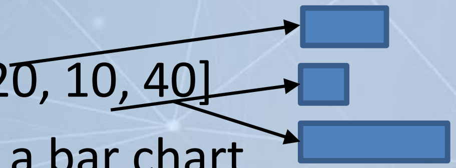
Data Binding (Data Joining)

Data Visualization

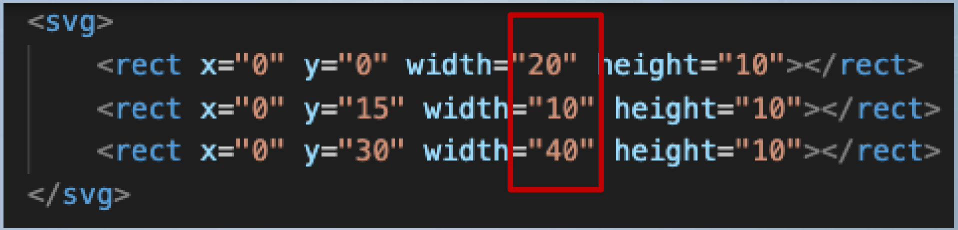
Data-Driven Documents (D3)

- D3 can map data to HTML/SVG elements
  - Construct the DOM from data
- Each data value has a corresponding HTML/SVG elements
  - D3 helps us maintain this mapping

- Data: [20, 10, 40]
- Map to a bar chart



In this example, we want to bind our data to width of the rectangles



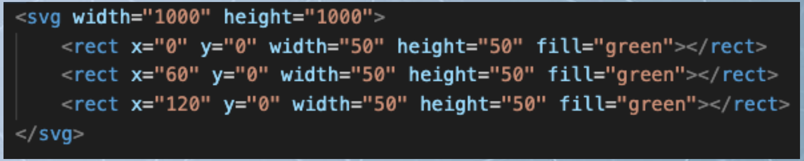
```
<svg>
    <rect x="0" y="0" width="20" height="10"></rect>
    <rect x="0" y="15" width="10" height="10"></rect>
    <rect x="0" y="30" width="40" height="10"></rect>
</svg>
```

Ex03-1

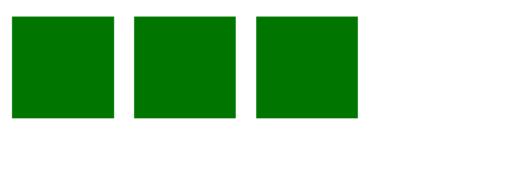
- D3 to update bar chart appearance

- File
  - index.html
  - main.js

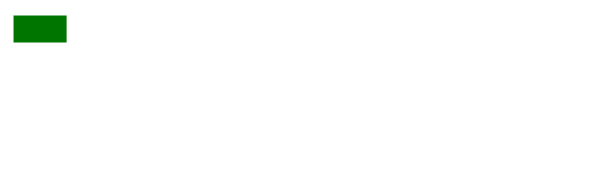
index.html



```
<svg width="1000" height="1000">
    <rect x="0" y="0" width="50" height="50" fill="green"></rect>
    <rect x="60" y="0" width="50" height="50" fill="green"></rect>
    <rect x="120" y="0" width="50" height="50" fill="green"></rect>
</svg>
```



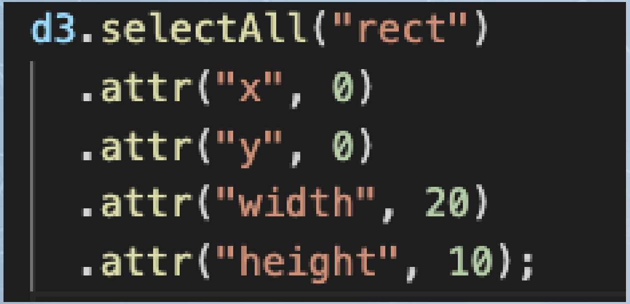
Without main.js



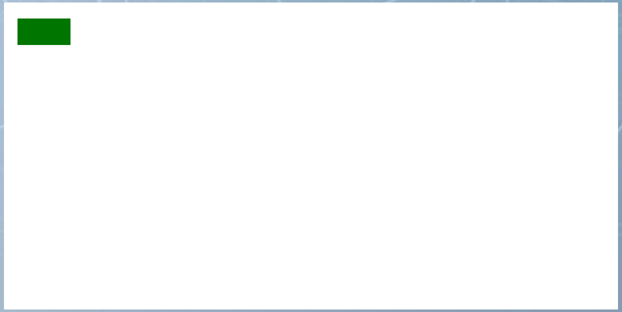
With main.js

Ex03-1

- main.js
- It selects all rectangles (we have three)
  - Set all x=0, y=0, width=20, height=10 to all rectangles
  - So, you only see one rectangle



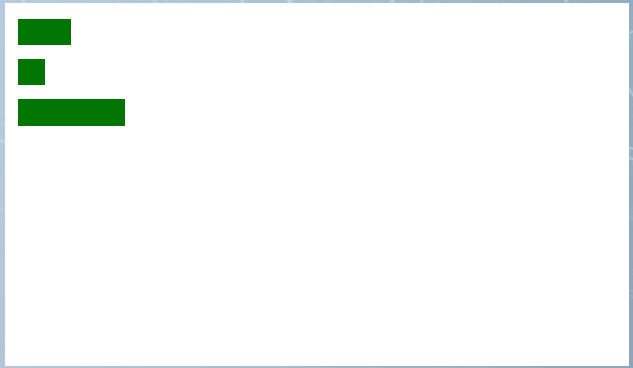
```
d3.selectAll("rect")
    .attr("x", 0)
    .attr("y", 0)
    .attr("width", 20)
    .attr("height", 10);
```



Ex03-2

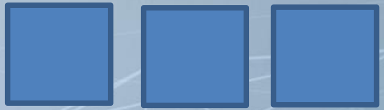
- Add data (a javascript array) [20, 10, 40] in to main.js
- Bind the data array to rectangles and update the appearances of them

