03. D3JS Data Join.

Data join is another important concept in D3.js. It works along with selections and enables us to manipulate the HTML document with respect to our data set (*a series of numerical values*). By default, *D3.js* gives data set the highest priority in its methods and each item in the data set corresponds to a HTML element. This chapter explains data joins in detail.

3.1. What is Data Join?

Data join enables us to inject, modify and remove elements (*HTML element as well as embedded SVG elements*) based on the data set in the existing HTML document. By default, each data item in the data set corresponds to an element (*graphical*) in the document.

As the **data set** changes, the corresponding element can also be manipulated easily. Data join creates a close relationship between our data and graphical elements of the document. Data join makes manipulation of the elements based on the data set a very simple and easy process.

3.2. How Data Join Works?

The primary purpose of the Data join is to map the elements of the existing document with the given data set. It creates a virtual representation of the document with respect to the given data set and provides methods to work with the virtual representation. Let us consider a simple data set as shown below.

Example:

```
[10, 20, 30, 25, 15]
```

The data set has five items and so, it can be mapped to five elements of the document. Let us map it to the li element of the following document using the selector's *selectAll*() method and data join's data() method.

Example: HTML

```
<ul id = "list">
   <li><li>
   <li></li>
</ul>
```

Example: D3.js code

```
d3.select("#list").selectAll("li").data([10, 20, 30, 25, 15]);
```

Now, there are five virtual elements in the document. The first two virtual elements are the two li element defined in the document as shown below.

Output:

```
1. li - 10
2. li - 20
```

We can use all the selector's element modifying methods like *attr(), style(), text()*, etc., for the first two **li** as shown below.

Example:

```
d3.select("#list").selectAll("li")
   .data([10, 20, 30, 25, 15])
   .text(function(d) { return d; });
```

The function in the text() method is used to get the li elements mapped data. Here, d represent 10 for first li element and 20 for second li element.

The next three elements can be mapped to any elements and it can be done using the data join's enter() and selector's append() method. The enter() method gives access to the remaining data (which is not mapped to the existing elements) and the append() method is used to create a new element from the corresponding data. Let us create li for the remaining data items as well. The data map is as follows −

Output:

```
3. li - 30
4. li - 25
5. li - 15
```

The code to create new a li element is as follows −

Example:

```
d3.select("#list").selectAll("li")
   .data([10, 20, 30, 25, 15])
   .text(function(d) { return "Pre-existing element with value: "+d;}}
   .enter()
   .append("li")
   .text(function(d)
      { return "New element with value:  " + d; });
```

Data join provides another method called as the exit() method to process the data items removed dynamically from the data set as shown below.

Example:

```
d3.selectAll("li")
   .data([10, 20, 30, 15])
   .exit()
   .remove()
```

Here, we have removed the fourth item from the data set and its corresponding li using the exit() and the remove() methods.

Example: The complete code is as follows.

```
<!DOCTYPE html>
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
      </style>
   </head>

   <body>
      <ul id = "list">
         <li></li>
         <li></li>
      </ul>

      <input type = "button" name = "remove" value = "Remove 4th value"
         onclick = "javascript:remove()" />

      <script>
         d3.select("#list").selectAll("li")
            .data([10, 20, 30, 25, 15])
            .text(function(d)
               { return "Pre-existing element with value " + d; })
```

```
                .enter()
                .append("li")
                .text(function(d)
                   { return "New element with value " + d; });

            function remove() {
                d3.selectAll("li")
                .data([10, 20, 30, 15])
                .exit()
                .remove()
            }
        </script>
    </body>
</html>
```

The result of the above code will be as follow:

Output:

- Pre-existing element with value is 10
- Pre-existing element with value is 20
- New element with value is 30
- New element with value is 25
- New element with value is 15

3.3. Data Join Methods

Data join provides the following four methods to work with data set :

- **datum()**
- **data()**
- **enter()**
- **exit()**

The datum() Method

The datum() method is used to set value for a single element in the HTML document. It is used once the element is selected using selectors. For example, we can select an existing element (p tag) using the select() method and then, set data using the datum() method. Once data is set, we can either change the text of the selected element or add new element and assign the text using the data set by datum() method.

Example: Create a page "*datajoin_datum.html*" and add the following code

```
<!DOCTYPE html>
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>
   <body>
      <p></p>
      <div></div>
      <script>
         d3.select("p")
         .datum(50)
         .text(function(d) {
            return "Used existing paragraph with data " + d ;
         });

         d3.select("div")
         .datum(100)
         .append("p")
         .text(function(d) {
            return "Created new paragraph with data " + d ;
         });
      </script>
   </body>
</html>
```

The output of the above code will be as follows.

```
Used existing paragraph with data 50.
Created new paragraph with data 100.
```

The data() method

The data() method is used to assign a data set to a collection of elements in a HTML document. It is used once the HTML elements are selected using selectors. In our list example, we have used it to set the data set for the li selector.

Example:

```
d3.select("#list").selectAll("li")
    .data([10, 20, 30, 25, 15]);
```

The enter() method

The enter() method outputs the set of data item for which no graphic element existed before. In our list example, we have used it to create new li elements.

Example:

```
d3.select("#list").selectAll("li")
    .data([10, 20, 30, 25, 15])
    .text(function(d) { return "Existing element with value " + d; })
    .enter()
    .append("li")
    .text(function(d) { return "New element with value " + d; });
```

The exit() method

The exit() method outputs the set of graphic elements for which no data exists any longer. In our list example, we have used it to remove one of the li element dynamically by removing the data item in the data set.

Example:

```
function remove() {
   d3.selectAll("li")
      .data([10, 20, 30, 15])
      .exit()
      .remove()
}
```

3.4. Data Function

In the DOM manipulation chapter, we learned about different DOM manipulation methods in D3.js such as style(), text(), etc. Each of these functions normally takes a constant value as its parameter. Nevertheless, in the context of Data join, it takes an anonymous function as a parameter. This anonymous function takes the corresponding data and the index of the data set assigned using the data() method. So, this anonymous function will be called for each of our data values bound to the DOM. Consider the following **text**() function.

Example:

```
.text(function(d, i) {
   return d;
});
```

Within this function, we can apply any logic to manipulate the data. These are anonymous functions, meaning that there is no name associated with the function. Other than the data (d) and index (i) parameter, we can access the current object using this keyword as shown below:

Example:

```
.text(function (d, i) {
   console.log(d); // the data element
   console.log(i); // the index element
   console.log(this); // the current DOM object
   return d;
});
```

Consider the following example.

Example:

```
<!DOCTYPE html>
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
      <style>
         body { font-family: Arial; }
      </style>
   </head>
   <body>
      <p></p>
      <p></p>
      <p></p>
      <script>
         var data = [1, 2, 3];
         var paragraph = d3.select("body")
         .selectAll("p")
         .data(data)
         .text(function (d, i) {
            console.log("d: " + d);
            console.log("i: " + i);
```

```
                console.log("this: " + this);
                return "The index is " + i + " and the data is " + d;
            });
        </script>
    </body>
</html>
```

The above script will generate the following result:

Output:

```
The index is 0 and the data is 1

The index is 1 and the data is 2

The index is 2 and the data is 3
```

In the above example, the parameter "d" gives you your data element, "i" gives you the index of data in the array and "this" is a reference of the current DOM element. In this case, it is the paragraph element. Notice that we have called .data(data) function above. The data() function provides data to the selected elements, in our case it is data array.