D3 Data Binding

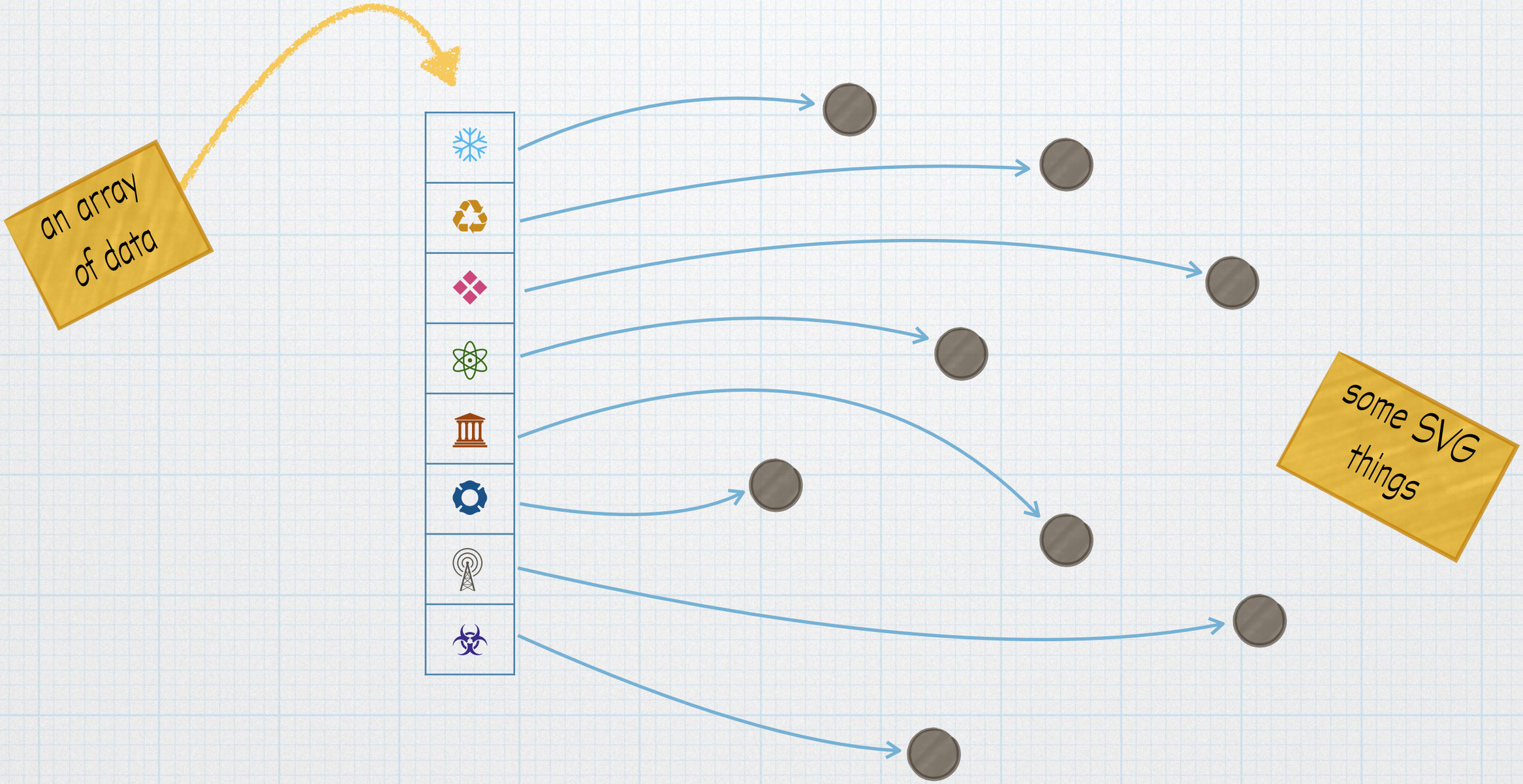
select | enter | exit

Scenario One

In the Beginning. . .

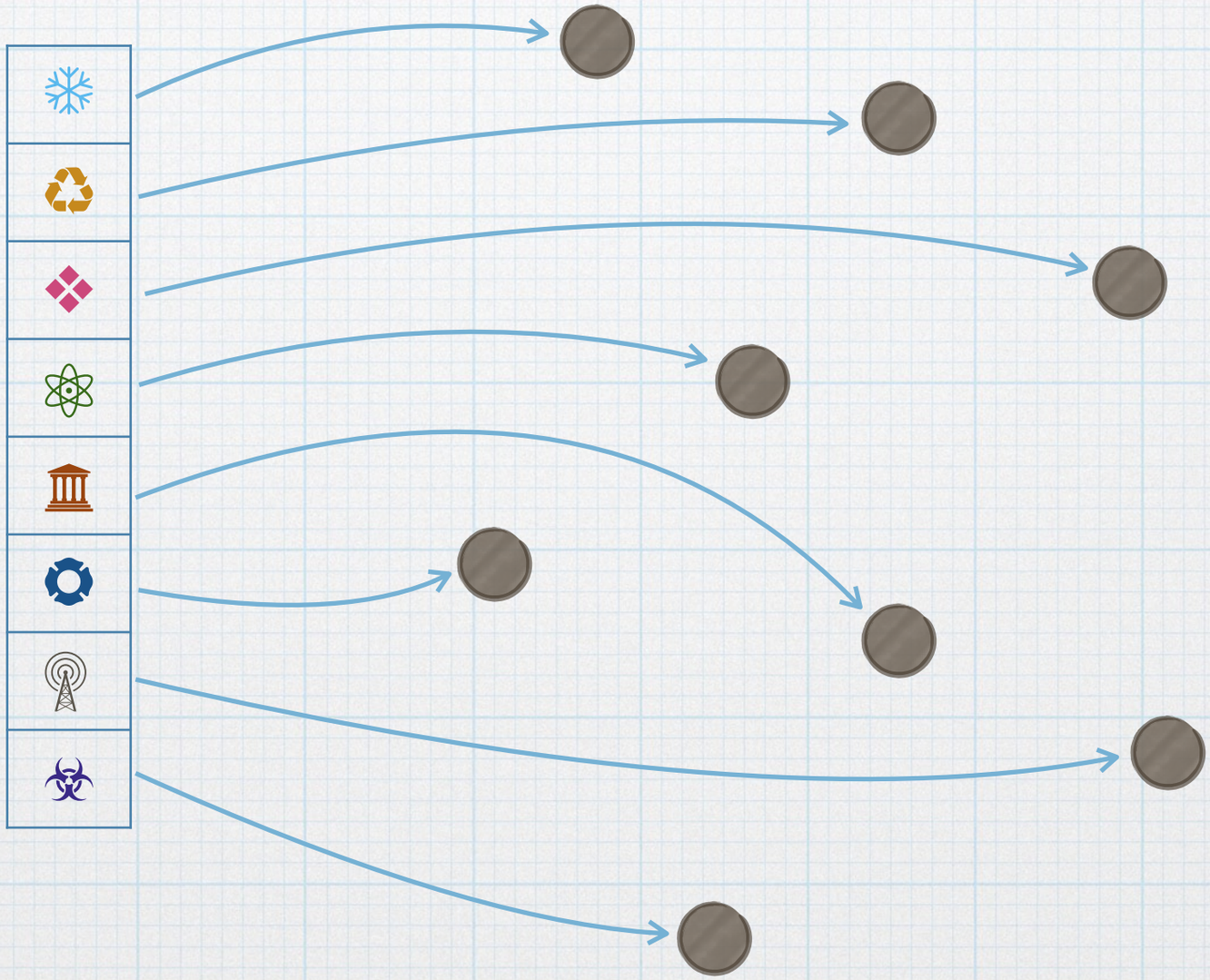
DATA BINDING

One of D3's jobs is to correlate, or to connect or to map, any individual piece of data to its own individual SVG element. This correlation or connection is called ***binding***.



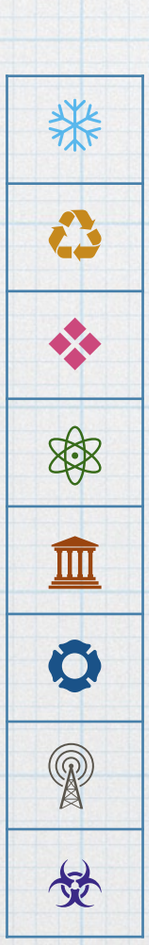
DATA BINDING

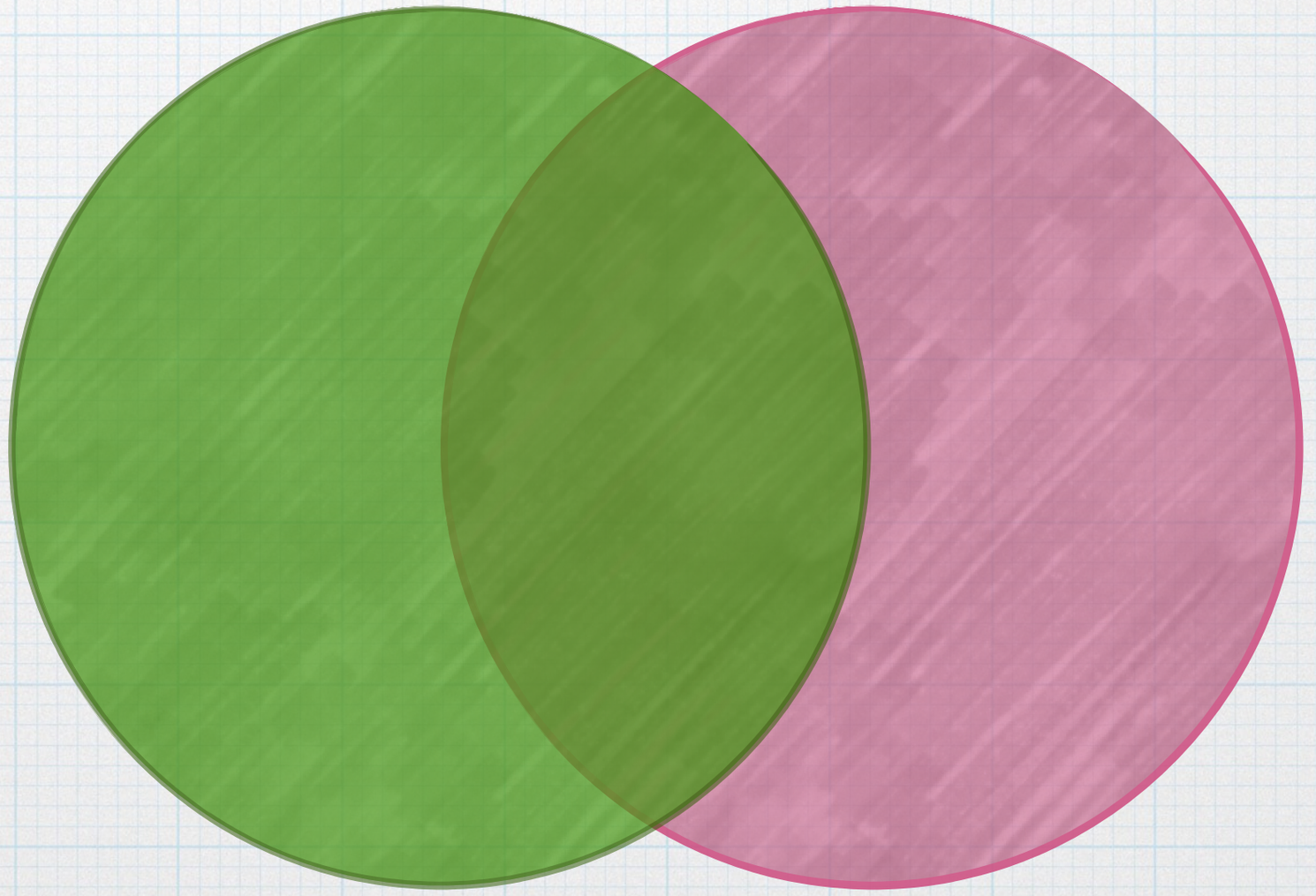
Unlike in the regular DOM tree, a change to our data is NOT automatically updated on our SVG elements. D3 needs the bindings in order to know which element belongs to which piece of data, but it awaits further instruction from us regarding what to change and how.



THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

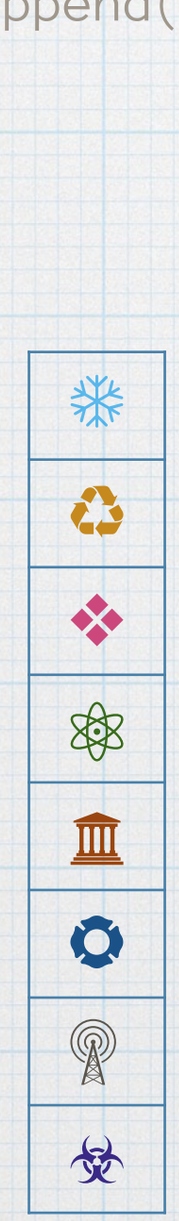


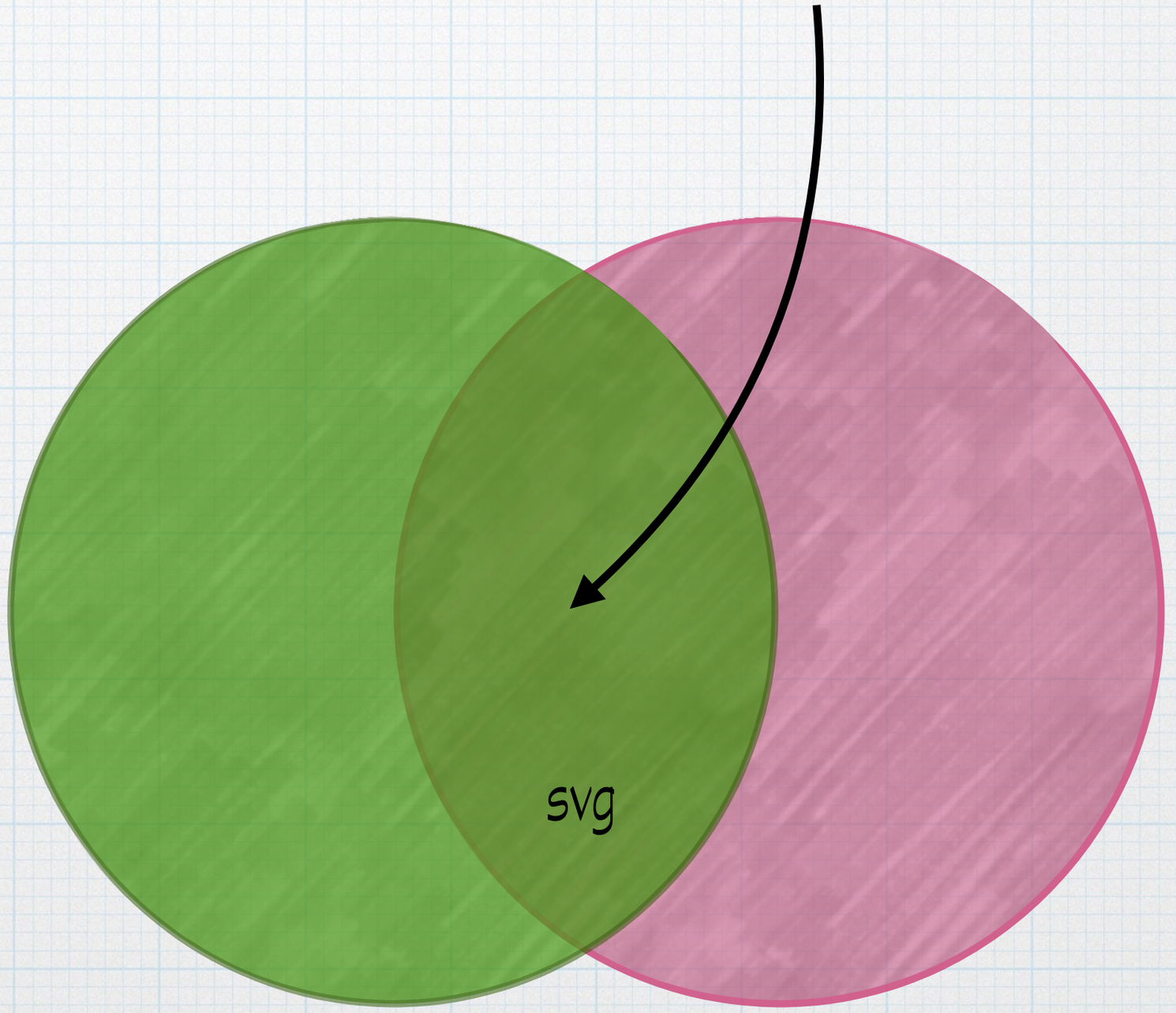


THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

The football-shaped intersection of these two sets is the main container that we've selected. Usually, that's the SVG container, which contains the stuff that's visible on the webpage.

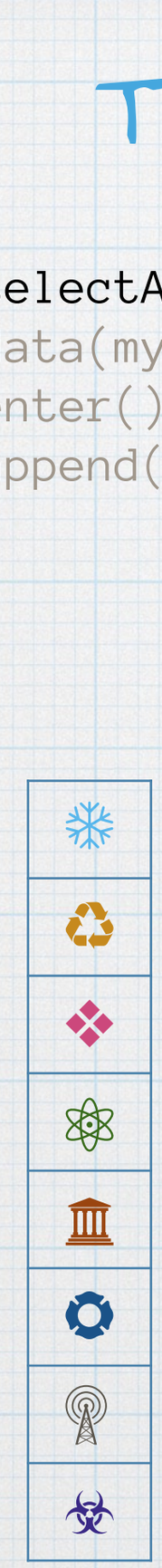


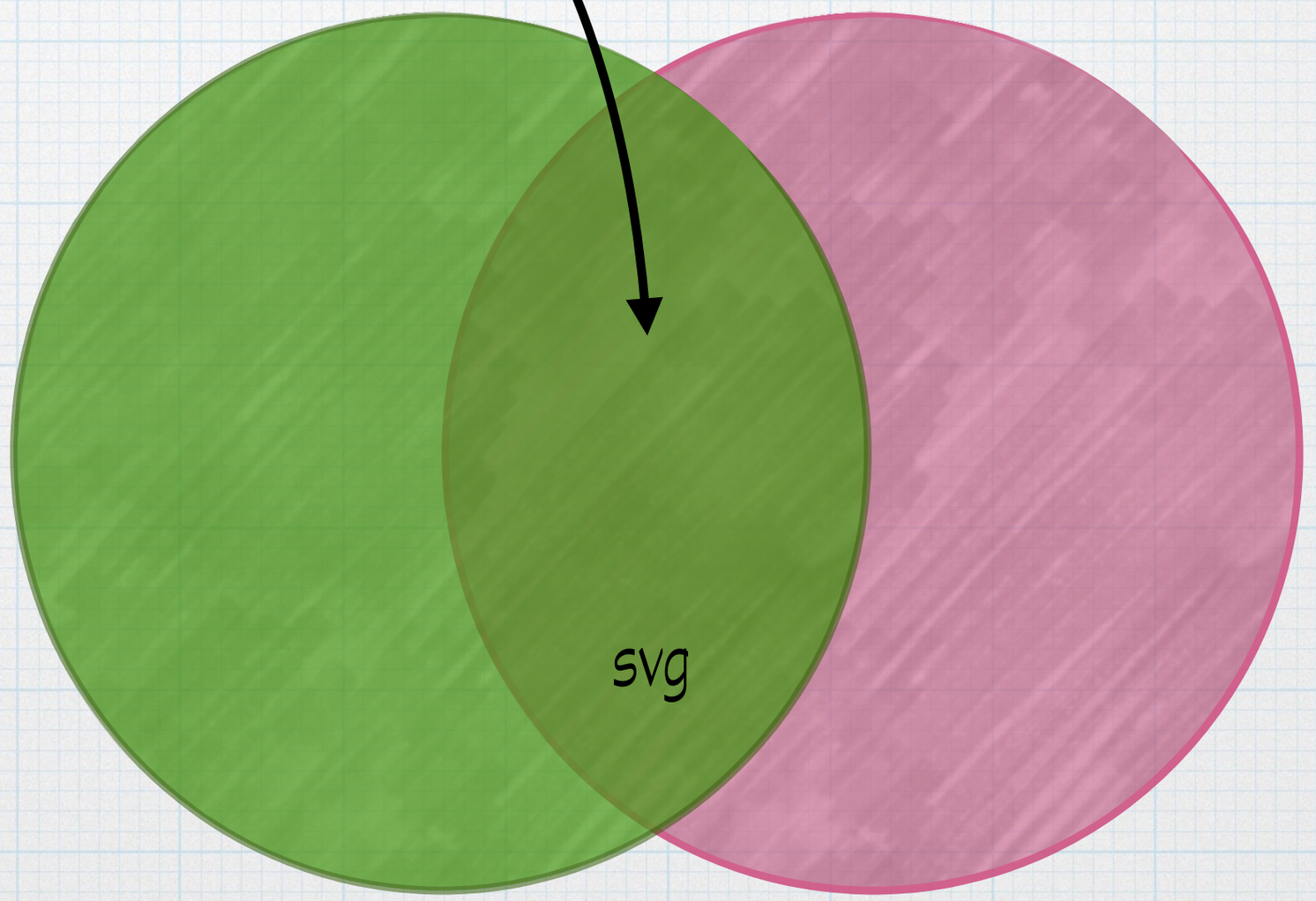


THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

When we **selectAll** the circles, we're selecting all the circles that exist in this football-shaped area. Right now, there are none.