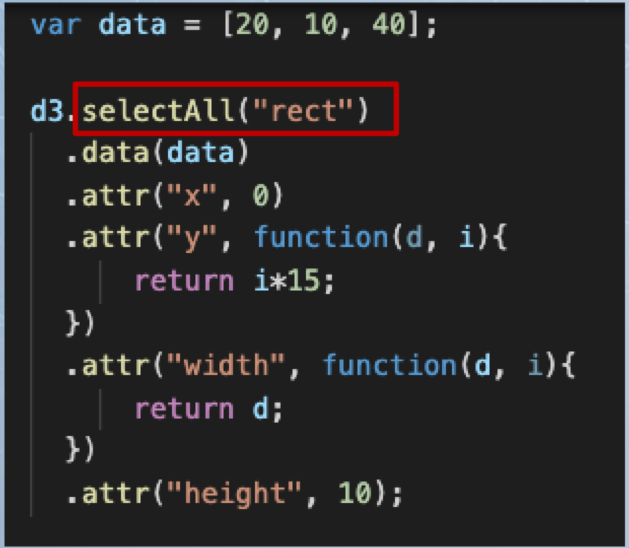
- File
  - index.html
  - main.js



Ex03-2

- main.js

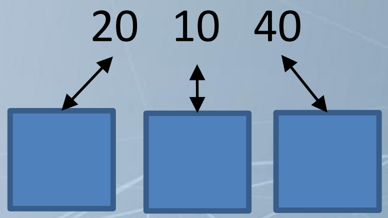


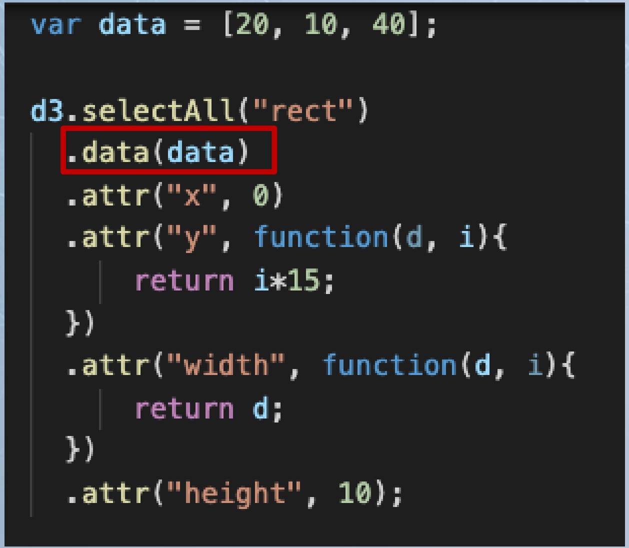


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

Ex03-2

- main.js
  - .data: bind data to elements

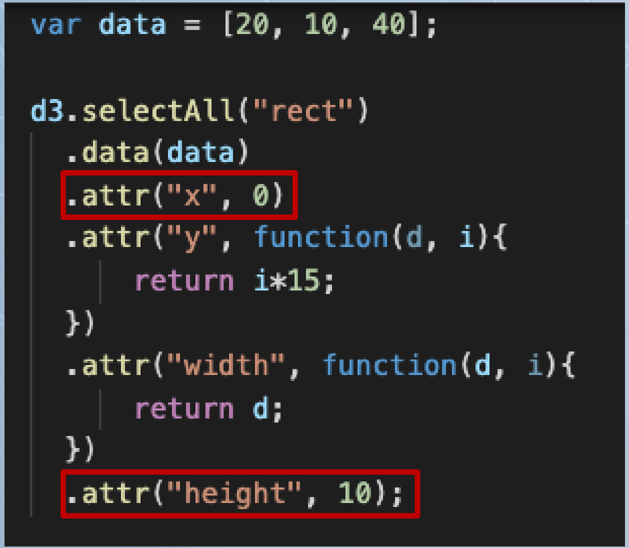




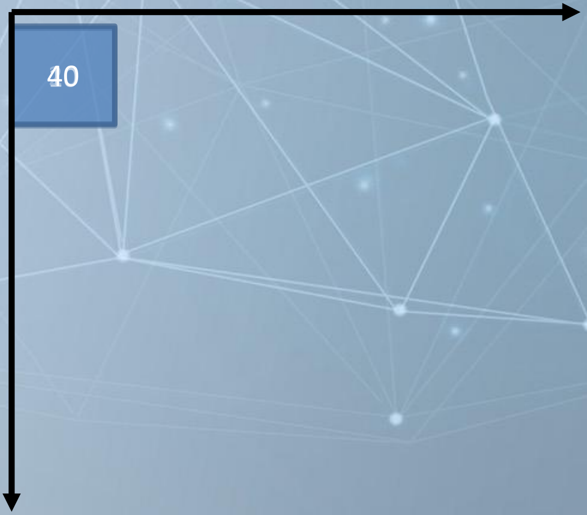
```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

Ex03-2

- main.js

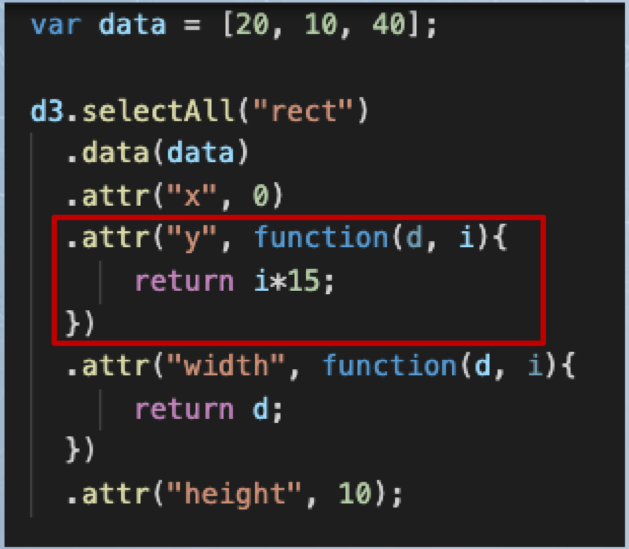


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

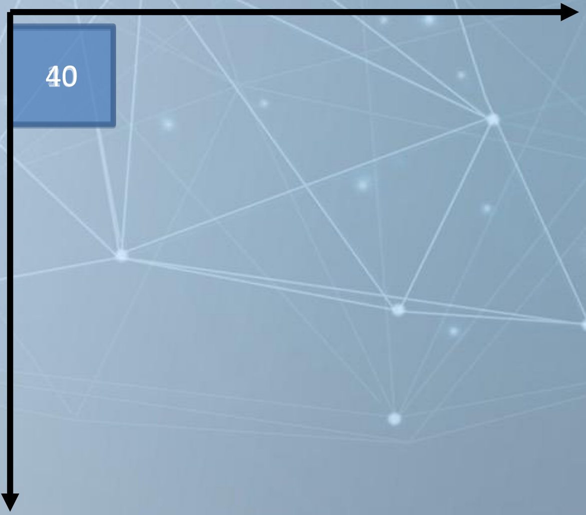


Ex03-2

- main.js
- The function(d, i) iterates through all elements one by one. The first argument(d) is the attached data, the second argument(i) is the index

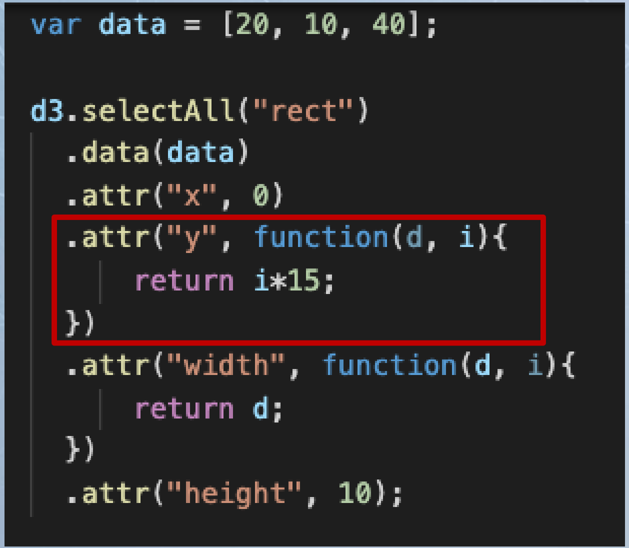


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

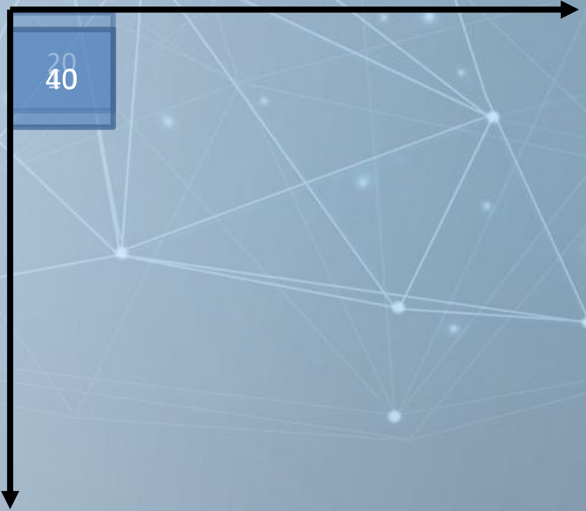


Ex03-2

- main.js
- Ex: the first rectangle attaches data value 20. So, in the iteration for the first rectangle d=20, i=0.
- This line set y of the first rectangle to 0 (0*15)

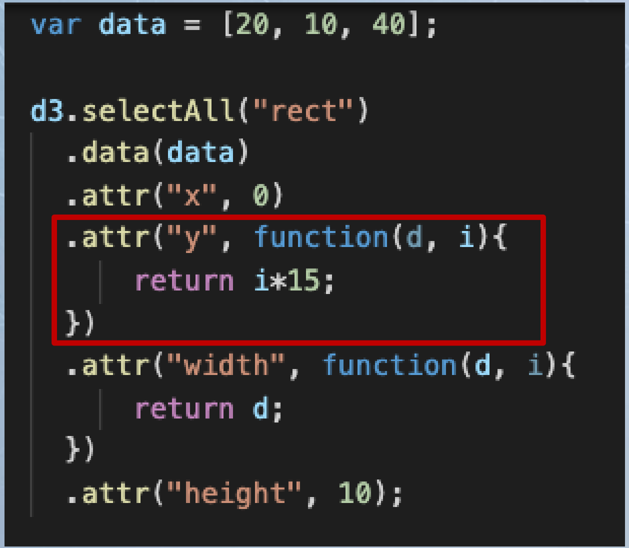


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

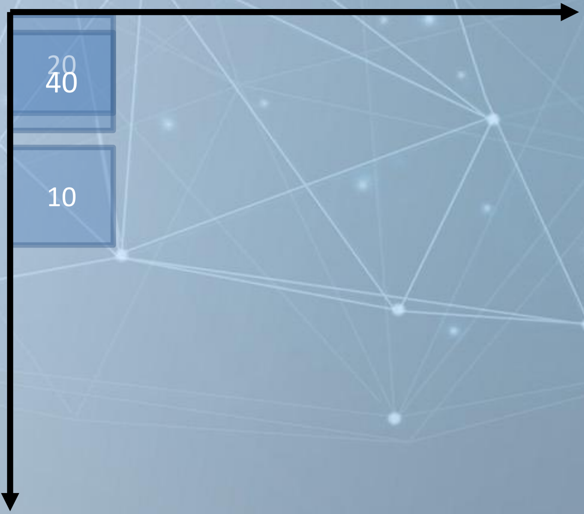


Ex03-2

- main.js
- Ex: the second rectangle attaches data value 20. So, in the iteration for the second rectangle d=10, i=1.
- This line set y of the second rectangle to 15 (1*15)

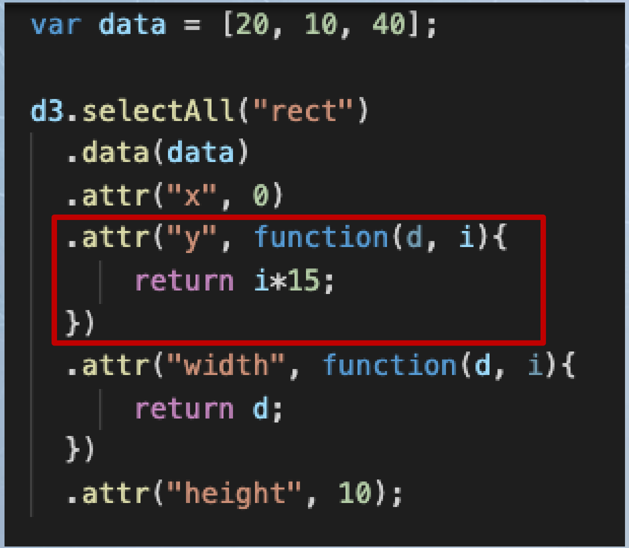


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

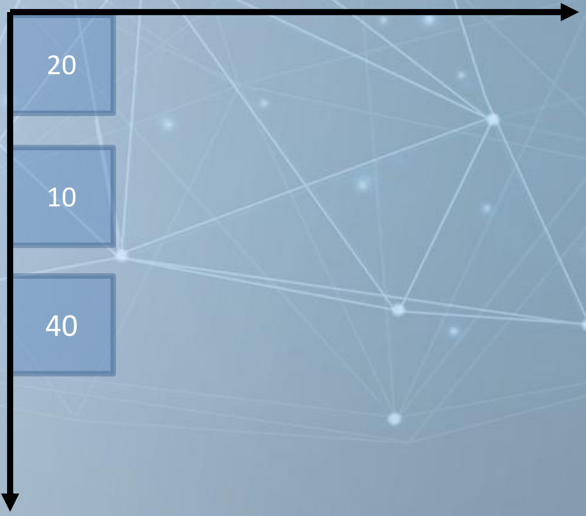


Ex03-2

- main.js
- Ex: the third rectangle attaches data value 40. So, in the iteration for the third rectangle d=40, i=2.
- This line set y of the third rectangle to 30 (2*15)

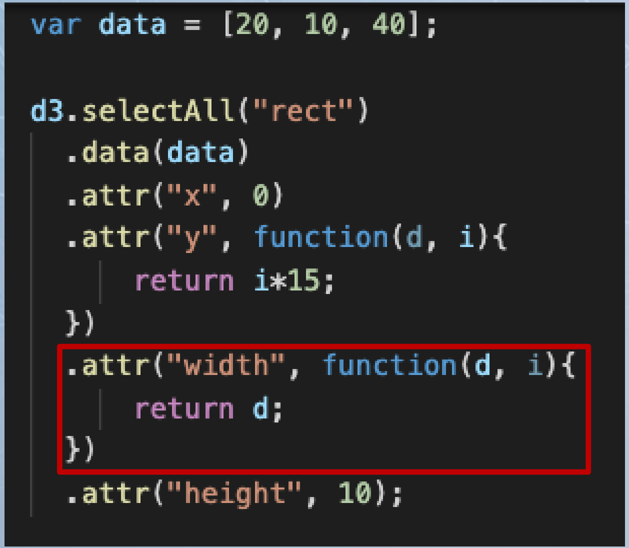


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

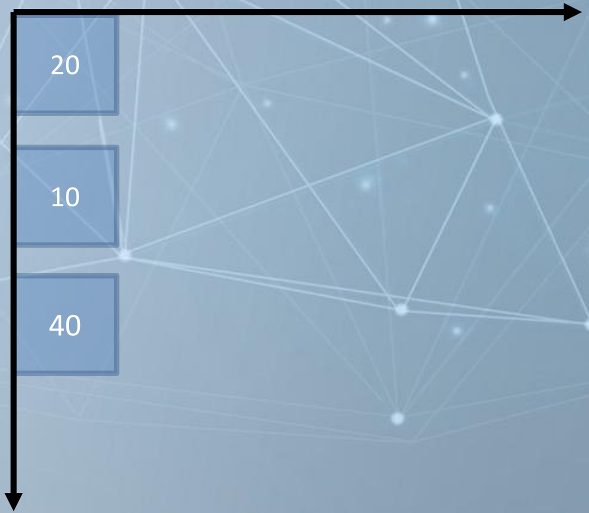


Ex03-2

- main.js
- Ex: the first rectangle attaches data value 20. So, in the iteration for the first rectangle d=20, i=0.
- This line set width of the first rectangle to 20

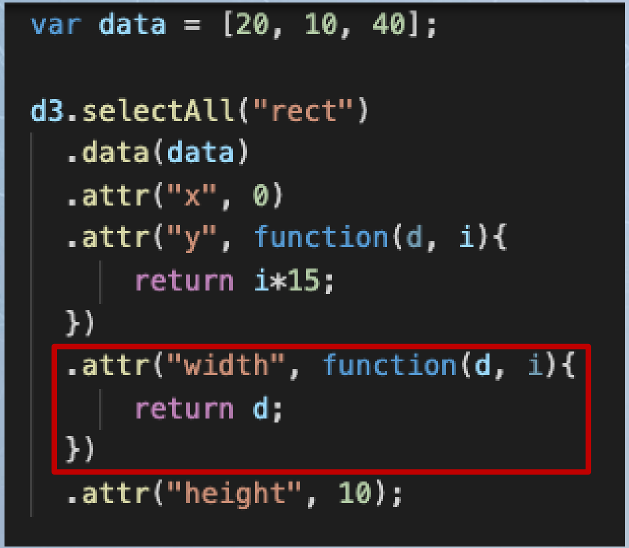


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
    return i*15;
  })
  .attr("width", function(d, i){
    return d;
  })
  .attr("height", 10);
```

