Using Data Bound to DOM Elements

Basic Example

In this example, you will use D3.js to bind data to DOM elements of a basic webpage. Then you will use D3.js to use the data bound to DOM Elements to update the basic webpage.

In the last section, we started with a basic webpage:

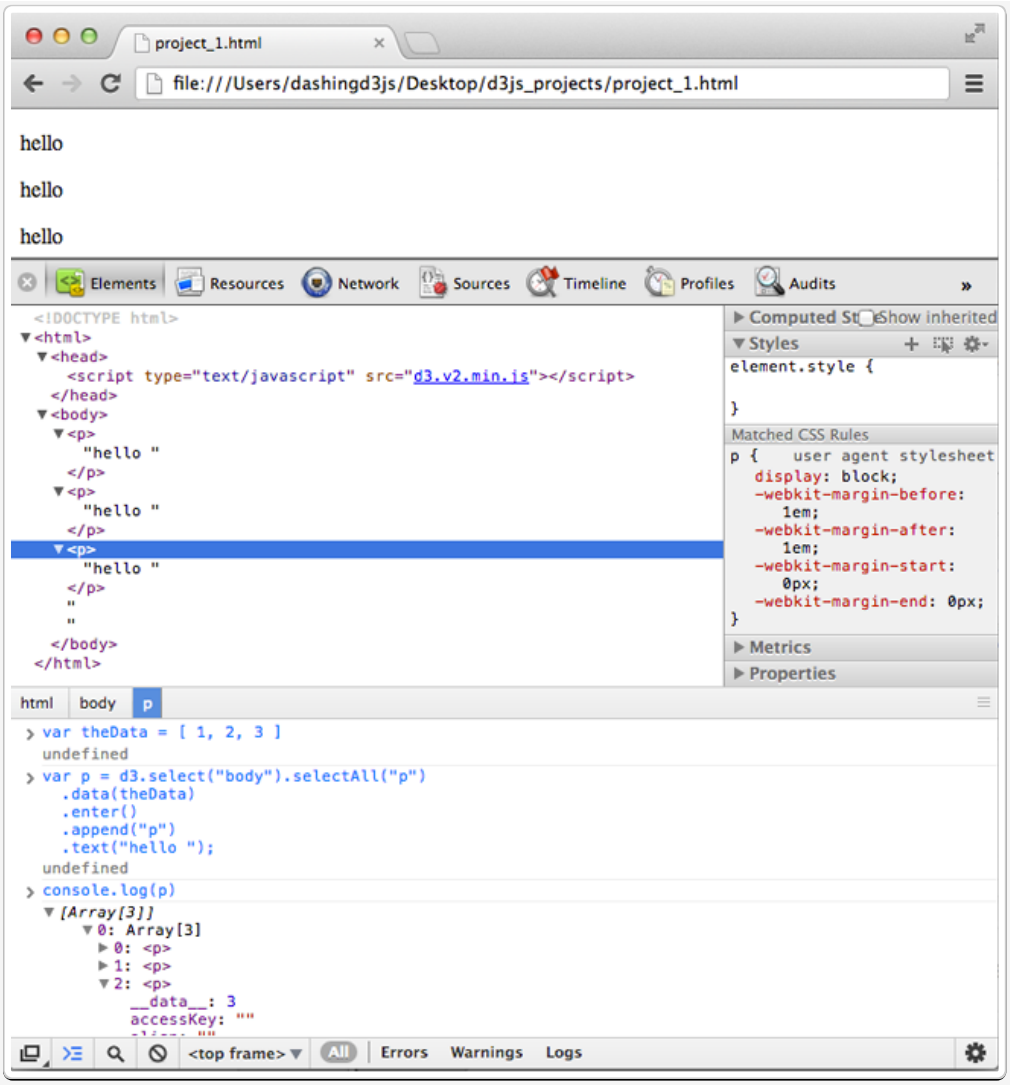
```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="d3.v2.min.js"></script>
  </head>
  <body>
  </body>
</html>
```