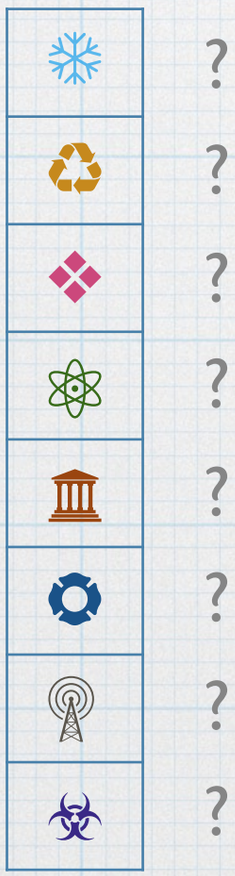




THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

When we **bind** our data set, D3 figures out which data items do not yet have a corresponding SVG element (that's all of them!) . . . .



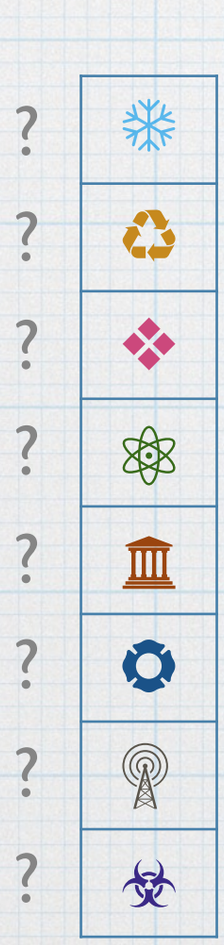
...AND...

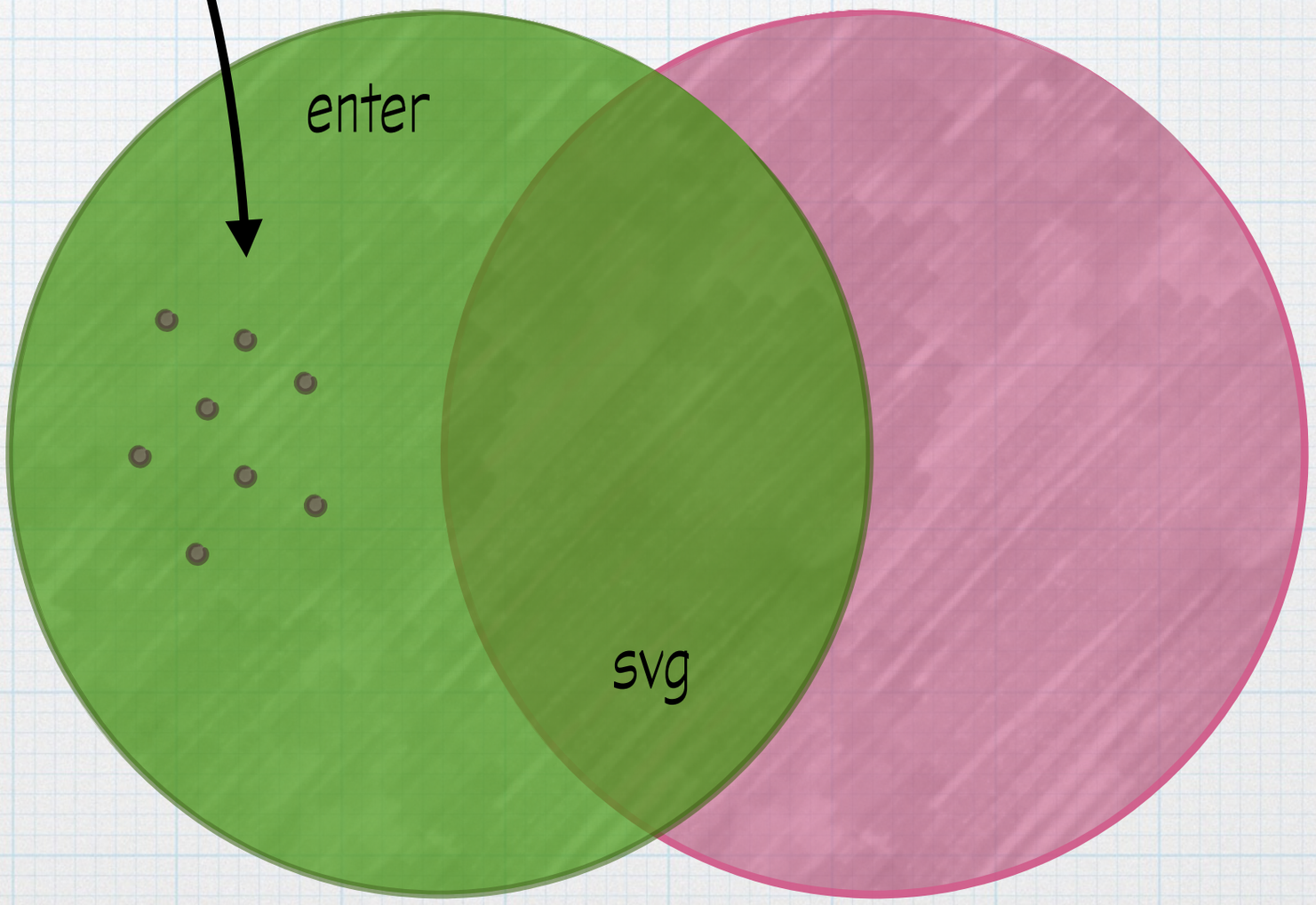
THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

When we **bind** our data set, D3 figures out which data items do not yet have a corresponding SVG element (that's all of them!) . . . . *AND:*

Unbound items are put into the **enter()** space.

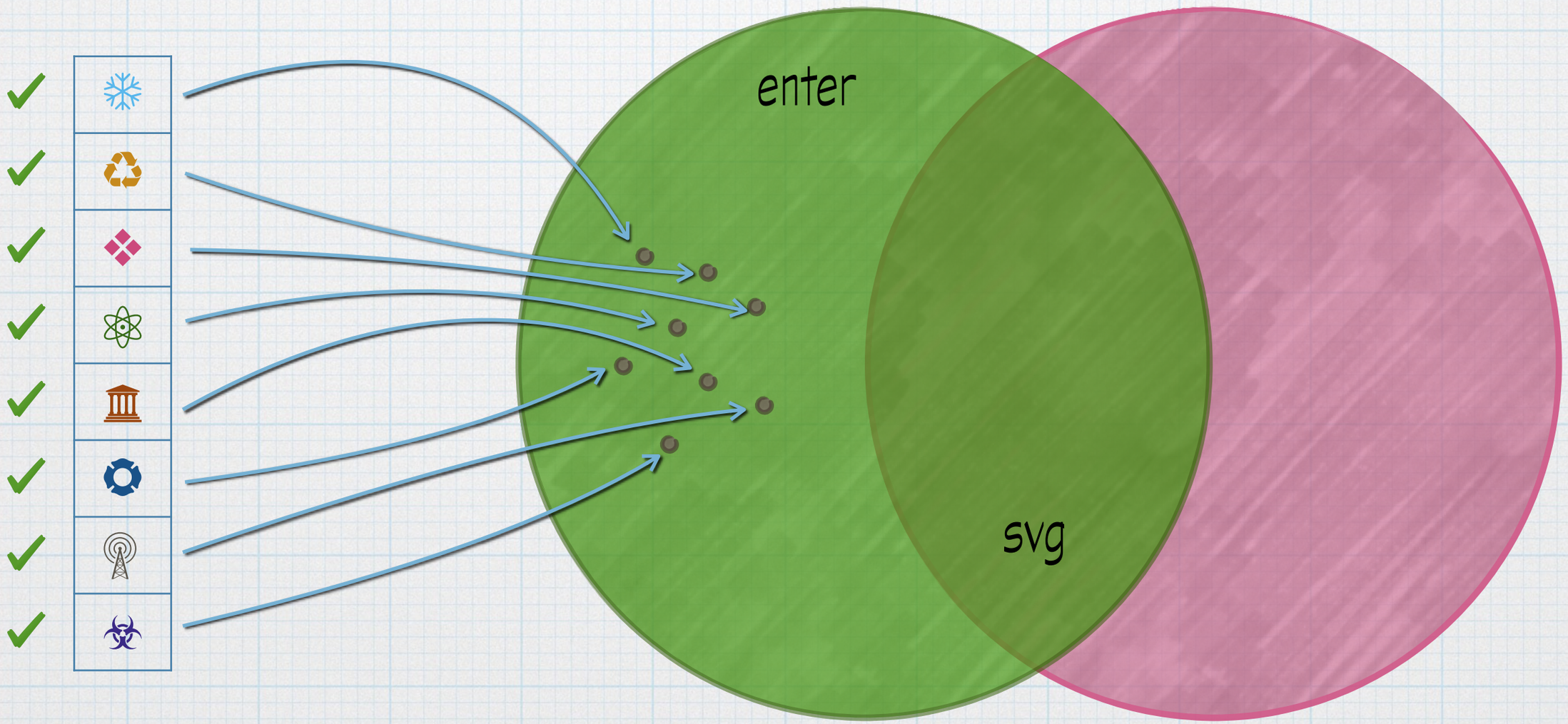




THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

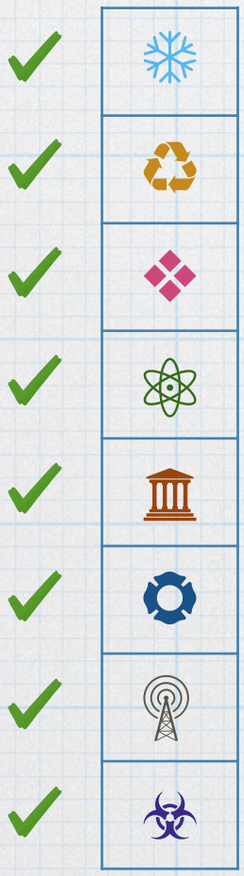
Now the dataset has been bound. Each data item is accounted for. None of the bound elements is visible because it has not yet been assigned to a particular SVG representation (circle, rect, line, etc.)



