

```
    Name: product.Name,
    Price: product.Price,
    Stock: e.target.checked
  })
 }
  return(
    <div className="container-fluid">
      <h1 className="text-center">{title}</h1>
      <div className="row">
          <div className="col-3">
           <h2>Register Product</h2>
           <dl>
              <dt>Name</dt>
              <dd>
                 <input type="text" value={product.Name} onChange={handleNameChange} />
              </dd>
              <dt>Price</dt>
              <dd>
                 <input type="text" value={product.Price} onChange={handlePriceChange} />
              </dd>
              <dt>Stock</dt>
              <dd className="form-check form-switch">
                <input className="form-check-input" type="checkbox"
checked={product.Stock} onChange={handleStockChange} />
              </dd>
           </dl>
          </div>
          <div className="col-9">
          <h2>Product Details</h2>
          <dl>
```

```
            <dt>Name</dt>
            <dd>{product.Name}</dd>
            <dt>Price</dt>
            <dd>{product.Price}</dd>
            <dt>Stock</dt>
            <dd>{(product.Stock==true)?"Available":"Out of Stock"}</dd>
          </dl>
          </div>
        </div>
      </div>
   )
}
```

State with React Class Component

- React class component is a state full component.
- It implicitly provides state for component.
- "React.Component" base class provides state, which you can use in your component.
- State is defined for component at the time of allocating memory for component and creating an object for component.
- It is handled by the constructor of class.

Syntax:

```
class Demo extends React.Component
{
  constructor(props){
        super(props);
        this.state = {  property: value }
  }
}
<div> {this.state.property} </div>
```

- To store a new value or to assign value into state property you have to use the method “setState()”.

```
setState({
    property: newValue
})
```

- Class component will not allow the events to handle state directly.
- Class component can handle events directly without any state function.
  [Events can’t handle state of class component directly, they need a binding technique that will bind the events to class components]
- Event requires a “binding” technique that bind the event with class
    - **You can bind in constructor**
      constructor(props)
      {
        super(props);
        this.handleClick = this.handleClick.bind(this);
      }
    - **You can bind in event handler**
      <button onClick={this.handleClick.bind(this)}>
    - **You can configure a call back in event handler.**
      <button onClick={()=>this.handleClick()}>

Ex:

DataBindingComponent.js

```
import React from 'react';


export default class DataBindingComponent extends React.Component
{
   constructor(props) {
      super(props);
      this.state = {
         Name: '',
         Price: 0,
         Stock: false
      }
```

```
    this.handleNameChange = this.handleNameChange.bind(this);
    this.handlePriceChange = this.handlePriceChange.bind(this);
    this.handleStockChange = this.handleStockChange.bind(this);
}

handleNameChange(e){
    this.setState({
        Name:e.target.value,
        Price: this.state.Price,
        Stock: this.state.Stock
    })
}

handlePriceChange(e){
    this.setState({
        Name: this.state.Name,
        Price: e.target.value,
        Stock: this.state.Stock
    })
}

handleStockChange(e){
    this.setState({
        Name: this.state.Name,
        Price: this.state.Price,
        Stock: e.target.checked
    })
}
```

```
  render(){
    return(
      <div className="container-fluid">
      <h1 className="text-center">State in Class Component</h1>
      <div className="row">
        <div className="col-3">
         <h2>Register Product</h2>
         <dl>
           <dt>Name</dt>
           <dd>
             <input type="text" onChange={this.handleNameChange}
value={this.state.Name} className="form-control" />
           </dd>
           <dt>Price</dt>
           <dd>
             <input type="text" onChange={this.handlePriceChange}
value={this.state.Price} className="form-control" />
           </dd>
           <dt>Stock</dt>
           <dd className="form-check form-switch">
             <input onChange={this.handleStockChange} checked={this.state.Stock}
className="form-check-input" type="checkbox" />
           </dd>
         </dl>
        </div>
        <div className="col-9">
        <h2>Product Details</h2>
        <dl>
          <dt>Name</dt>
          <dd>{this.state.Name}</dd>
```

```
                <dt>Price</dt>
                <dd>{this.state.Price}</dd>
                <dt>Stock</dt>
                <dd>{(this.state.Stock==true?"Available":"Out of Stock")}</dd>
              </dl>

              </div>
            </div>
          </div>
          )
        }
    }
```

Presenting Complex Data

Array and Array of Objects

- React uses “map()” to perform iteration over collection and present the items in collection.

  Syntax:
  collection.map(item => { item })

- Every item in iterator requires a “Key”, in React it is mandatory to define a unique key for every iterating element.