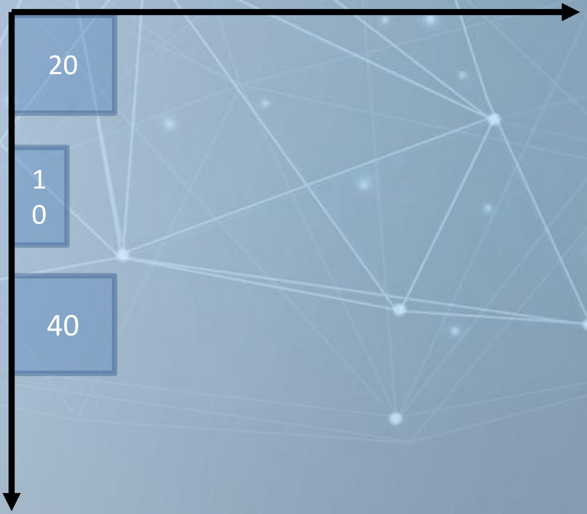


Ex03-2

- main.js
- Ex: the second rectangle attaches data value 10. So, in the iteration for the second rectangle d=10, i=1.
- This line set width of the second rectangle to 10

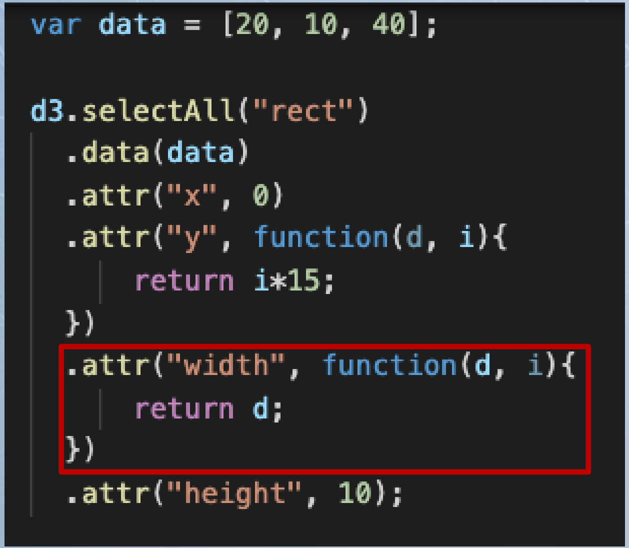


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
     return i*15;
  })
  .attr("width", function(d, i){
     return d;
  })
  .attr("height", 10);
```

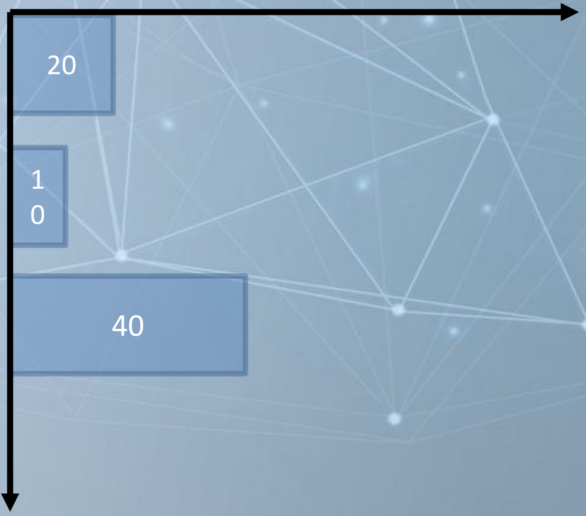


Ex03-2

- main.js
- Ex: the third rectangle attaches data value 40. So, in the iteration for the third rectangle d=40, i=2.
- This line set width of the third rectangle to 40

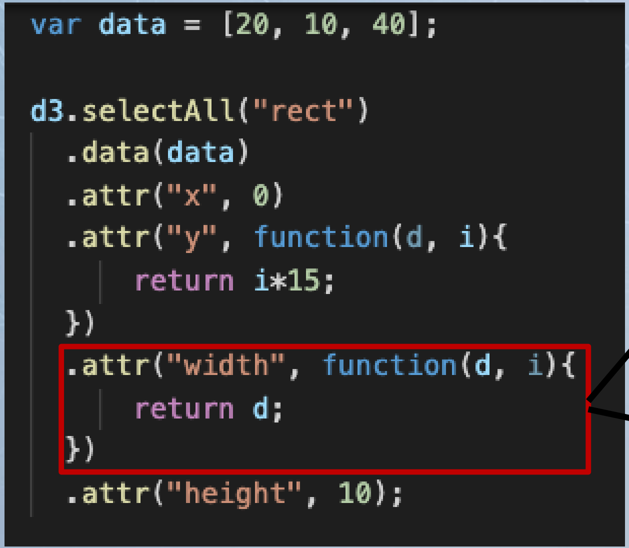


```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
    return i*15;
  })
  .attr("width", function(d, i){
    return d;
  })
  .attr("height", 10);
```



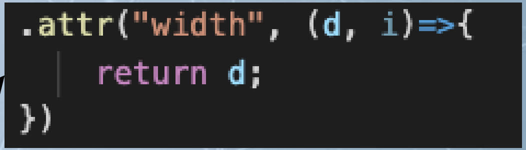
Ex03-2

- main.js
- Arrow function expression
  - A compact alternative to a function expression
- The following three expressions are the same



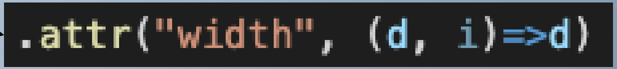
```
var data = [20, 10, 40];

d3.selectAll("rect")
  .data(data)
  .attr("x", 0)
  .attr("y", function(d, i){
    return i*15;
  })
  .attr("width", function(d, i){
    return d;
  })
  .attr("height", 10);
```

You can remove “function” but use “=>”