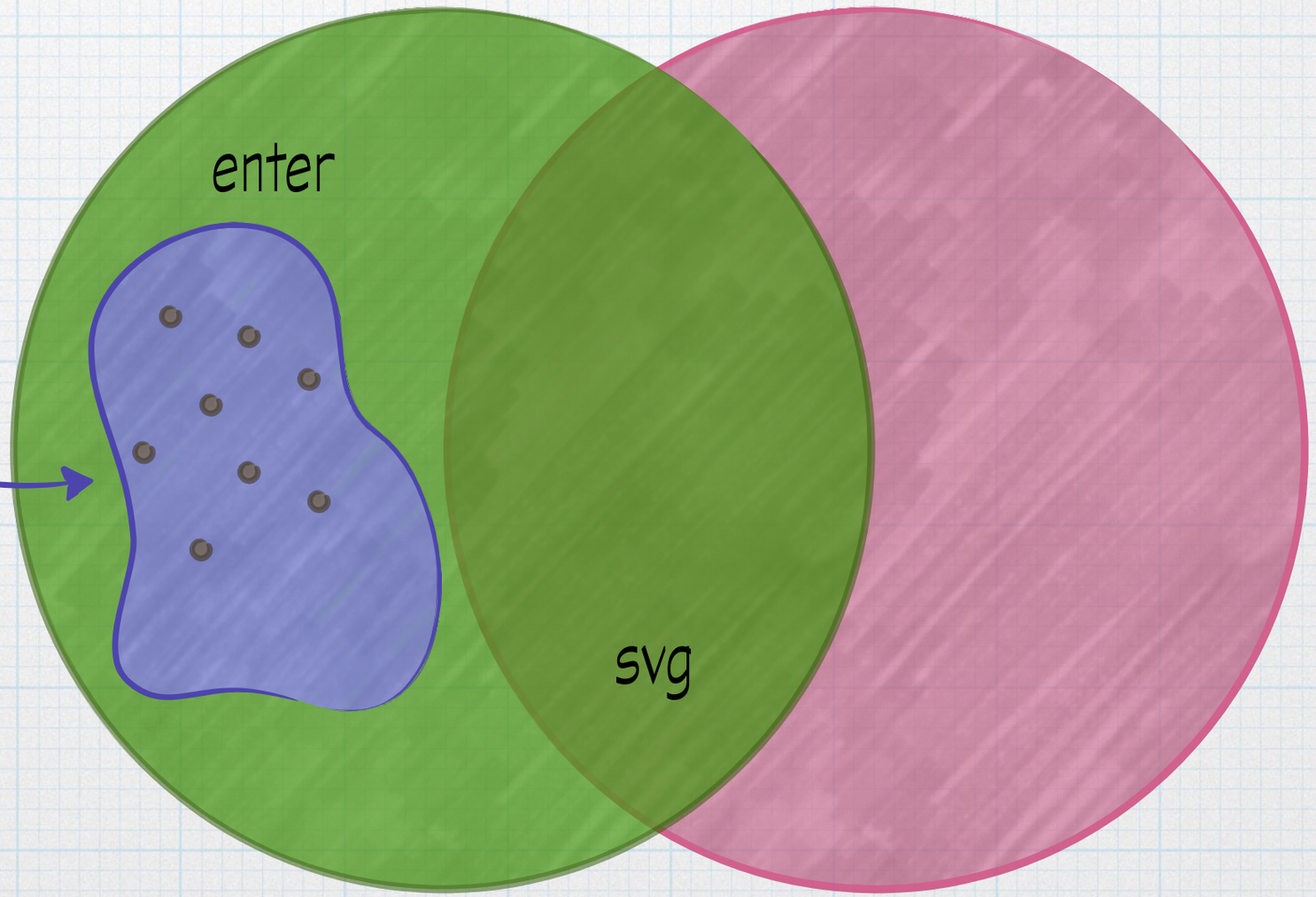
THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

Next, I tell D3 that all the code that follows in the chain should be applied to data items in the **enter()** set.



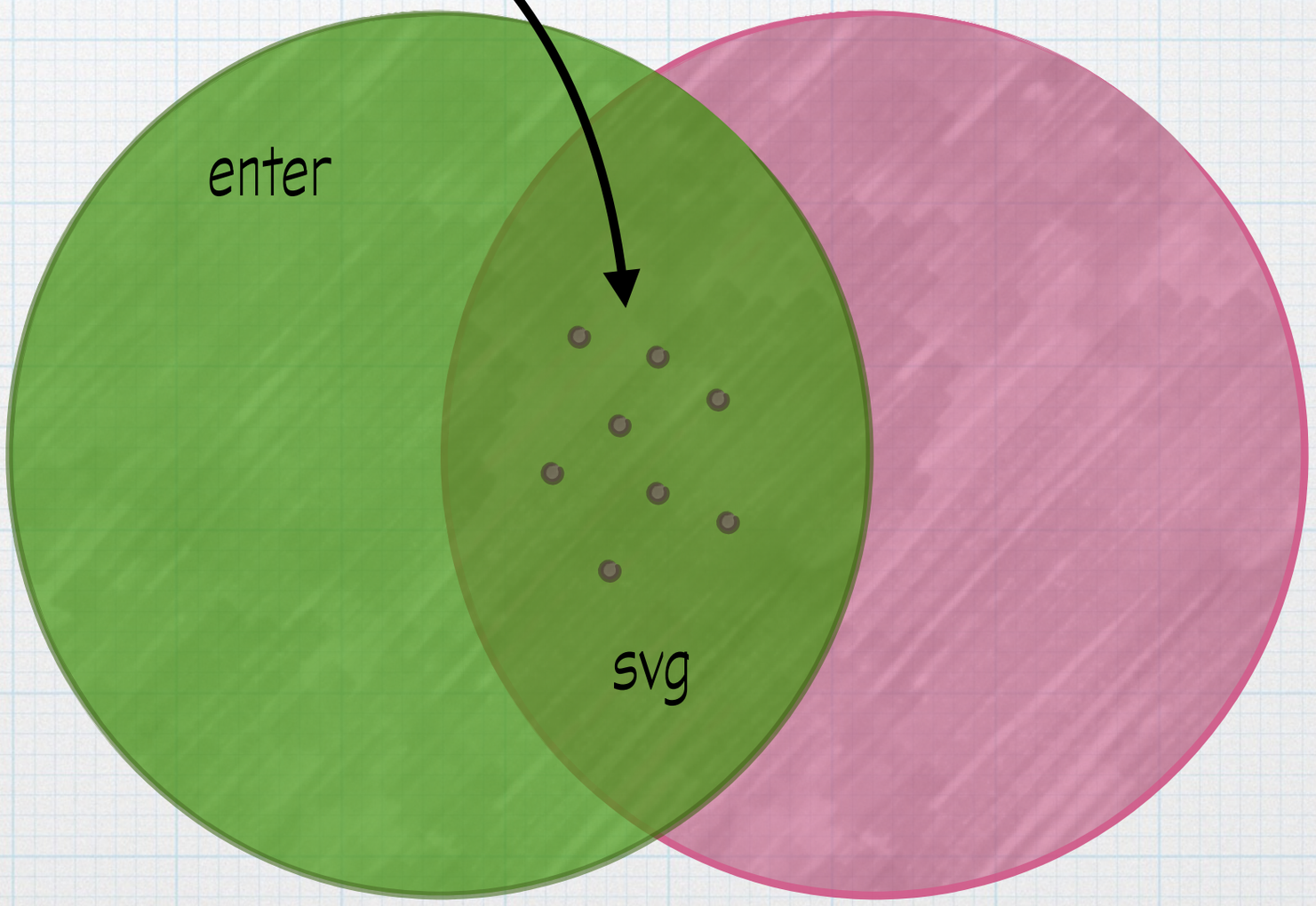
Use these, please!



THE DATA ENTER WORKFLOW

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

For each item in **enter()**, the **append()** command attaches a new circle element to the SVG container. By setting **.attr()**, which are not shown here, D3 gives the circles concrete appearance. They are now visible.