And then bound data to paragraph HTML elements through the JavaScript Console:

```
var theData = [ 1, 2, 3 ]

var p = d3.select("body").selectAll("p")
  .data(theData)
  .enter()
  .append("p")
  .text("hello ");
```

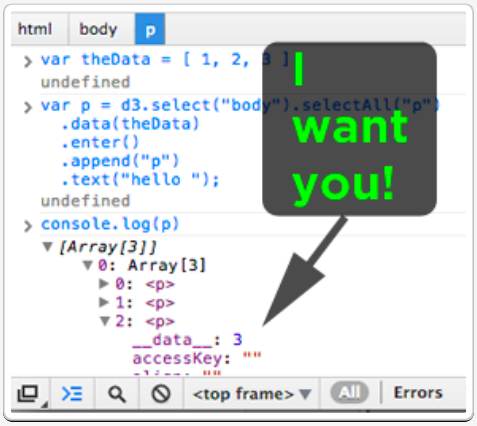
Which gave us this:



Which is amazing - we have bound data to the DOM elements using D3.js.

However, what we really want is to get our data back on to the basic webpage.

Put another way, how do we get this value out of the DOM element?



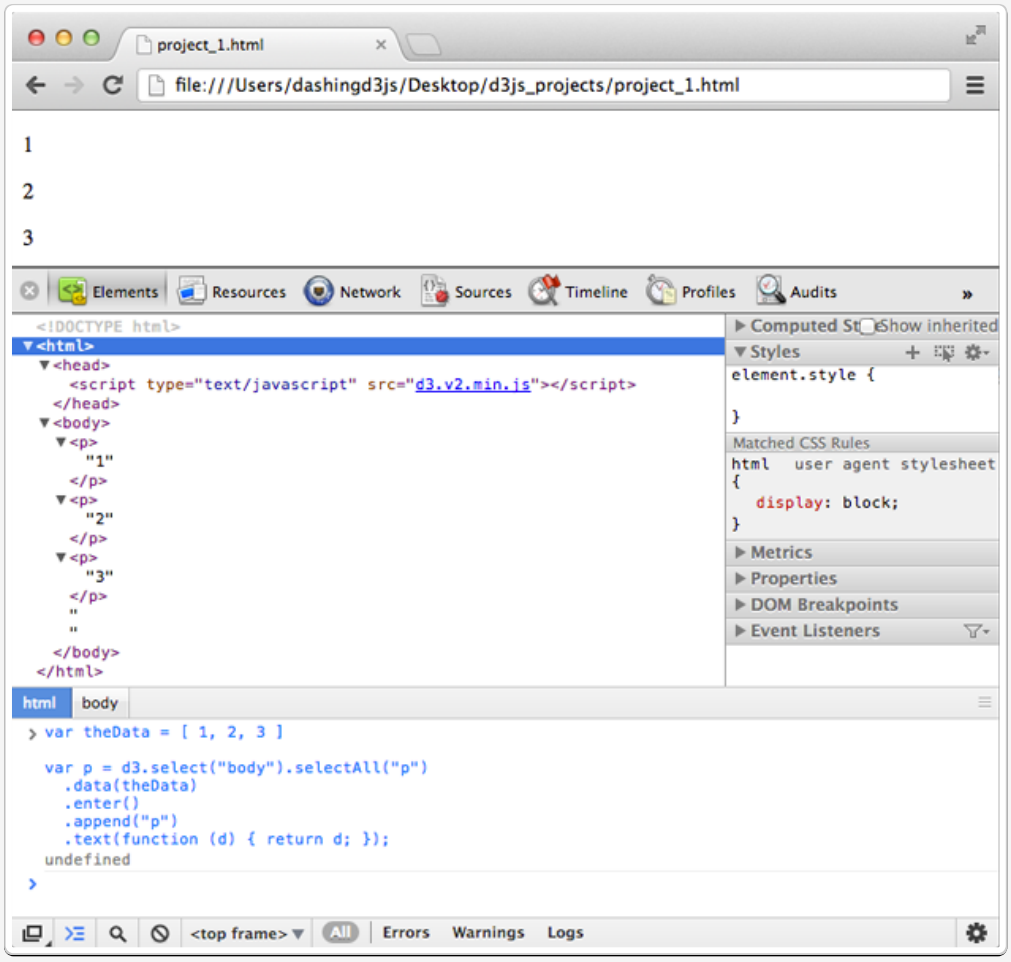
Open the Webkit Inspector to get the JavaScript Console going along side the Elements Inspector.