Ex:

ProductsComponent.js

```
import { useState } from "react";

export default function ProductsComponent()
{
   const [categories, setCategories] = useState(['All','Electronics','Footwear','Fashion']);
   return(
      <div className="container-fluid">
```

```
<h2>Categories List</h2>
<ol>
  {
    categories.map(category =>
      <li key={category}>{category}</li>
    )
  }
</ol>
<h2>Select a Category</h2>
<select className="form-select w-25">
  {
    categories.map(category =>
      <option key={category} value={category}>{category}</option>
    )
  }
</select>
<h2>Categories List</h2>
<table className="table table-hover w-25">
  <thead>
    <tr>
      <th>Choose Category</th>
    </tr>
  </thead>
  <tbody>
    {
      categories.map(category =>
        <tr key={category}>
          <td>{category}</td>
        </tr>
```

```
                    )
                }
            </tbody>
        </table>
    </div>
  )
}
```

Ex:

ProductsComponent.js

```
import { useState } from "react";


export default function ProductsComponent()
{
  const [products, setProducts] = useState([{Id:1, Name:'JBL Speaker', Price:4500.55, Photo:'images/speaker.jpg'},{Id:2, Name:'Nike Casuals', Price:7000.55, Photo:'images/shoe.jpg'}]);
  return(
    <div className="container-fluid">
      <h2>Products Table</h2>
      <table className="table table-hover w-50">
        <thead>
          <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Preview</th>
          </tr>
        </thead>
        <tbody>
          {
```

```
                products.map(product =>
                  <tr key={product.Id}>
                     <td>{product.Name}</td>
                     <td>{product.Price}</td>
                     <td><img alt={product.Name} src={product.Photo} height="50" width="50"
/></td>
                  </tr>
                 )
              }
            </tbody>
        </table>
        <h2>Products Catalog</h2>
        <div className="d-flex flex-wrap flex-row">
          {
             products.map(product =>
               <div className="card m-3 p-2" key={product.Id}>
                 <img alt={product.Name} src={product.Photo} height="200" className="card-
img-top" />
                 <div className="card-header">
                  <h3>{product.Name}</h3>
                  <p> &#8377; {product.Price}</p>
                 </div>
               </div>
               )
          }
        </div>
      </div>
   )
}
```

Ex:

ProductsComponent.js

```
import { useState } from "react";


export default function ProductsComponent()
{
   const [data, setData] = useState([{Category:'Electronics',
Products:['TV','Mobile']},{Category:'Footwear', Products:['Nike Casuals', 'Lee Boot']}]);
   const [product, setProduct] = useState('');


   function handleProductChange(e){
      setProduct(e.target.value);
   }


   return(
      <div className="container-fluid">
         <h2>Select a Product</h2>
         <ol>
            {
               data.map(item =>
                  <li key={item.Category}>
                     {item.Category}
                     <ul>
                        {
                           item.Products.map(product=>
                            <li key={product}>
                               <input type="checkbox" /> {product}
                           </li>
                           )
                        }
```

```
                    </ul>
                  </li>
                )
              }
            </ol>
            <h2>Select Product</h2>
            <select className="form-select w-25" onChange={handleProductChange}>
              {
                data.map(item =>
                  <optgroup key={item} label={item.Category}>
                    {
                      item.Products.map(product=>
                        <option key={product}>
                          {product}
                        </option>
                      )
                    }
                  </optgroup>
                )
              }
            </select>
            <div className="mt-3">
              Selected Product : {product}
            </div>
          </div>
        )
}
```

Fetch Data from API and Present

- React can use various techniques for interacting with API
  - JavaScript “fetch()”

    - jQuery “$.ajax()”
    - 3rd Party Library
        - Axios
        - WhatwgFetch
- Fetch
    - It is JavaScript method that uses “XmlHttpRequest” object
    - It returns data in binary format.
    - You have to convert into JSON explicitly.
    - It requires configuration for CORS explicitly
- Axios
    - It uses “XmlHttpRequest” object
    - It returns data in JSON
    - Handles CORS
    - Handles XSS [Cross Site Scripting Attacks]
    - More secured

Ex: Fetch API

FetchDemoComponent.js

```
import { useState, useEffect } from "react";


export default function FetchDemoComponent()
{
   const [categories, setCategories] = useState([]);
   const [products, setProducts] = useState([]);


   useEffect(()=>{
      fetch('http://fakestoreapi.com/products/categories')
      .then(response => response.json())
      .then(data => {
         let allcategories = data;
         allcategories.unshift('All');
         setCategories(data)
      });
      fetch('http://fakestoreapi.com/products')
      .then(response=> response.json())
```

```
    .then(data => {
        setProducts(data)
    })
},[])

function handleCategoryChange(e){
   if(e.target.value=='All'){
    fetch(`http://fakestoreapi.com/products`)
    .then(response=> response.json())
    .then(data=>{
        setProducts(data);
    })
   } else {
    fetch(`http://fakestoreapi.com/products/category/${e.target.value}`)
    .then(response=> response.json())
    .then(data=>{
        setProducts(data);
    })
   }
}

return(
    <div className="container-fluid">
        <header className="bg-danger text-white text-center p-2">
            <h2> <span className="bi bi-cart2"></span> Shopping Online</h2>
        </header>
        <div className="row">
          <div className="col-3">
            <h3>Select a Category</h3>
```

```
            <select className="form-select" onChange={handleCategoryChange}>
              {
                 categories.map(category =>
                    <option value={category}
key={category}>{category.toUpperCase()}</option>
                  )
              }
            </select>
          </div>
          <div className="col-9">
            <h2>Products List</h2>
            <div className="d-flex flex-wrap flex-row" style={{height:'500px',
overflow:'auto'}} >
              {
                 products.map(product=>
                   <div style={{width:'200px'}} className="card m-2" key={product.id}>
                     <img className="card-img-top"  src={product.image} alt={product.title}
height="200" />
                     <div className="card-header" style={{height:'200px'}}>
                       <p>{product.title}</p>
                     </div>
                     <div className="card-footer">
                       <p>$ {product.price}</p>
                       <button className="btn btn-danger w-100">
                          <span className="bi bi-cart2"></span>
                          Add to Cart
                       </button>
                     </div>
                   </div>
```

```
                )
            }
          </div>
        </div>
      </div>
    </div>
  )
}
```