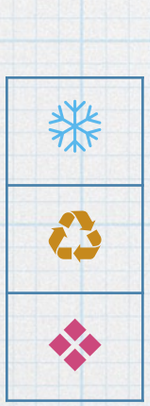
Scenario Two

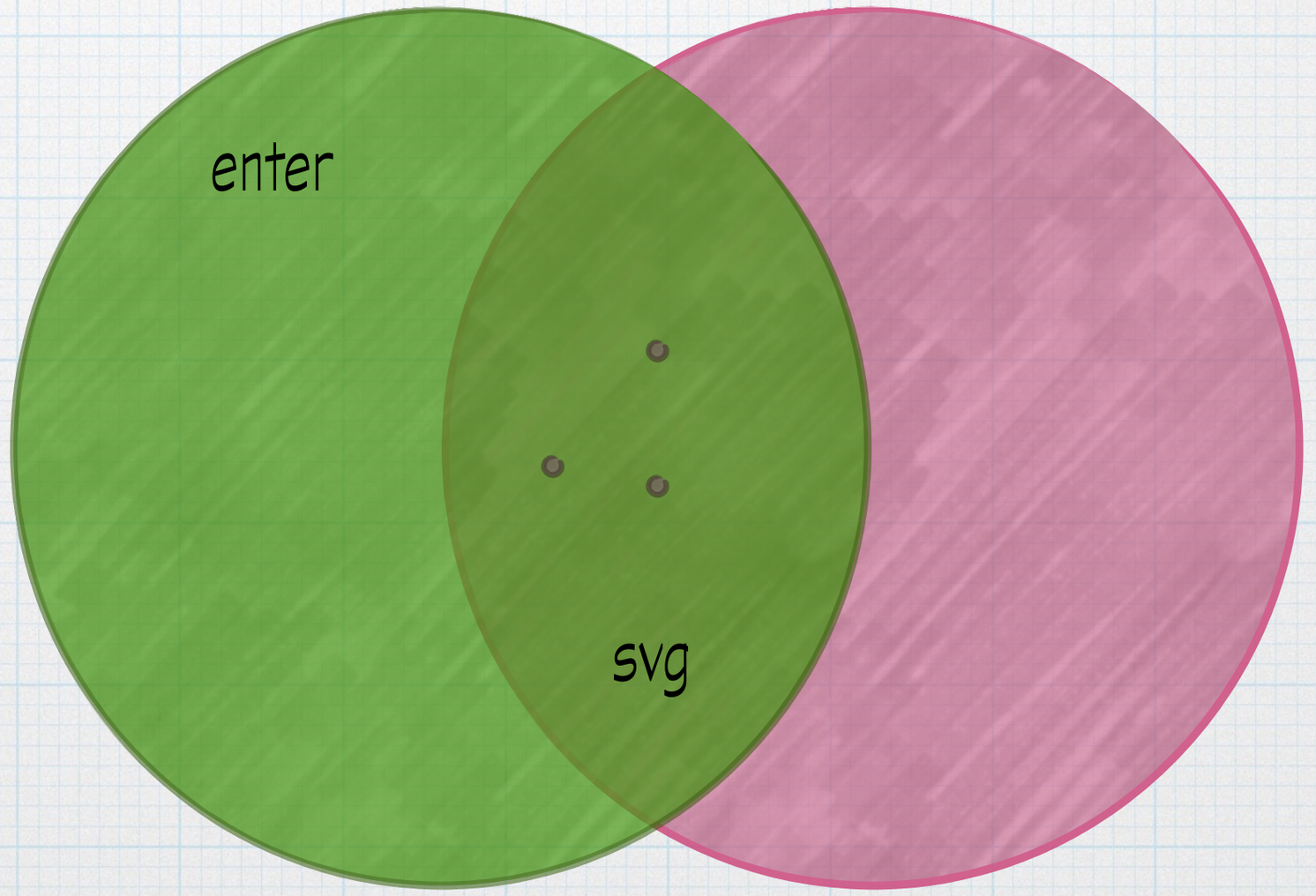
We Now Join Our Regularly Scheduled Dataviz Already In Progress. . .

START AGAIN

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

New scenario! Let's say that our web visitor has already done some interactive work, and they're currently visualizing three data points. But they just clicked a button to add five more. What happens?

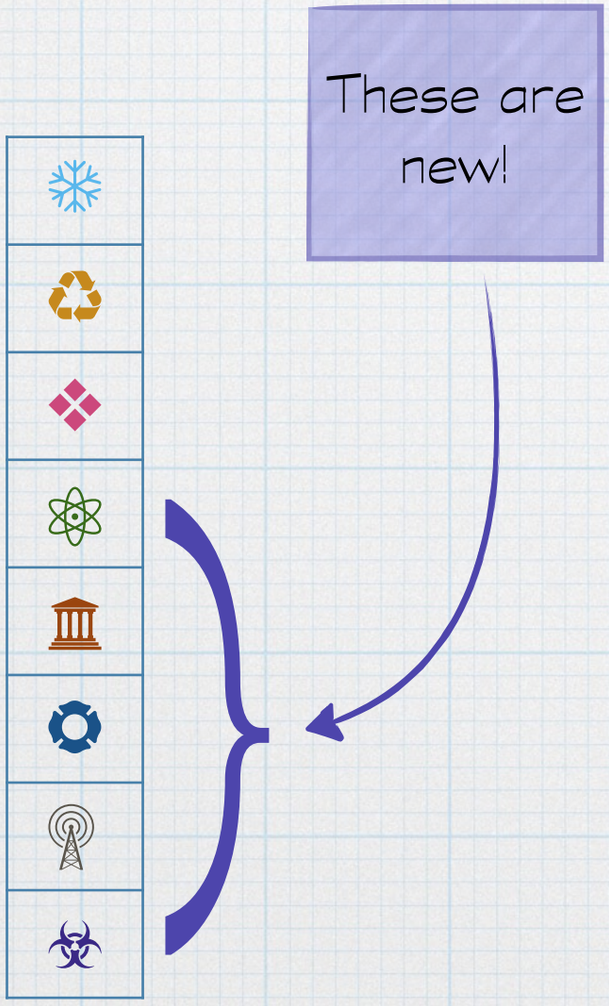


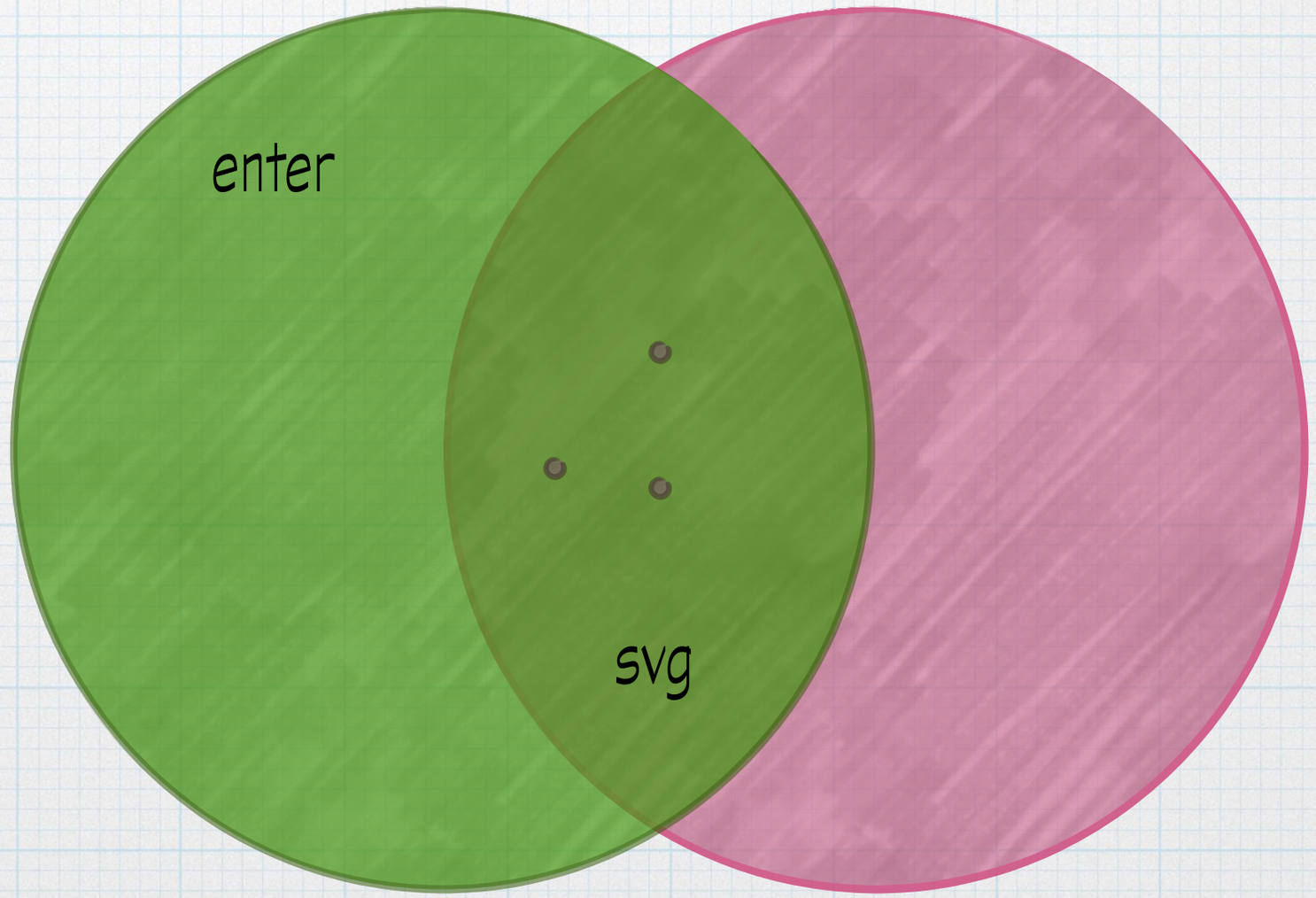


ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

First, the data set changes to add the new elements. It might be a CSV or a JSON load. This doesn't yet affect D3 at all.