Type the following into the JavaScript Console:

```
1  var theData = [ 1, 2, 3 ]
2
```

```
var p = d3.select("body").selectAll("p")
  .data(theData)
  .enter()
  .append("p")
  .text( function (d) { return d; } );
```

Which gives us the following:



As you can see - instead of three paragraphs that say "Hello", we now have three paragraphs that contain our data! Congratulations - you have used data bound to DOM elements to update the webpage using D3.js!

D3.js Text Operator Revisited

The only part of the JavaScript code that changed from our previous example to this example is that

```
.text( "hello ");
```

became:

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

The selection that the **.text()** operator is working on in both cases is the following:

```
d3.select("body").selectAll("p")
  .data(theData)
```

```
    .enter()
    .append("p")
```

This selection represents three HTML **<p>** elements.

As we covered in the previous section, the Text Operator sets the textContent of the node to the specified value on all selected elements.

What is also true of the Text Operator is that if the value passed to it (the Text Operator) is a function, then the function is evaluated for each element in order. **And** the functions result is used to set each element's text context.

Which in our case means, that instead of setting "Hello" as the text value, the function goes through each element, gets the __data__ property and returns it to the Text Operator, so that it can set that result as the text content for the element.

JavaScript Functions in D3.js Operators

We are passing a JavaScript function to the Text operator:

```
function (d) { return d; }
```

If you are new to JavaScript functions, here is a brief synopsis:

```
function functionName (variableName) {
  return variableName;
}
```

The "function" operator defines a function named "functionName" which takes a variable "variableName" and then returns the same variable.