```
.attr("width", (d, i)=>{
    return d;
})
```

if the return statement is the only line in the function, you can remove “return” and the “curly brackets”



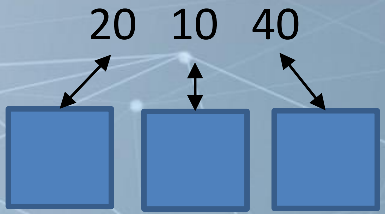
```
.attr("width", (d, i)=>d)
```

Try Ex03-2

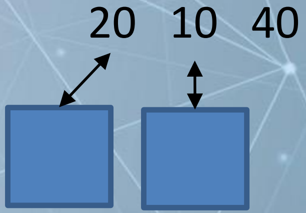
- Run it and make sure you know how data binding works
  - (hard to understand but too important)
- And try the alternative function expressions

D3 Update Pattern

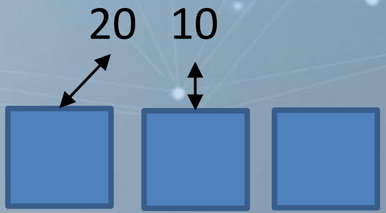
- In Ex02-3, the number of <rect> in index.html is the same of the number of values in data array



- What if the number of <rect> in index.html < the number of values in data array
  - Add elements



- What if the number of <rect> in index.html > the number of values in data array
  - Remove elements



“enter” and “exit”

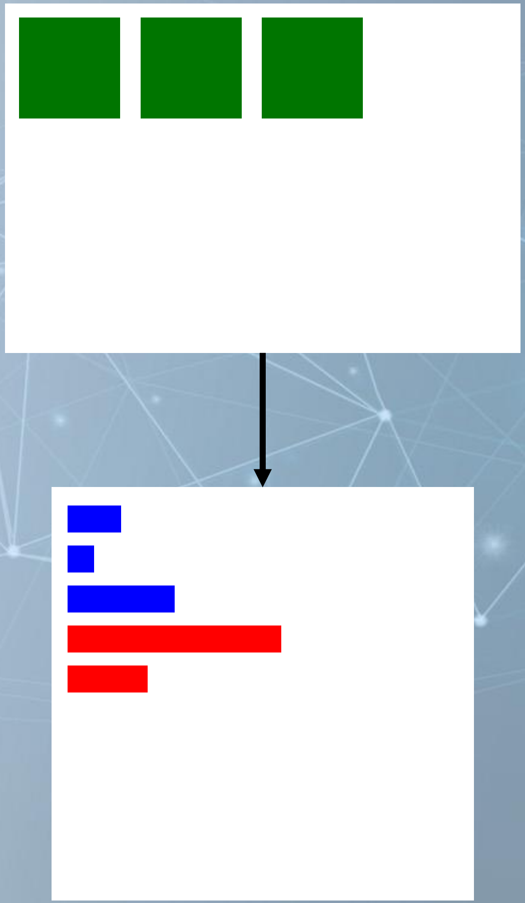
- After you select elements and bind data to them

- D3 automatically determines
  - how many elements should be added
    - **enter**
  - how many elements should be removed
    - **exit**

Ex03-3

- If we have data [20, 10, 40, 80, 30]
- But we only have three rectangles in index.html
- How to use the above data to show five bars (rectangles)
  - Color the old three rects by blue
  - Color the new two rect by red

- We have to get the hold of the old three rectangles and set their x, y, width, height and color by the corresponding data
- Then, append two more rectangles to “svg” and set their x, y, width, height and color by the corresponding data

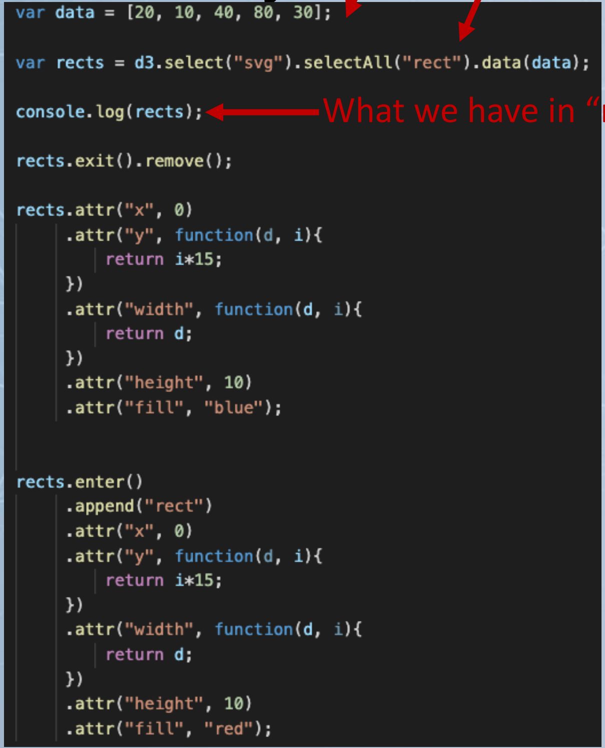


Ex03-3

- main.js

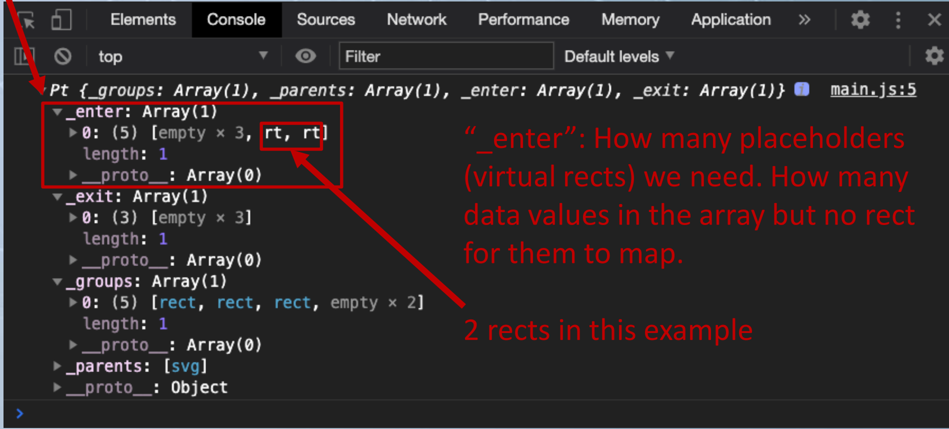
Our data

Select all rectangles from svg and bind the data



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "blue");


rects.enter()
     .append("rect")
     .attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "red");
```

What we have in “rects”



```
Pt {_groups: Array(1), _parents: Array(1), _enter: Array(1), _exit: Array(1)}   main.js:5
 ▼_enter: Array(1)
   ▶0: (5) [empty × 3, rt, rt]
    length: 1
   ▶__proto__: Array(0)
 ▼_exit: Array(1)
   ▶0: (3) [empty × 3]
    length: 1
   ▶__proto__: Array(0)
 ▼_groups: Array(1)
   ▶0: (5) [rect, rect, rect, empty × 2]
    length: 1
   ▶__proto__: Array(0)
 ▶_parents: [svg]
 ▶__proto__: Object
```

“_enter”: How many placeholders (virtual rects) we need. How many data values in the array but no rect for them to map.

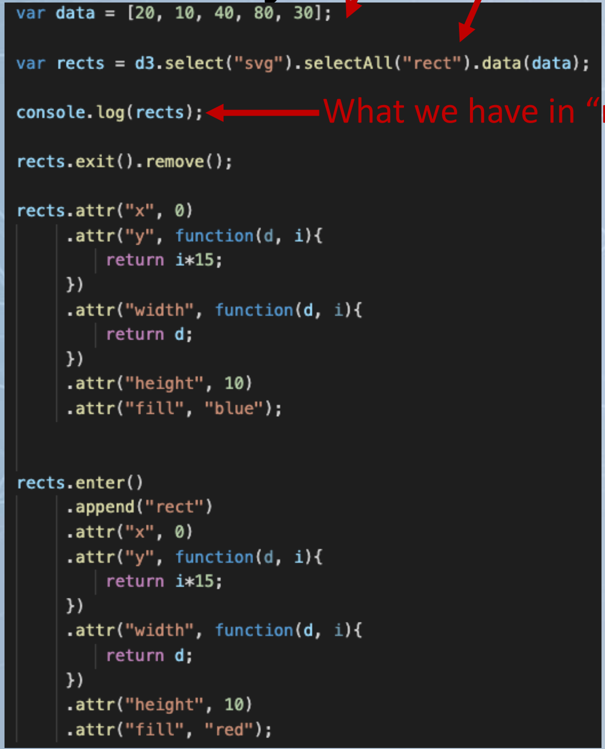
2 rects in this example

Ex03-3

Select all rectangles from svg and bind the data

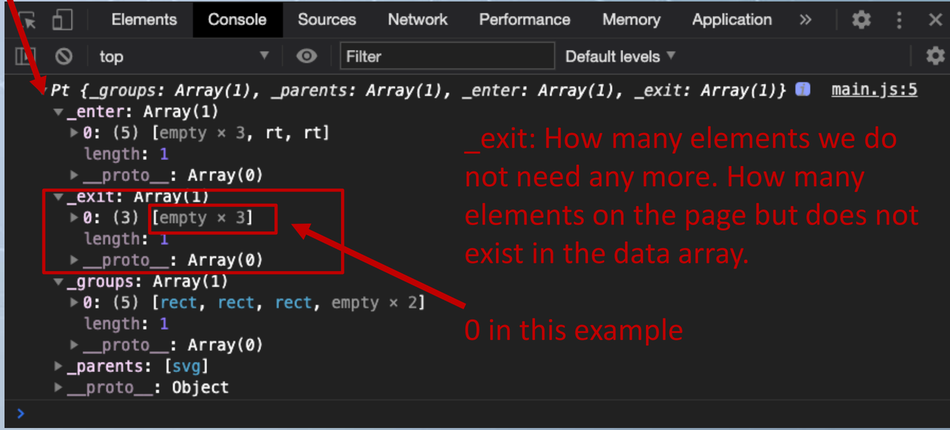
Our data

- main.js



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "blue");


rects.enter()
     .append("rect")
     .attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "red");
```

What we have in “rects”



```
▾Pt {_groups: Array(1), _parents: Array(1), _enter: Array(1), _exit: Array(1)}   main.js:5
  ▾_enter: Array(1)
    ▸0: (5) [empty × 3, rt, rt]
     length: 1
    ▸__proto__: Array(0)
  ▾_exit: Array(1)
    ▸0: (3) [empty × 3]
     length: 1
    ▸__proto__: Array(0)
  ▾_groups: Array(1)
    ▸0: (5) [rect, rect, rect, empty × 2]
     length: 1
    ▸__proto__: Array(0)
  ▸_parents: [svg]
  ▸__proto__: Object
>
```

_exit: How many elements we do not need any more. How many elements on the page but does not exist in the data array.

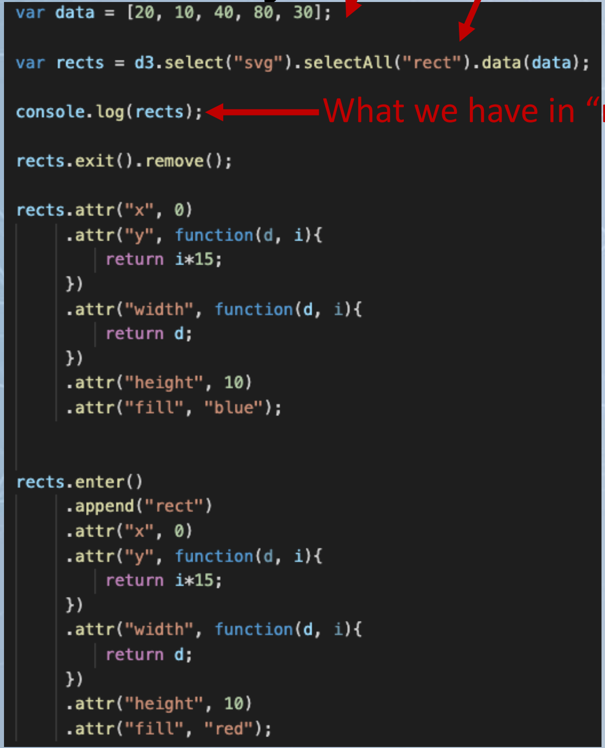
0 in this example

Ex03-3

- main.js

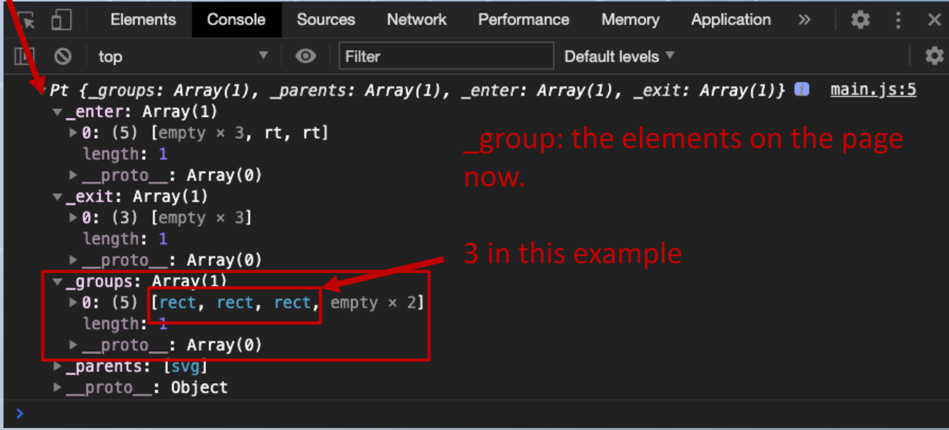
Our data

Select all rectangles from svg and bind the data



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "blue");


rects.enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "red");
```