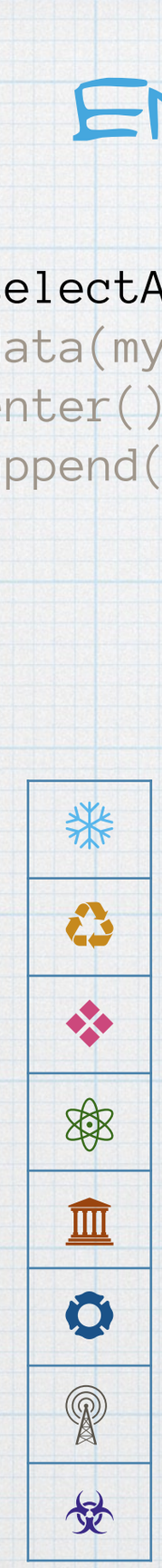


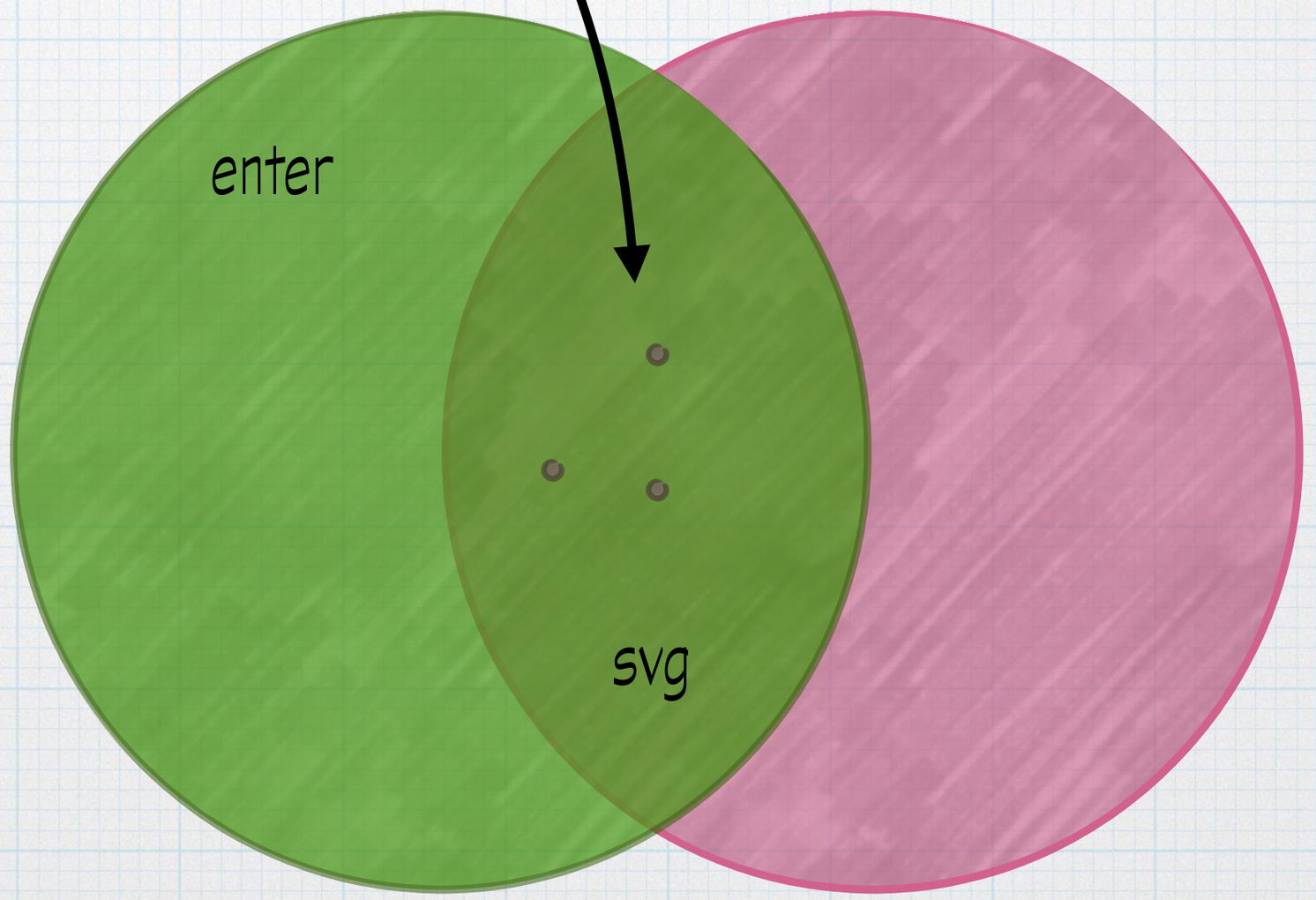


ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

Next, we **selectAll** the currently existing circles because, in order to proceed, D3 needs to figure out where we currently stand.

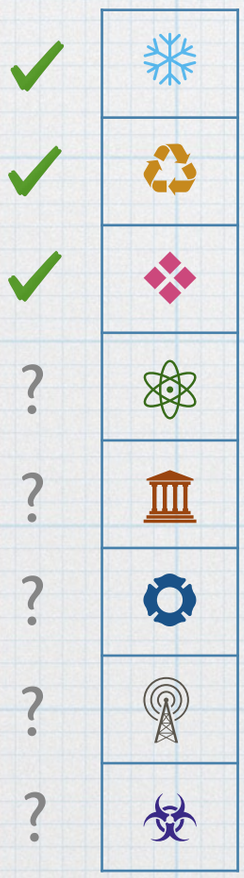


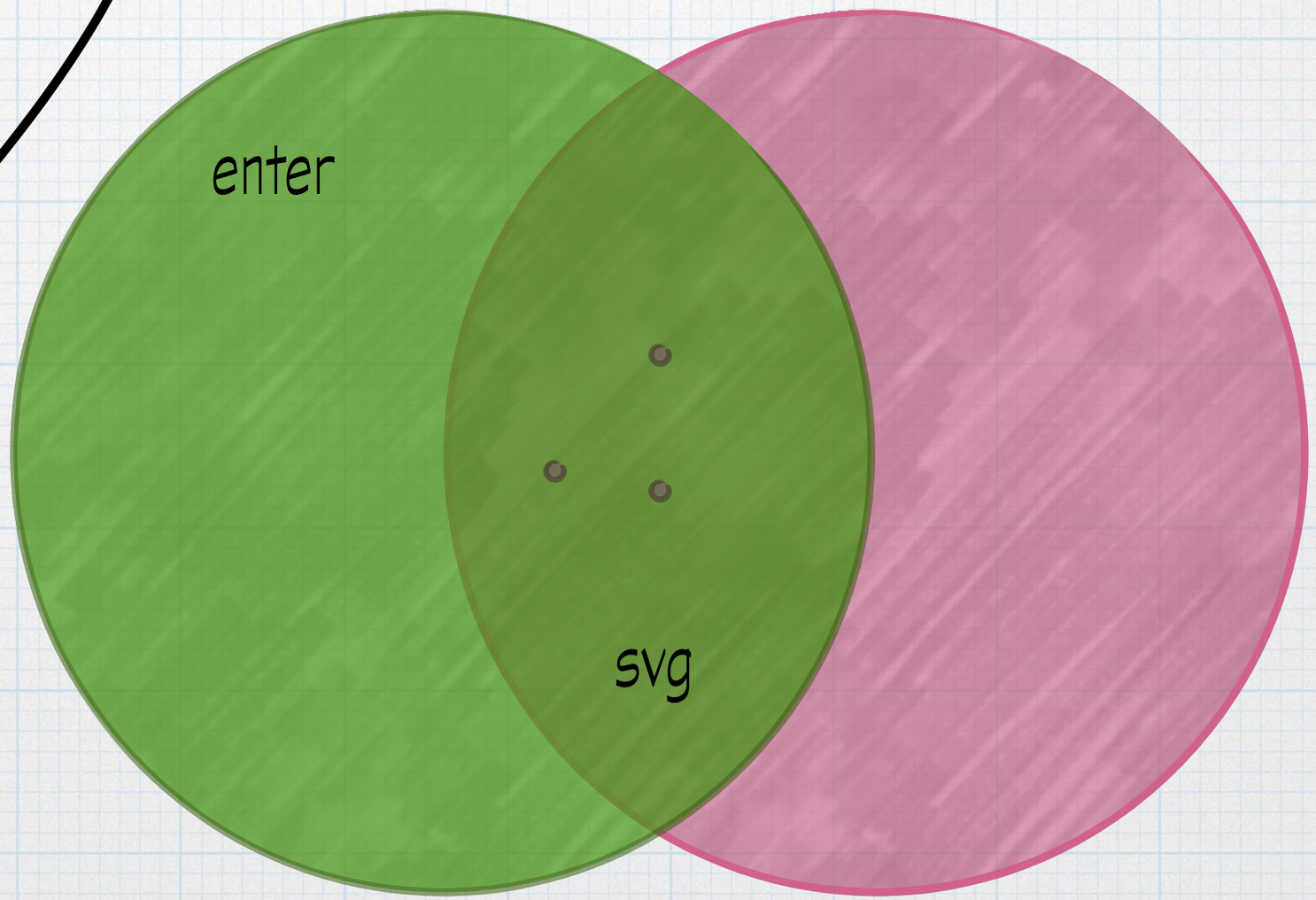


ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

When we **bind** our updated data set, D3 checks to see which data items already have bindings and which do not. And . . .



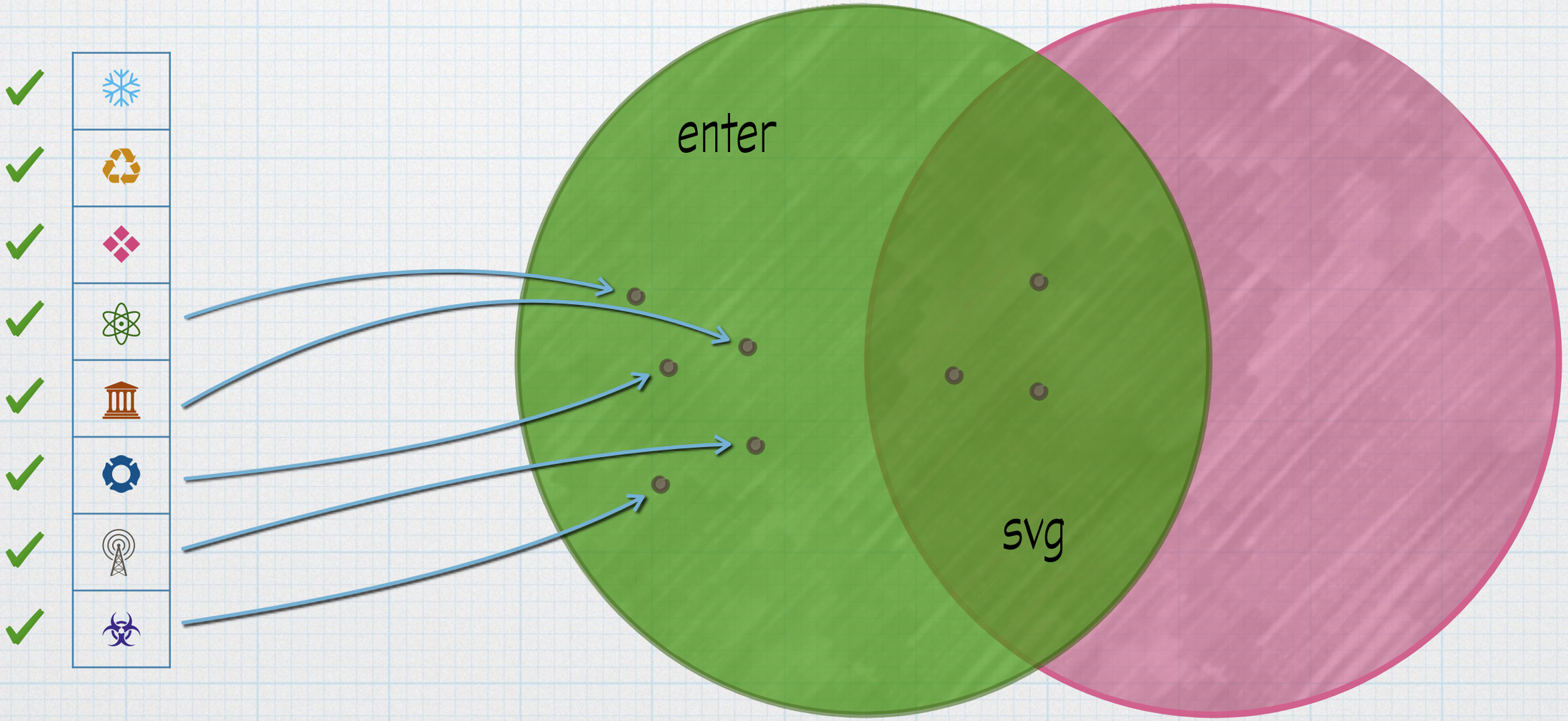


ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

When we **bind** our updated data set, D3 checks to see which data items already have bindings and which do not. And . . .