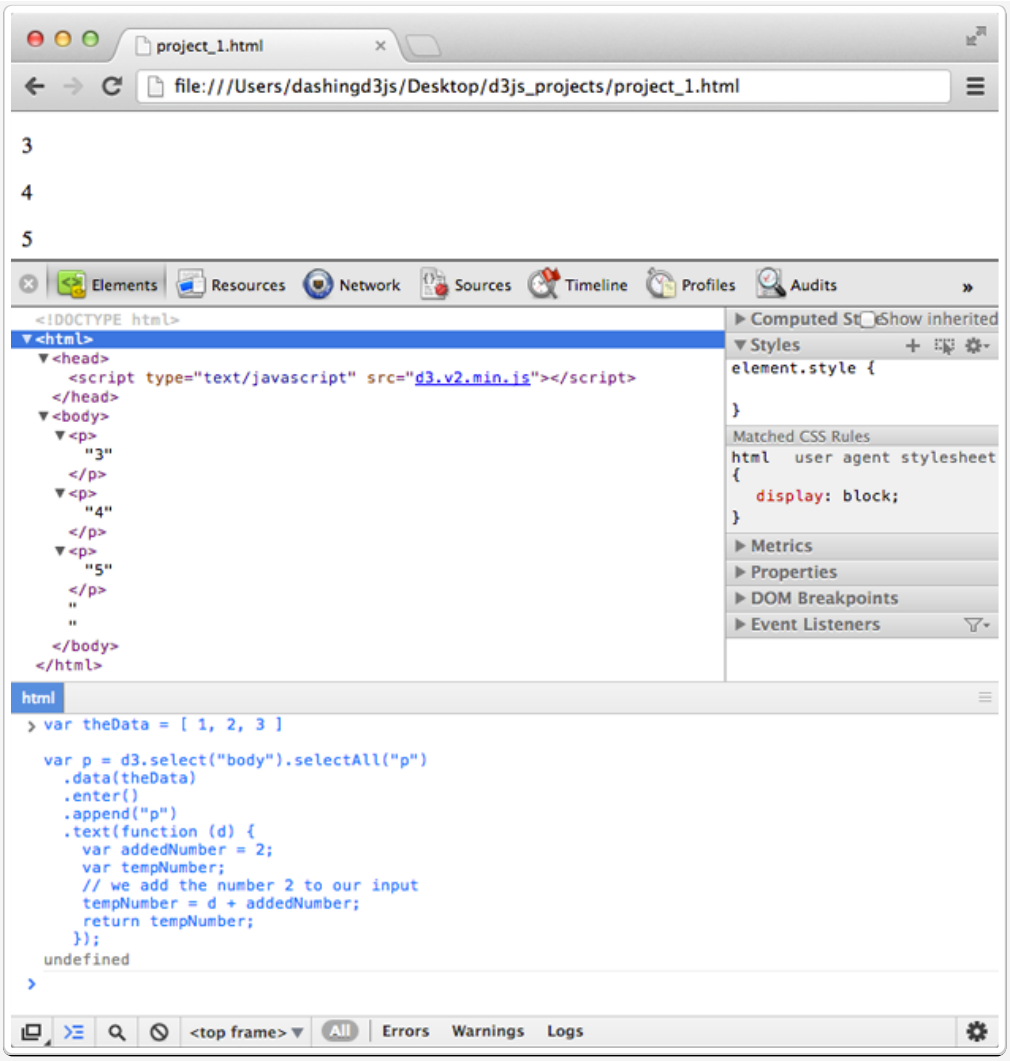
If the functionName is missing, then it is called an *anonymous function*. Anonymous functions are used to make code more concise when declaring a function that will only be used in one place. In our case, the function is used only inside of the D3.js Text Operator:

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

Note - our function doesn't have to be one line or just a simple return, we could so something like this:

```
function (d) {
  var addedNumber = 2;
  var tempNumber;
  // we add the number 2 to our input
  tempNumber = d + addedNumber;
  return tempNumber;
}
```

Which when put into the JavaScript Console as part of our overall D3.js functions, gives us the following:



As you can see, our function has added the number "2" to our data set and updated the text of the DOM elements.

Using these functions, you can write anything you want using JavaScript to manipulate the data. This is powerful in data visualizations when you will want differentiate style and format of SVG objects based on what values the data contains.

Variables Available inside D3.js Operators

In our example:

```
1  .text( function (d) { return d; } );
```

You can see that the variable *d* is available for use in the anonymous function. This variable is provided to us by D3.js and it refers to the current *__data__* attribute for the specific element being processed.

The are two other variables provided to us by D3.js - *this* and *i*.

this refers to the current DOM element being evaluated.

i refers to the index of the current HTML element being evaluated in the selected elements selection. Remember that all the data is processed at once and in sequential order. **Note** - *i* starts at 0. This means in our data set

```
var theData = [ 1, 2, 3 ]
```