What we have in “rects”



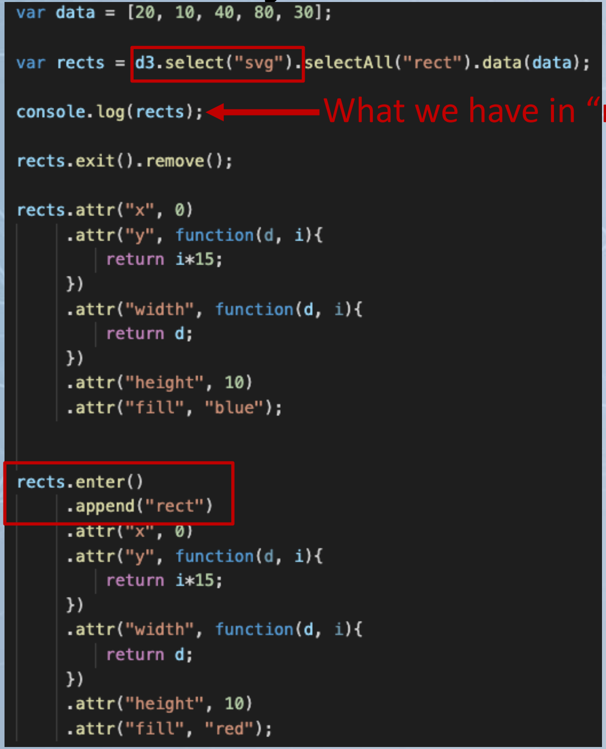
```
Pt {_groups: Array(1), _parents: Array(1), _enter: Array(1), _exit: Array(1)}   main.js:5
 ▼_enter: Array(1)
   ▶0: (5) [empty × 3, rt, rt]
    length: 1
   ▶__proto__: Array(0)
 ▼_exit: Array(1)
   ▶0: (3) [empty × 3]
    length: 1
   ▶__proto__: Array(0)
 ▼_groups: Array(1)
   ▶0: (5) [rect, rect, rect, empty × 2]
    length: 1
   ▶__proto__: Array(0)
 ▶_parents: [svg]
 ▶__proto__: Object
>
```

_group: the elements on the page now.

3 in this example

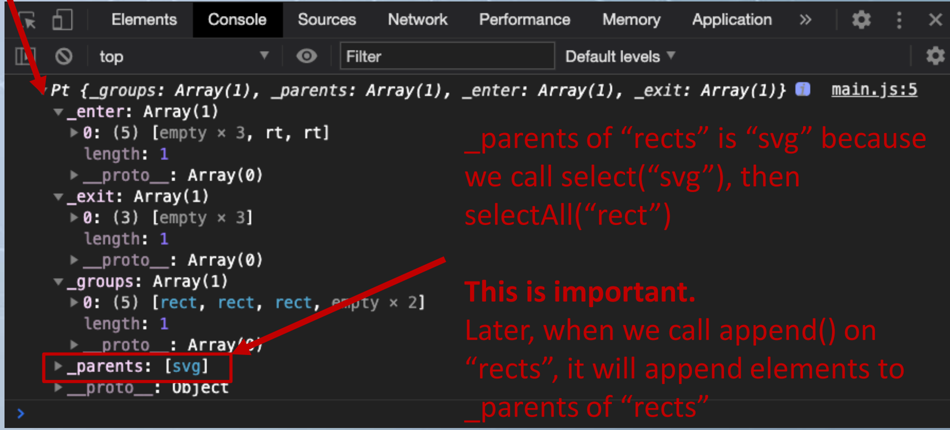
Ex03-3

- main.js



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "blue");


rects.enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "red");
```

What we have in "rects"



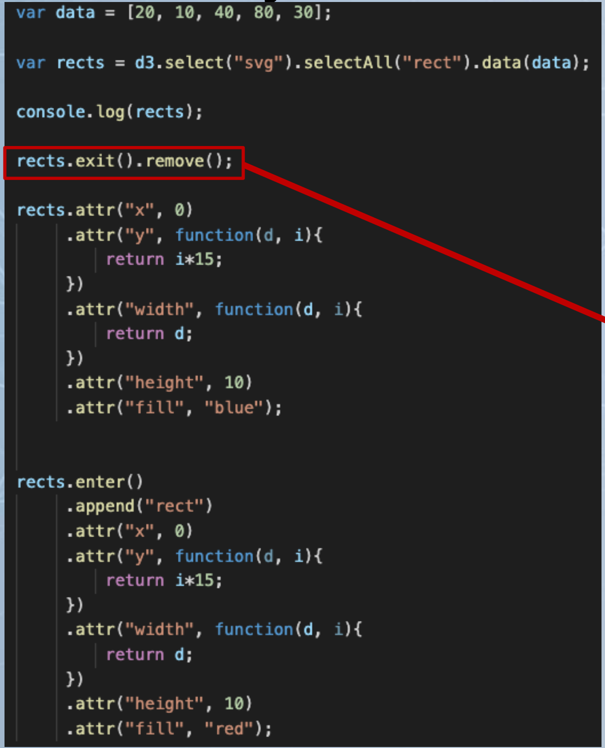
```
▾Pt {_groups: Array(1), _parents: Array(1), _enter: Array(1), _exit: Array(1)}  main.js:5
  ▾_enter: Array(1)
    ▸0: (5) [empty × 3, rt, rt]
     length: 1
    ▸__proto__: Array(0)
  ▾_exit: Array(1)
    ▸0: (3) [empty × 3]
     length: 1
    ▸__proto__: Array(0)
  ▾_groups: Array(1)
    ▸0: (5) [rect, rect, rect, empty × 2]
     length: 1
    ▸__proto__: Array(0)
  ▸_parents: [svg]
  ▸__proto__: Object
>
```

_parents of "rects" is "svg" because we call select("svg"), then selectAll("rect")

This is important.
Later, when we call append() on "rects", it will append elements to _parents of "rects"

Ex03-3

- main.js

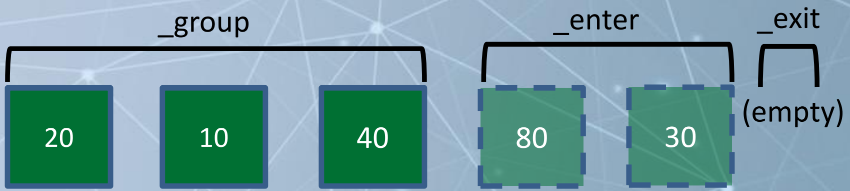


```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "blue");


rects.enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "red");
```

D3 Update Pattern

After binding the data

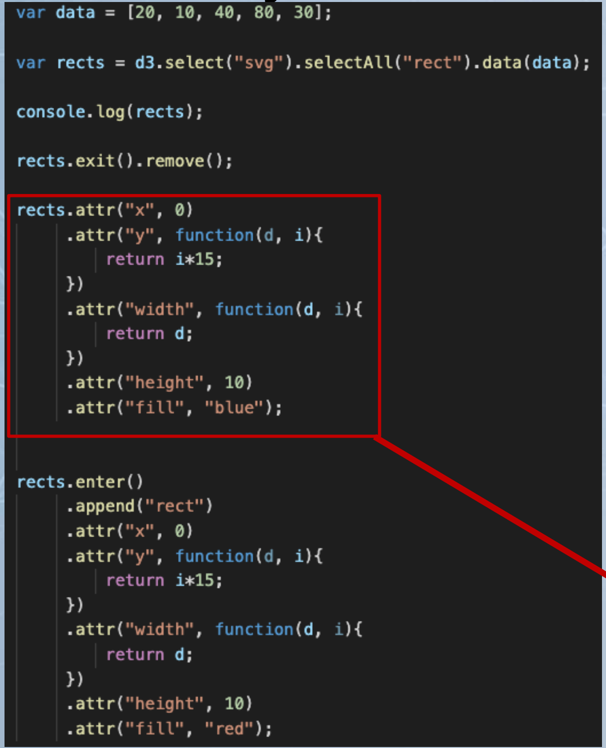
1. **Exit – use exit() to remove the elements we do not need**
2. Update – update the attributes of the existing elements
3. Enter – use enter() to append(add) new elements and set their attributes



Remove nothing in this example

Ex03-3

- main.js

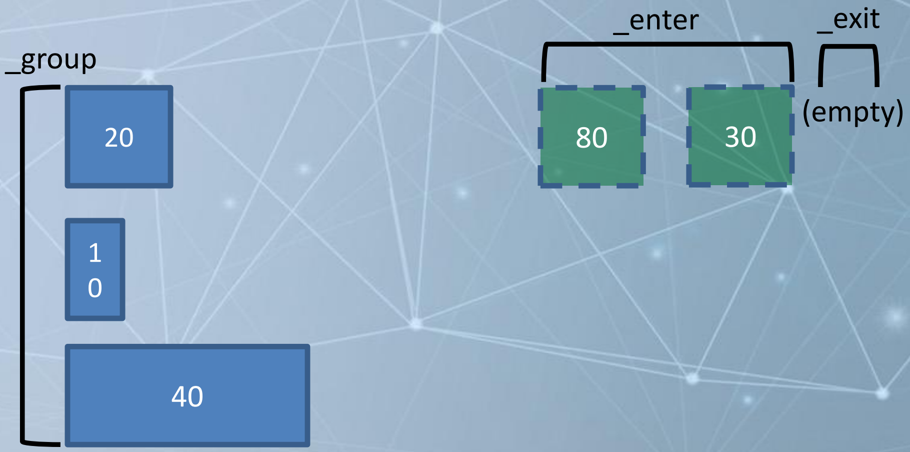