Connecting with API using "Axios"

- Axios can handle data by default in JSON format.
- It is suitable for legacy and modern browsers.
- It uses Async requests.
- It uses unblocking technique.
- It prevents XSS [Cross Site Scripting Attacks]
- It prevents Request Forgery
- It provides better error handling objects, that can track errors in communication with API.
- It supports CORS. [Cross Origin Resource Sharing]
- Axios can handle multiple requests simultaneously at the same time.

Syntax:

```
axios({
    method: 'GET|POST',
    url: 'API_URL',
    data: 'Data to POST'
})
```

Shorthand Technique

```
axios.get(url).then(function(response){});
axios.post(url, data);
```

Multiple Requests

```
axios.all({
   [
```

```
    axios.get(url),          0
    axios.get(url)           1
    ]
})
```

Response Object

- Axios response object comprises of all response details like
  - statusCode
  - statusText
  - header
  - data

Ex: Axios API Request

Install Axios in your project

> npm install axios --save

AxiosComponent.js

```
import { useState, useEffect } from "react";
import axios from 'axios';

export default function AxiosComponent()
{
    const [products, setProducts] = useState([]);
    const [categories, setCategories] = useState([]);
    const [filteredProducts, setFilteredProducts] = useState([]);

    useEffect(()=> {
        axios.get('http://fakestoreapi.com/products')
        .then(response=> {
            setProducts(response.data);
            setFilteredProducts(response.data);
        });

        axios.get('http://fakestoreapi.com/products/categories')
```

```
      .then(response=>{
         let data = response.data;
         data.unshift('All');
         setCategories(data);
      });

},[]);
function handleCategoryChange(e){
   if(e.target.value==='All')
   {
      setFilteredProducts(products);
   } else {
      setFilteredProducts(products.filter(product=> product.category===e.target.value));
   }
}
return(
   <div className="container-fluid">
      <div className="row">
         <div className="col-3">
            <h3>Select a Category</h3>
            <select className="form-select" onChange={handleCategoryChange} >
               {
                  categories.map(category=>
                     <option key={category} value={category}>
                        {category.toUpperCase()}
                     </option>
                     )
               }
            </select>
```

```
            </div>
            <div className="col-9">
              <h3>Products List</h3>
              <div className="d-flex flex-row flex-wrap">
                {
                  filteredProducts.map(product=>
                    <div style={{width:'200px'}} className="card m-2" key={product.id}>
                    <img className="card-img-top"  src={product.image} alt={product.title}
height="200" />
                    <div className="card-header" style={{height:'200px'}}>
                      <p>{product.title}</p>
                    </div>
                    <div className="card-footer">
                      <p>$ {product.price}</p>
                      <button className="btn btn-danger w-100">
                        <span className="bi bi-cart2"></span>
                        Add to Cart
                      </button>
                    </div>
                    </div>
                    )
                }
              </div>
            </div>
          </div>
        </div>
    )
}
```

Events in React

- React uses all JavaScript Browser events

- React uses “camel case” for events.
    - onClick
    - onChange
    - onBlur
    - onFocus
    - onMouseover
    - onMouseout
    - onKeyup etc..
- Event handler can’t use “this” keyword as event argument.
- Only event object is allowed.
    - JavaScript
      <button onclick=”InsertClick(this, event)”>
      InsertClick(obj, e)
      {
      }
    - React
      <button onClick={InsertClick}>
      function InsertClick(e)
      {
      e.eventProperties;
      e.target.objectProperties;
      }

Note: React is not using Browser Events, these are virtual events known as “Synthetic Events”

“SyntheticEvent”

Ex:

```
import { useState } from "react";


export default function EventComponent()
{
   const[msg, setMsg] = useState('');
   function DatabaseClick(e){
      switch(e.target.value)
      {
         case 'Insert':
```

```
            setMsg('Record Inserted..');
            break;
            case 'Update':
            setMsg('Record Updated..');
            break;
            case 'Delete':
            setMsg('Record Deleted..');
            break;
        }
    }
    return(
        <div className="container-fluid">
            <div className="mt-3">
                <button onClick={DatabaseClick} value="Insert">Insert</button>
                <button onClick={DatabaseClick}  value="Update">Update</button>
                <button onClick={DatabaseClick} value="Delete">Delete</button>
                <h2>{msg}</h2>
            </div>
        </div>
    )
}
```

Prevent Default

```
function InsertClick(e)
{
e.preventDefault();
}
```

Forms and Validations