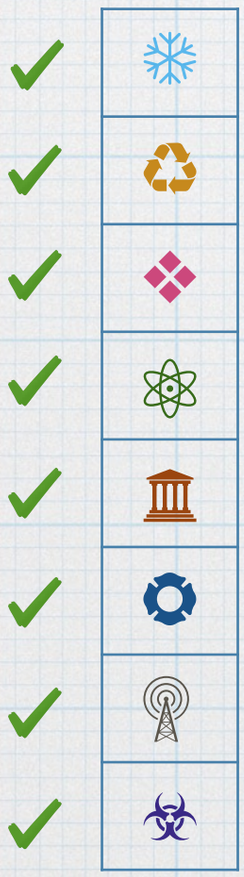
Binds the unattached data items.



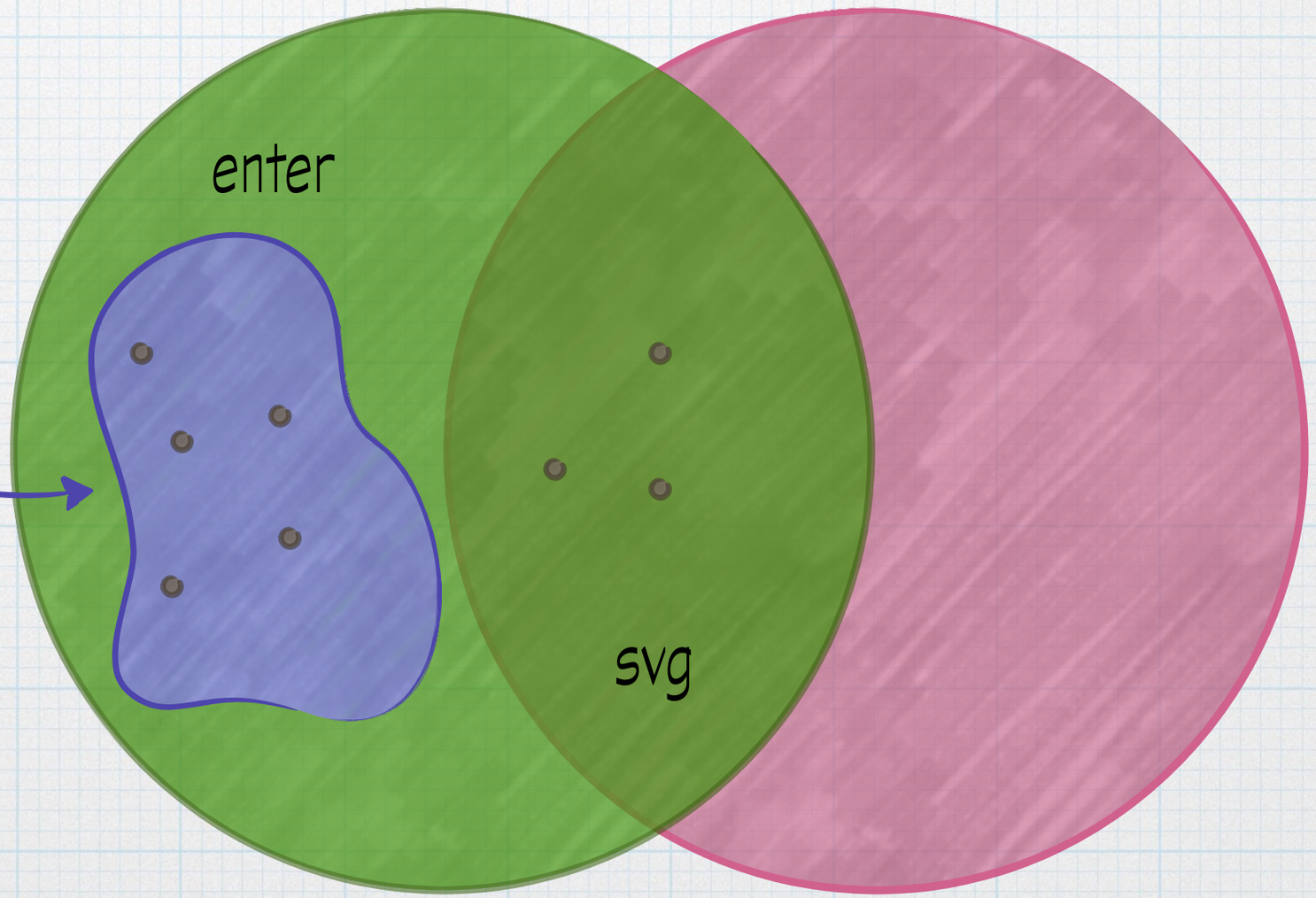
ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

The **enter()** command tells D3 that subsequent code instructions in the Javascript chain should be applied to the set of items currently in **enter()**.



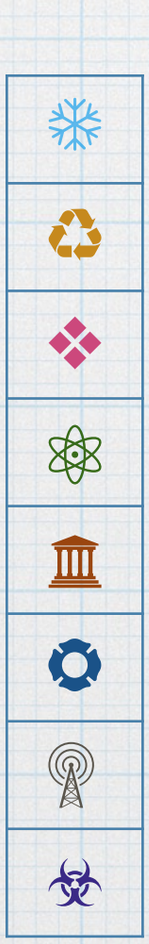
Use these, please!

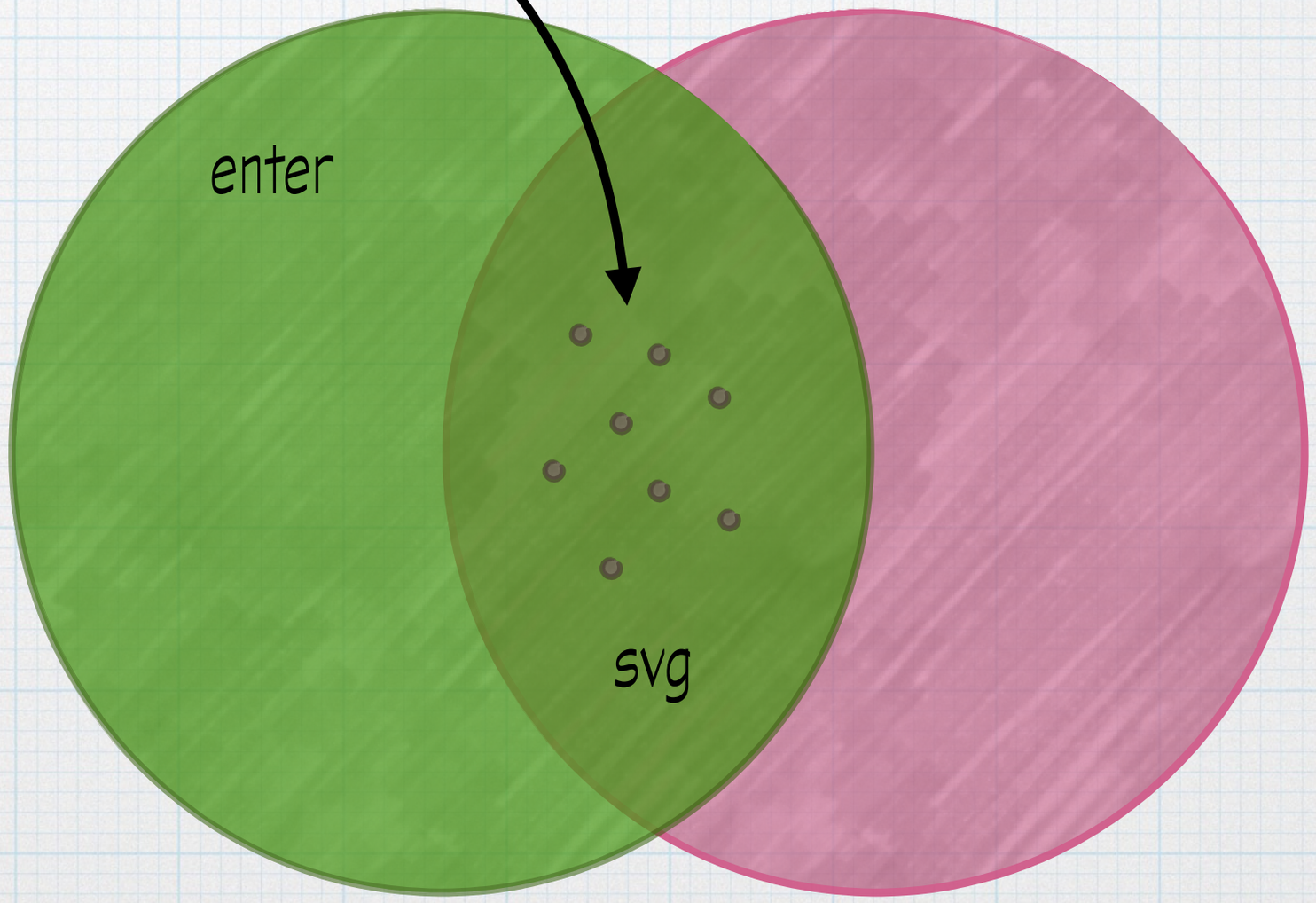


ENTER WORKFLOW TAKE TWO

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

And **append()** creates specific SVG shapes with specific CSS attributes and attaches them to the SVG sub-tree in the DOM.