```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "blue");


rects.enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "red");
```

D3 Update Pattern

After binding the data

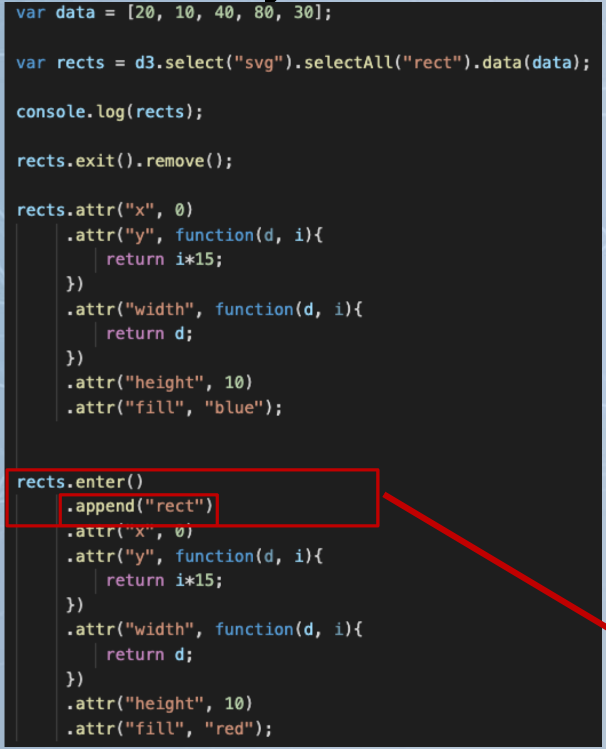
1. Exit – use exit() to remove the elements we do not need
2. **Update – update the attributes of the existing elements**
3. Enter – use enter() to append(add) new elements and set their attributes



"rects" indicates "_group"

Ex03-3

- main.js

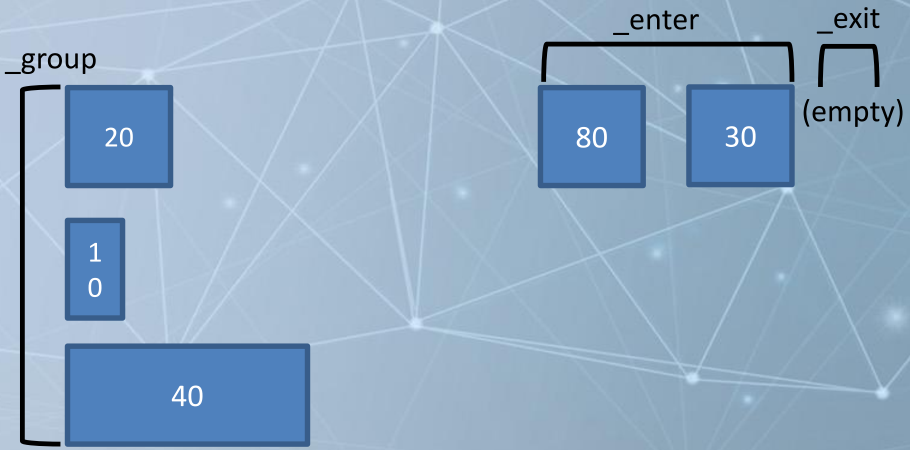


```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "blue");


rects.enter()
     .append("rect")
     .attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "red");
```

D3 Update Pattern

After binding the data

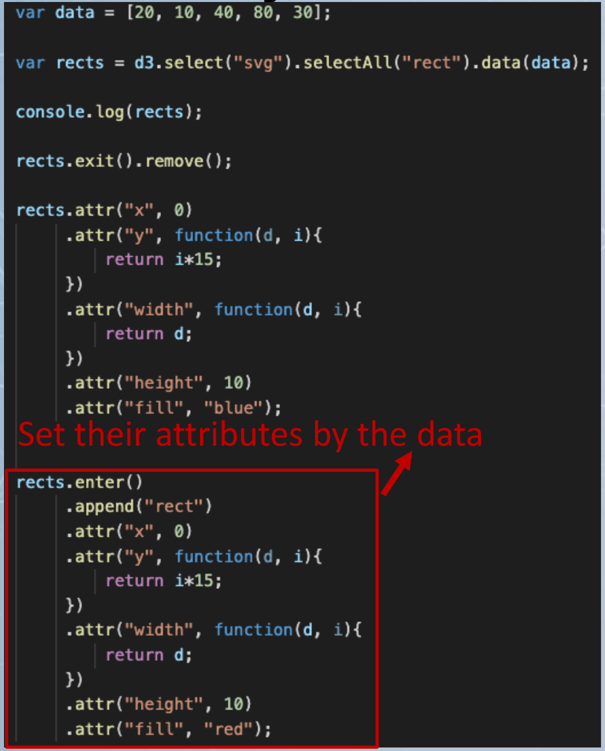
1. Exit – use exit() to remove the elements we do not need
2. Update – update the attributes of the existing elements
3. **Enter – use enter() to append(add) new elements and set their attributes**



Remember to append() first. Without append(), they does not exist in DOM (webpage)

Ex03-3

- main.js



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "blue");
```

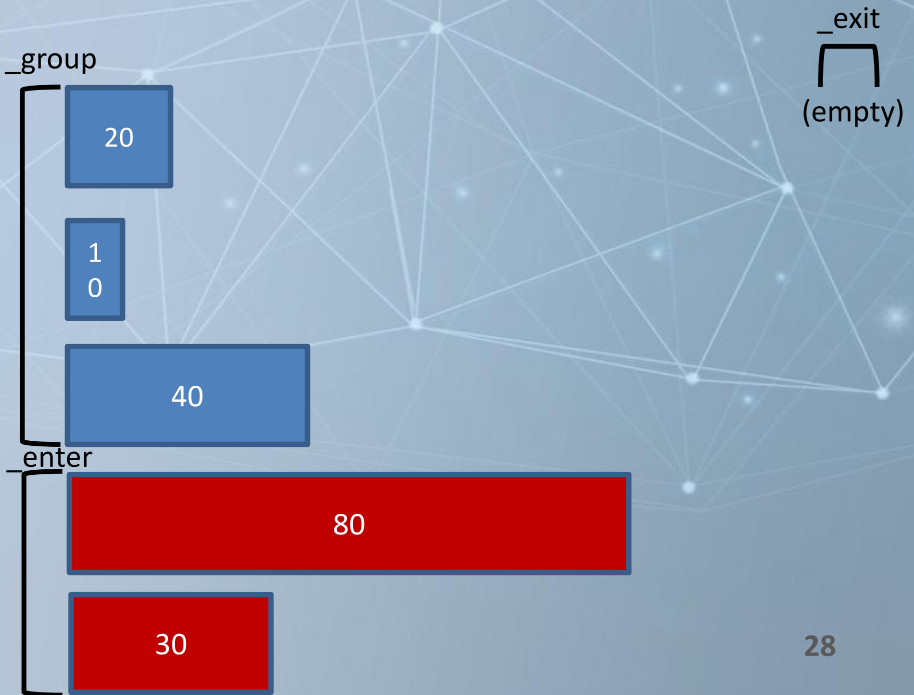
Set their attributes by the data

```
rects.enter()
     .append("rect")
     .attr("x", 0)
     .attr("y", function(d, i){
         return i*15;
     })
     .attr("width", function(d, i){
         return d;
     })
     .attr("height", 10)
     .attr("fill", "red");
```

D3 Update Pattern

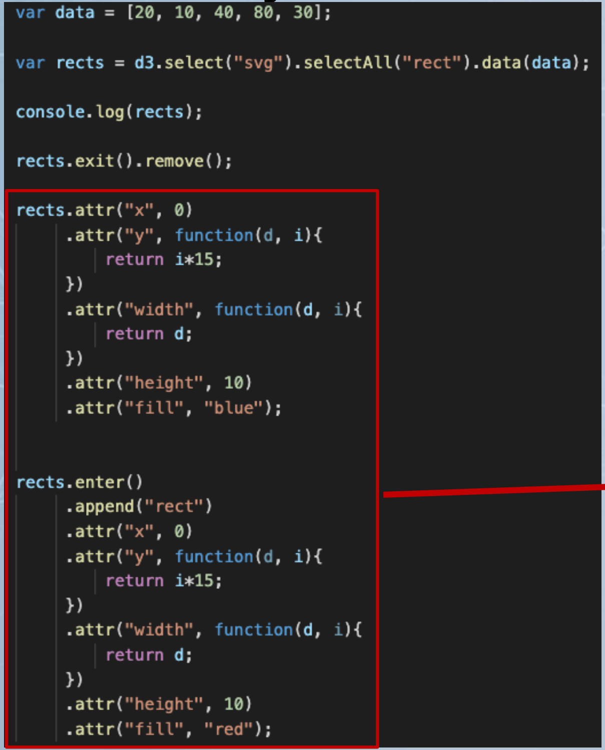
After binding the data

1. Exit – use exit() to remove the elements we do not need
2. Update – update the attributes of the existing elements
3. **Enter – use enter() to append(add) new elements and set their attributes**



Ex03-3

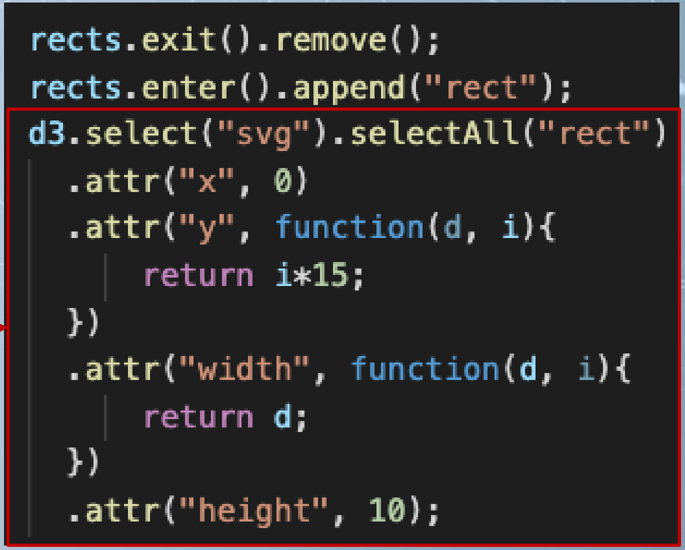
- main.js



```
var data = [20, 10, 40, 80, 30];

var rects = d3.select("svg").selectAll("rect").data(data);

console.log(rects);

rects.exit().remove();

rects.attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "blue");


rects.enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", function(d, i){
        return i*15;
    })
    .attr("width", function(d, i){
        return d;
    })
    .attr("height", 10)
    .attr("fill", "red");
```

if we do not care about the “color” in this example.
After exit() and enter().append(), we can simply selectAll(“rect”) again from the svg and update the rects’ attributes by the data

(this an alternative way to set the attributes, but the code is shorter)



```
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 0)
  .attr("y", function(d, i){
      return i*15;
  })
  .attr("width", function(d, i){
      return d;
  })
  .attr("height", 10);
```

Try it

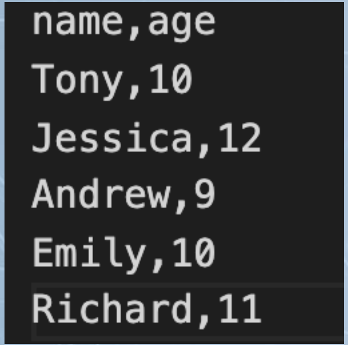
- Run it

- Review the code and check the execution result

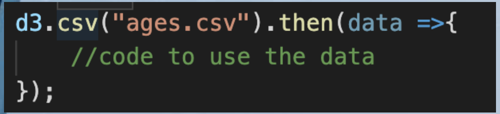
- Data binding and updating might the most important part in D3. But, this might be also the most difficult part to be understood.
  - If you have any question, ask now.