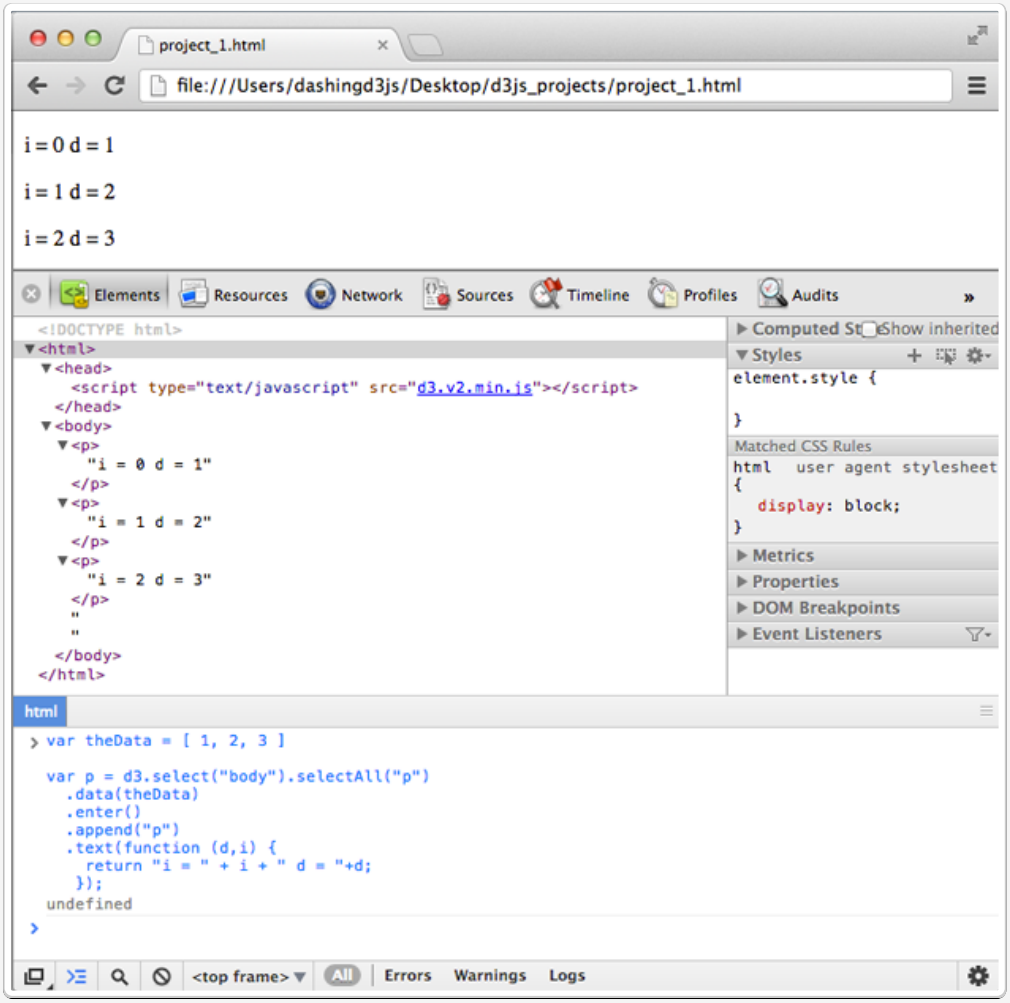
The data is processed in order of "1", "2" and then "3". With "1" having an *i* of 0, "2" having an *i* of 1 and "3" having an *i* of 2.

You can run the following in the JavaScript Console to see this demonstrated:

```
var theData = [ 1, 2, 3 ]

var p = d3.select("body").selectAll("p")
  .data(theData)
  .enter()
  .append("p")
  .text(function (d,i) {
    return "i = " + i + " d = "+d;
   });
```

Which gives us the following:



Having the index available allows for things like alternating style and formatting.

← Binding Data to DOM Elements

Creating SVG Elements Based on Data →

Learn D3.js

- D3 Tutorial
- D3 Screencasts
- D3 Mapping Training
- D3 Introductory Training
- D3 Intermediate Training
- D3 Advanced Training
- D3 Corporate Training

DashingD3js.com

- Blog
- About
- Hire Me
- D3 Examples
- D3 Resources
- D3 & Data Viz Newsletter Archive

Data Visualization & D3.js Weekly Newsletter

Get D3.js and Data Visualization news, articles, jobs and more delivered to your inbox every Tuesday:

E-mail

Get the Newsletter

Did you sign up for the newsletter? :)