Load External File

.csv
(comma separated values)

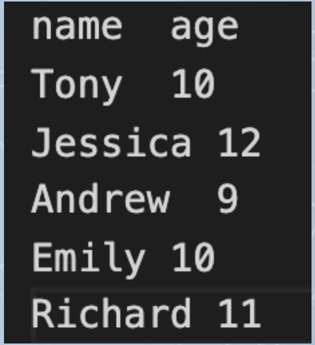


```
name,age
Tony,10
Jessica,12
Andrew,9
Emily,10
Richard,11
```

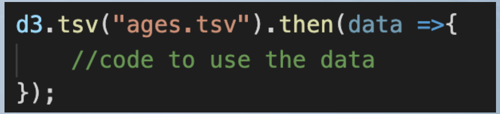


```
d3.csv("ages.csv").then(data =>{
    //code to use the data
});
```

.tsv
(tab separated values)

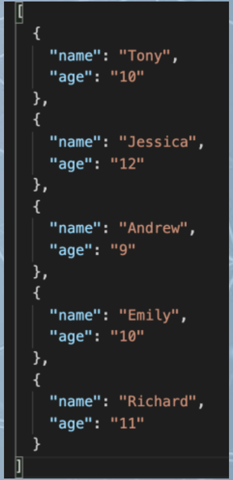


```
name	age
Tony	10
Jessica	12
Andrew	9
Emily	10
Richard	11
```

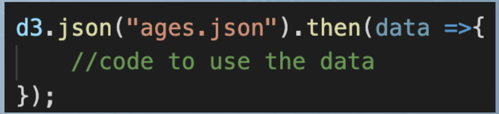


```
d3.tsv("ages.tsv").then(data =>{
    //code to use the data
});
```

.json
(Javascript object Notation)

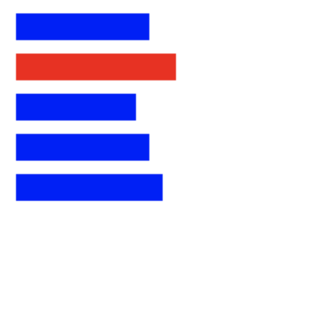


```
[
  {
    "name": "Tony",
    "age": "10"
  },
  {
    "name": "Jessica",
    "age": "12"
  },
  {
    "name": "Andrew",
    "age": "9"
  },
  {
    "name": "Emily",
    "age": "10"
  },
  {
    "name": "Richard",
    "age": "11"
  }
]
```



```
d3.json("ages.json").then(data =>{
    //code to use the data
});
```

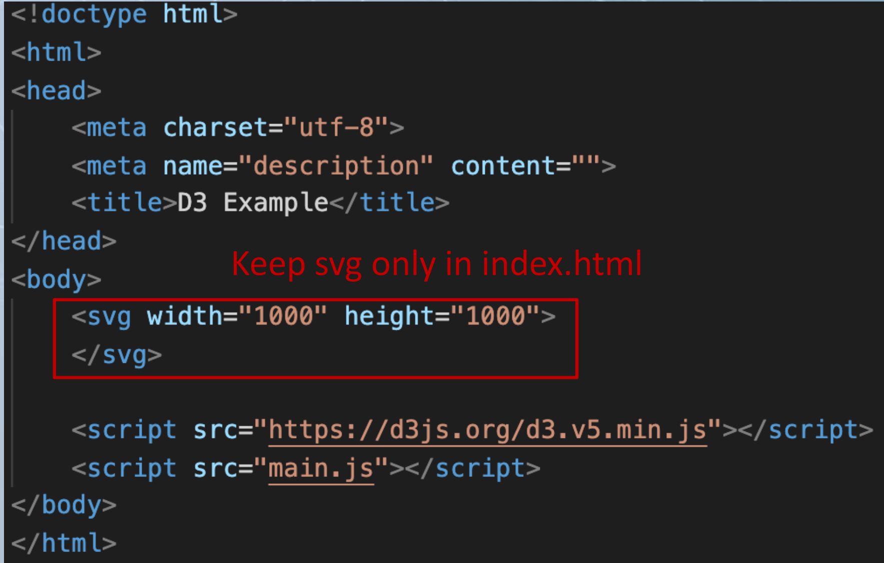
Ex03-4



- Load age.csv file and draw bars

- File
  - index.html
  - main.js

index.html



```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="description" content="">
    <title>D3 Example</title>
</head>
<body>
    <svg width="1000" height="1000">
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="main.js"></script>
</body>
</html>
```

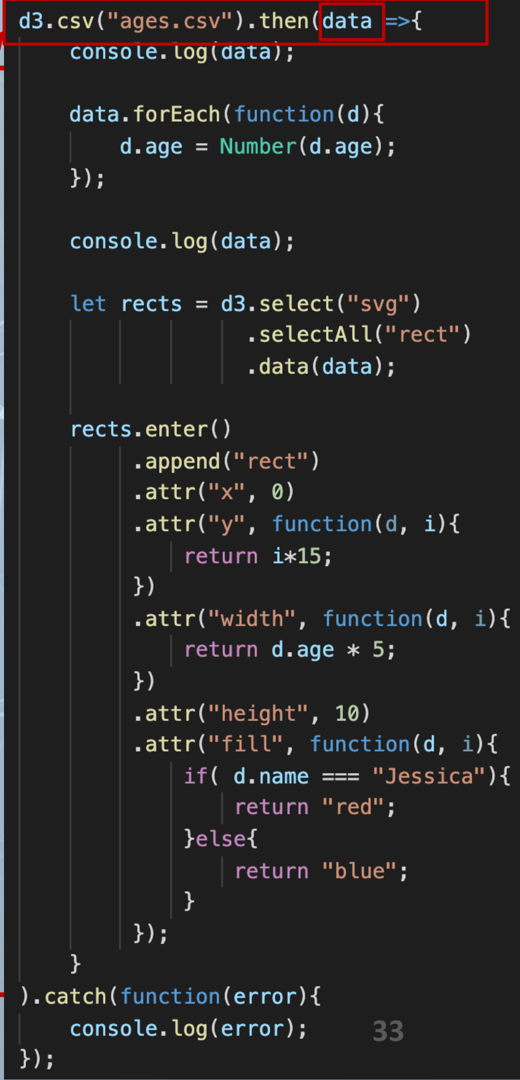
Keep svg only in index.html

Ex03-4

- main.js

If d3 loads the file successfully, it put the data in the variable "data"

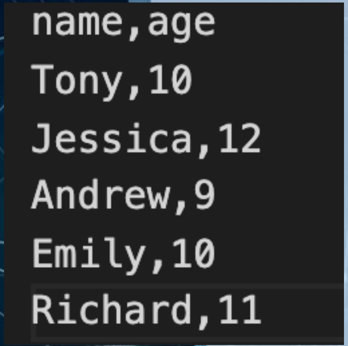
Code to use/visualize the data



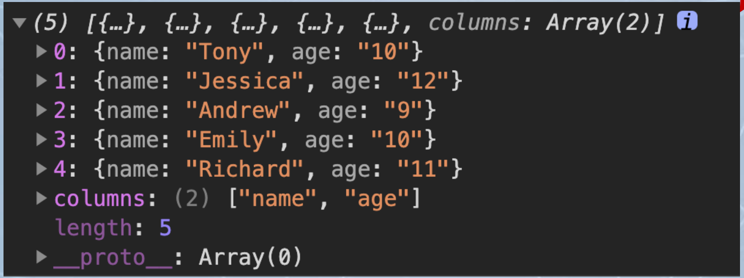
```
d3.csv("ages.csv").then(data =>{
    console.log(data);

    data.forEach(function(d){
        d.age = Number(d.age);
    });

    console.log(data);

    let rects = d3.select("svg")
                  .selectAll("rect")
                  .data(data);

    rects.enter()
        .append("rect")
        .attr("x", 0)
        .attr("y", function(d, i){
            return i*15;
        })
        .attr("width", function(d, i){
            return d.age * 5;
        })
        .attr("height", 10)
        .attr("fill", function(d, i){
            if( d.name === "Jessica"){
                return "red";
            }else{
                return "blue";
            }
        });
    }
).catch(function(error){
    console.log(error);
});
```

Ex03-4

- main.js

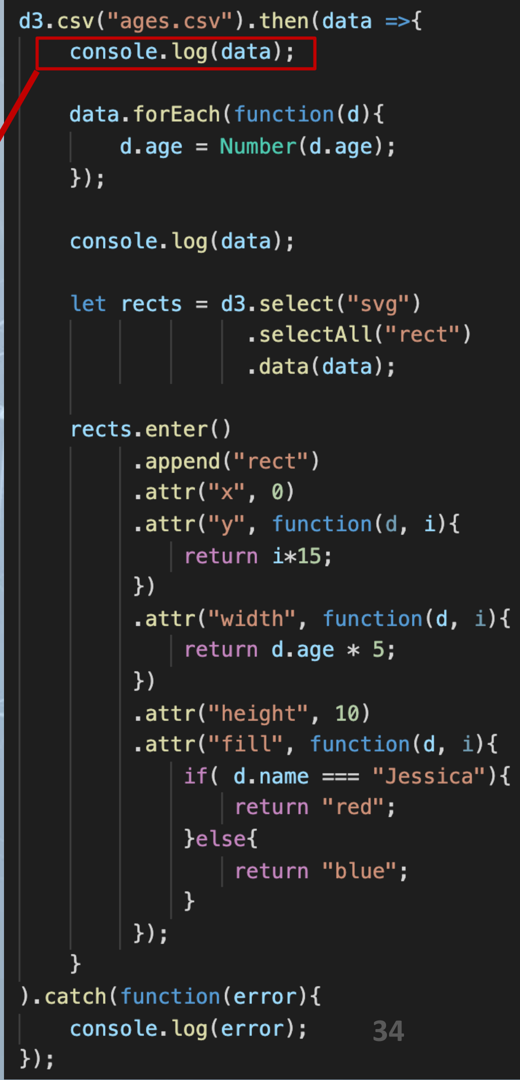


```
name,age
Tony,10
Jessica,12
Andrew,9
Emily,10
Richard,11
```



```
▼ (5) [{…}, {…}, {…}, {…}, {…}, columns: Array(2)] i
  ▶ 0: {name: "Tony", age: "10"}
  ▶ 1: {name: "Jessica", age: "12"}
  ▶ 2: {name: "Andrew", age: "9"}
  ▶ 3: {name: "Emily", age: "10"}
  ▶ 4: {name: "Richard", age: "11"}
  ▶ columns: (2) ["name", "age"]
    length: 5
  ▶ __proto__: Array(0)
```

What the “data” looks like
(array of dictionary and all strings)

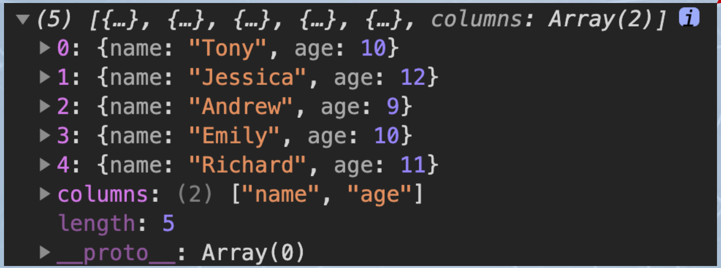


```
d3.csv("ages.csv").then(data =>{
    console.log(data);

    data.forEach(function(d){
        d.age = Number(d.age);
    });

    console.log(data);

    let rects = d3.select("svg")
                  .selectAll("rect")
                  .data(data);

    rects.enter()
        .append("rect")
        .attr("x", 0)
        .attr("y", function(d, i){
            return i*15;
        })
        .attr("width", function(d, i){
            return d.age * 5;
        })
        .attr("height", 10)
        .attr("fill", function(d, i){
            if( d.name === "Jessica"){
                return "red";
            }else{
                return "blue";
            }
        });
    }
).catch(function(error){
    console.log(error);
});
```

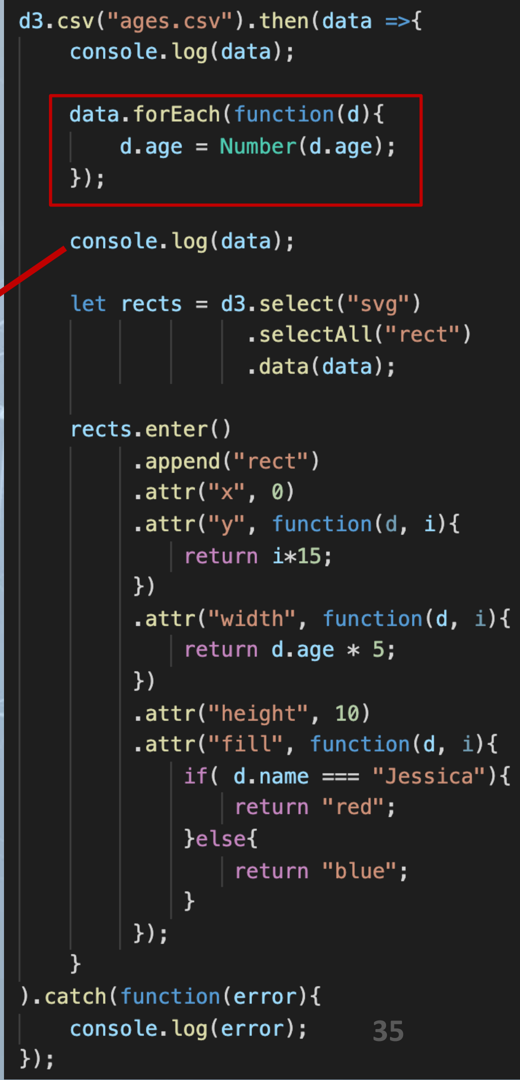
Ex03-4

- main.js

A way to iterate through all data and convert some attributes to "number"



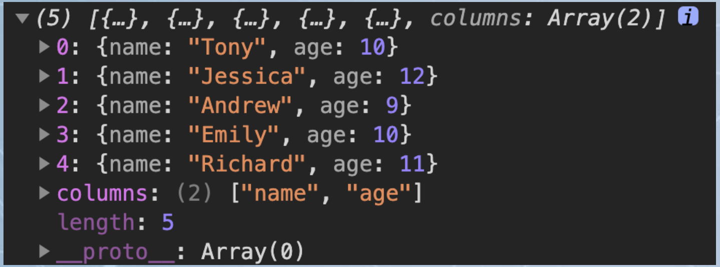
```
▼(5) [{…}, {…}, {…}, {…}, {…}, columns: Array(2)] i
  ▶0: {name: "Tony", age: 10}
  ▶1: {name: "Jessica", age: 12}
  ▶2: {name: "Andrew", age: 9}
  ▶3: {name: "Emily", age: 10}
  ▶4: {name: "Richard", age: 11}
  ▶columns: (2) ["name", "age"]
   length: 5
  ▶__proto__: Array(0)
```



```
d3.csv("ages.csv").then(data =>{
    console.log(data);

    data.forEach(function(d){
        d.age = Number(d.age);
    });

    console.log(data);

    let rects = d3.select("svg")
                  .selectAll("rect")
                  .data(data);

    rects.enter()
         .append("rect")
         .attr("x", 0)
         .attr("y", function(d, i){
             return i*15;
         })
         .attr("width", function(d, i){
             return d.age * 5;
         })
         .attr("height", 10)
         .attr("fill", function(d, i){
             if( d.name === "Jessica"){
                 return "red";
             }else{
                 return "blue";
             }
         });
    }
).catch(function(error){
    console.log(error);
});
```

Ex03-4

- main.js

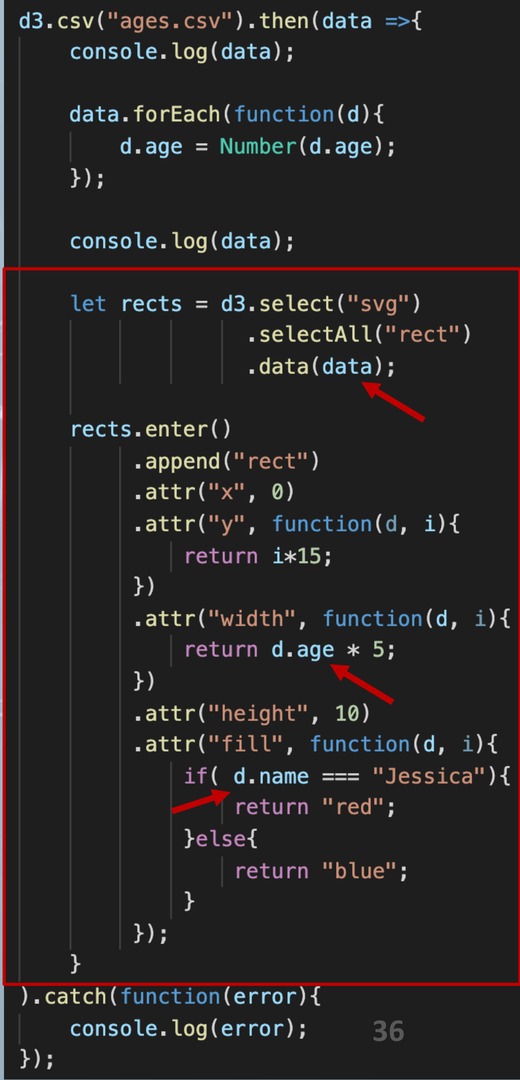


```
▼ (5) [{…}, {…}, {…}, {…}, {…}, columns: Array(2)] i
  ▶ 0: {name: "Tony", age: 10}
  ▶ 1: {name: "Jessica", age: 12}
  ▶ 2: {name: "Andrew", age: 9}
  ▶ 3: {name: "Emily", age: 10}
  ▶ 4: {name: "Richard", age: 11}
  ▶ columns: (2) ["name", "age"]
    length: 5
  ▶ __proto__: Array(0)
```

Add rects to the svg

“age” in the data determines lengths of bars

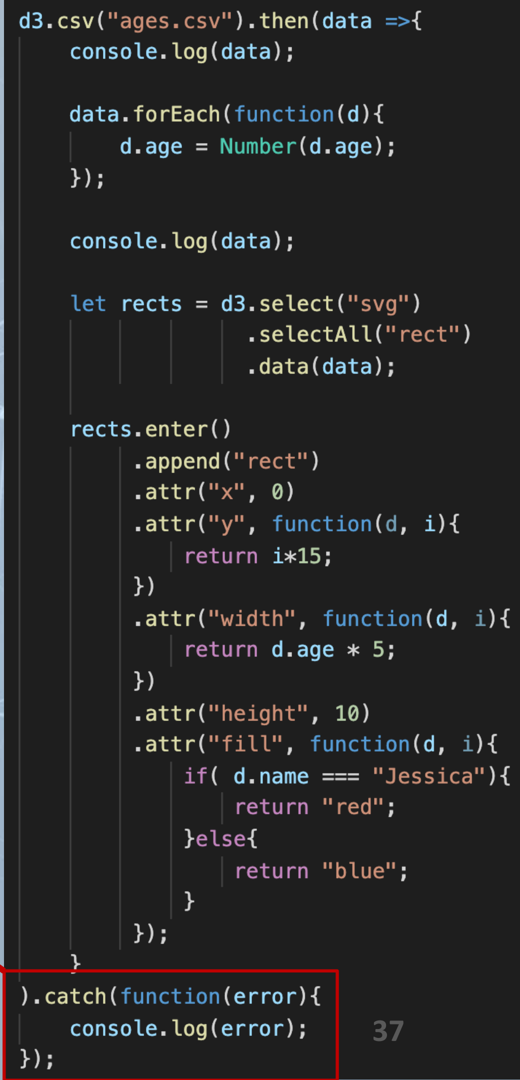
“name” to determine bar colors



```
d3.csv("ages.csv").then(data =>{
    console.log(data);

    data.forEach(function(d){
        d.age = Number(d.age);
    });

    console.log(data);

    let rects = d3.select("svg")
                  .selectAll("rect")
                  .data(data);

    rects.enter()
        .append("rect")
        .attr("x", 0)
        .attr("y", function(d, i){
            return i*15;
        })
        .attr("width", function(d, i){
            return d.age * 5;
        })
        .attr("height", 10)
        .attr("fill", function(d, i){
            if( d.name === "Jessica"){
                return "red";
            }else{
                return "blue";
            }
        });
    }
).catch(function(error){
    console.log(error);
});
```

Ex03-4

- main.js



```
d3.csv("ages.csv").then(data =>{
  console.log(data);

  data.forEach(function(d){
    d.age = Number(d.age);
  });

  console.log(data);

  let rects = d3.select("svg")
                .selectAll("rect")
                .data(data);

  rects.enter()
     .append("rect")
     .attr("x", 0)
     .attr("y", function(d, i){
       return i*15;
     })
     .attr("width", function(d, i){
       return d.age * 5;
     })
     .attr("height", 10)
     .attr("fill", function(d, i){
       if( d.name === "Jessica"){
         return "red";
       }else{
         return "blue";
       }
     });
  }
).catch(function(error){
  console.log(error);
});
```

Handle the error: e.g. if d3 cannot find the file (the error message is stored in “error” variable)

Load External File (old version)

- Before v5.x (callback)
  - You might see the following code to load external file

```
d3.csv(“ages.csv”, function(data){
                //code to process data
})
```