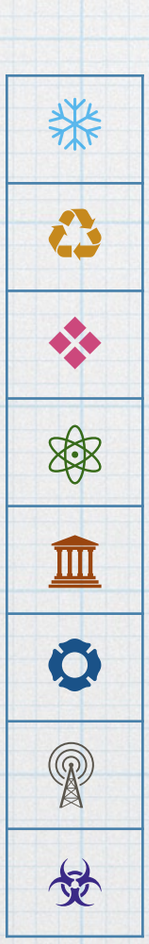
Scenario Three

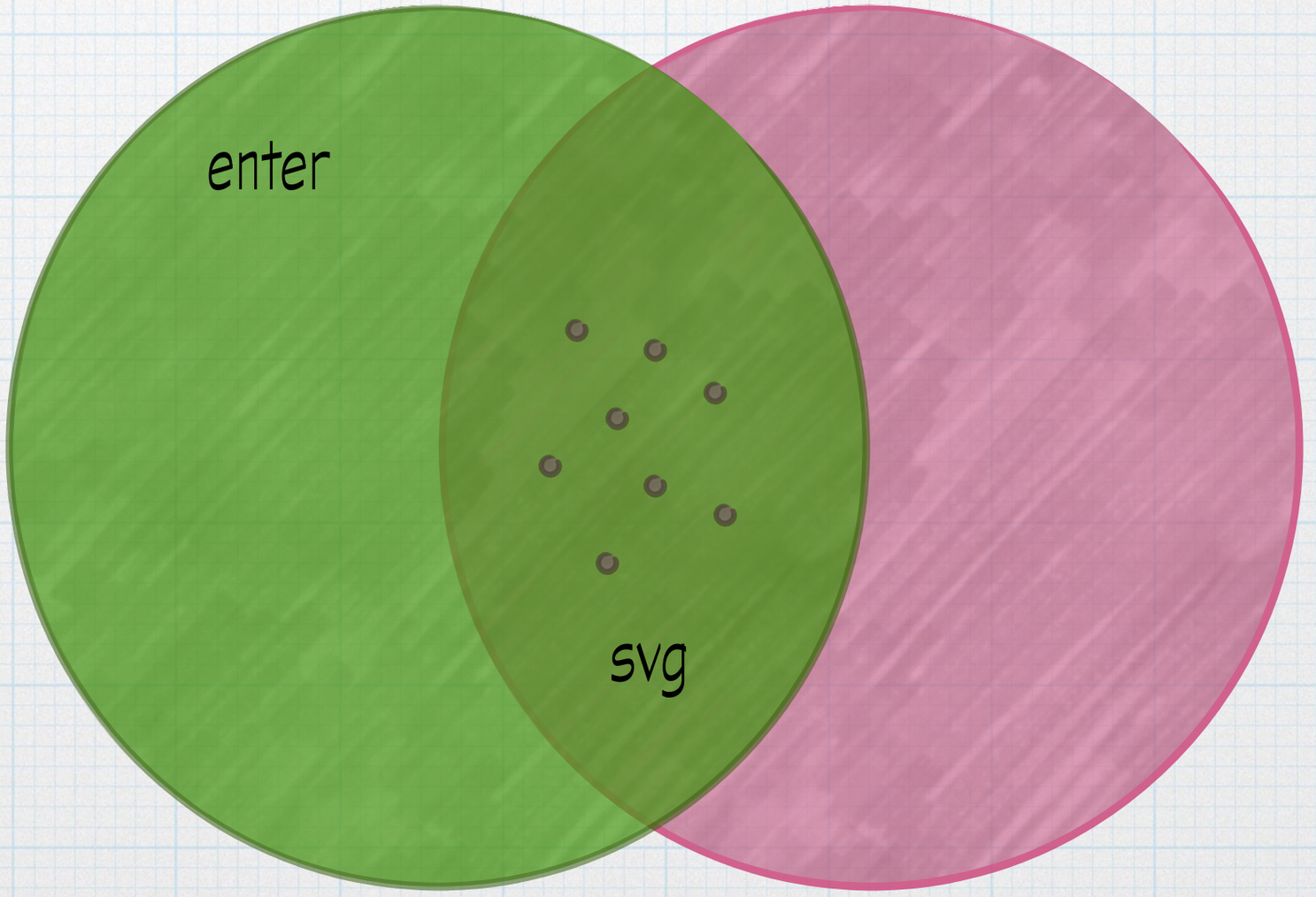
Exit

EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

New scenario! Our user has been exploring data interactively and now filters the data set in such a way that we need to remove some items.

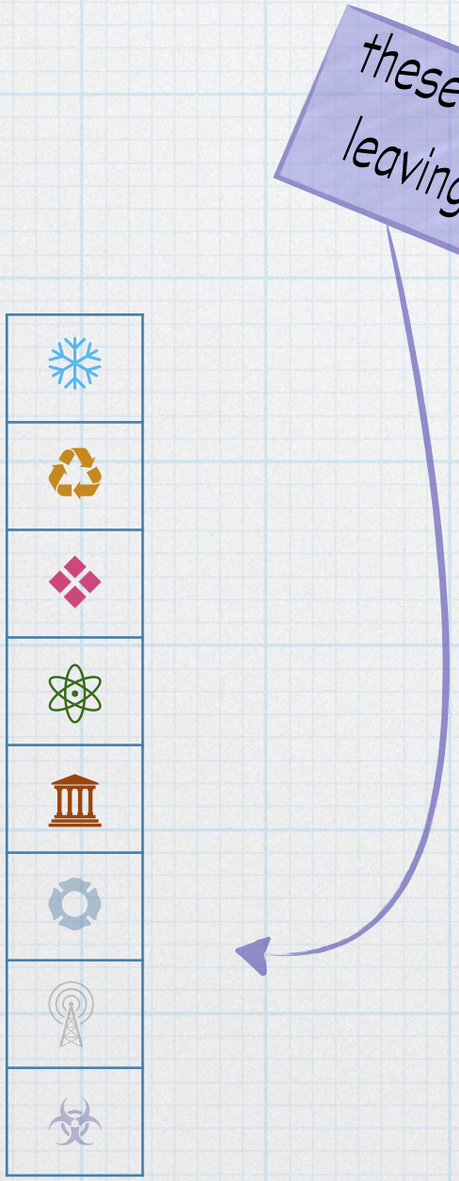


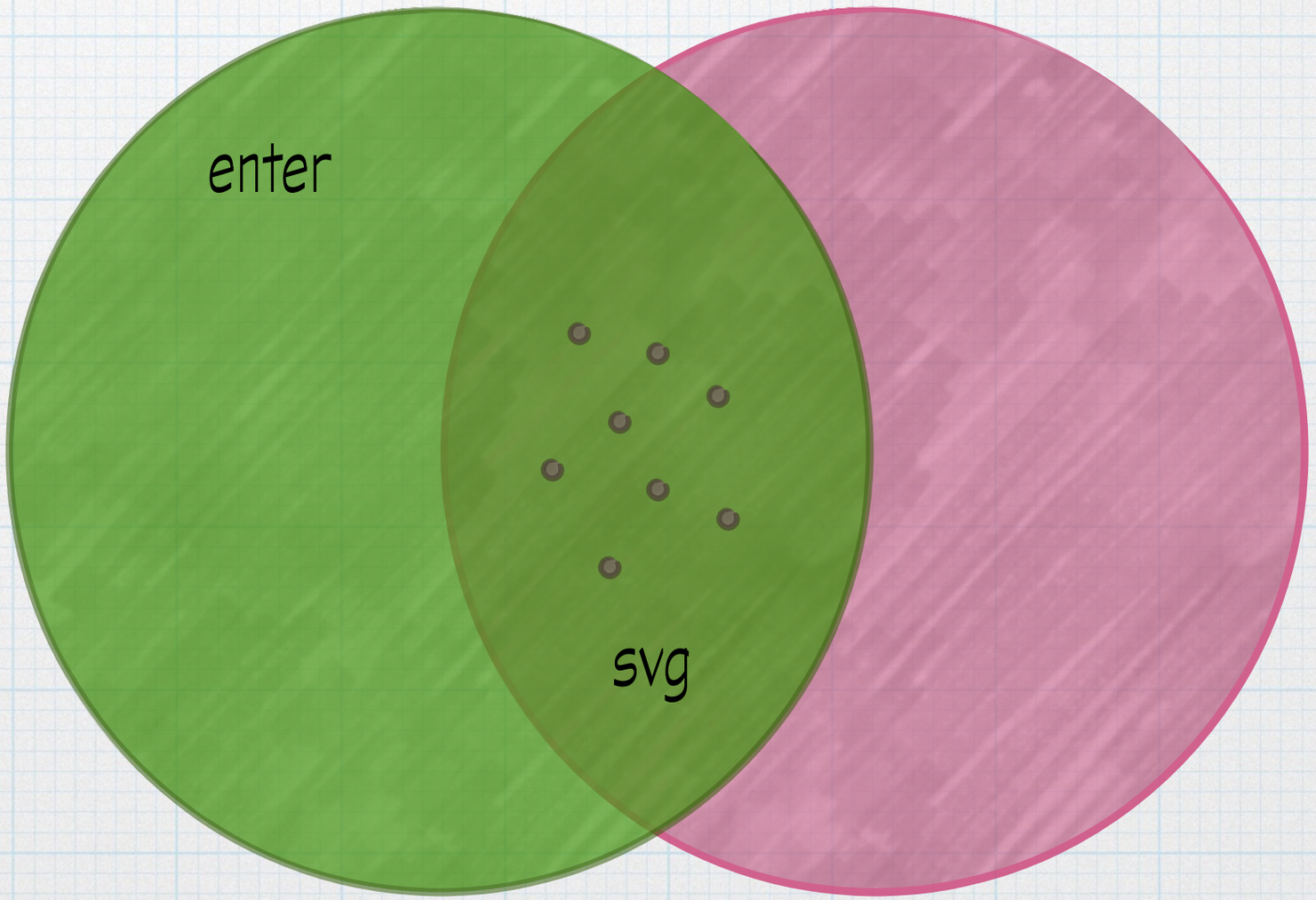


EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

Three items in the dataset have been deleted, and now we need to update the dataviz with an **exit()** workflow.

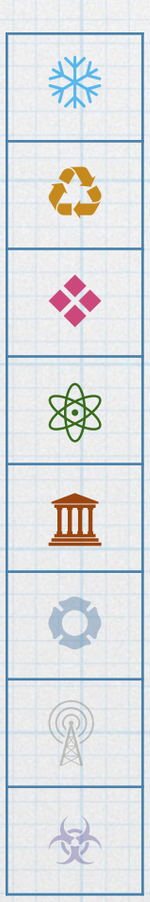


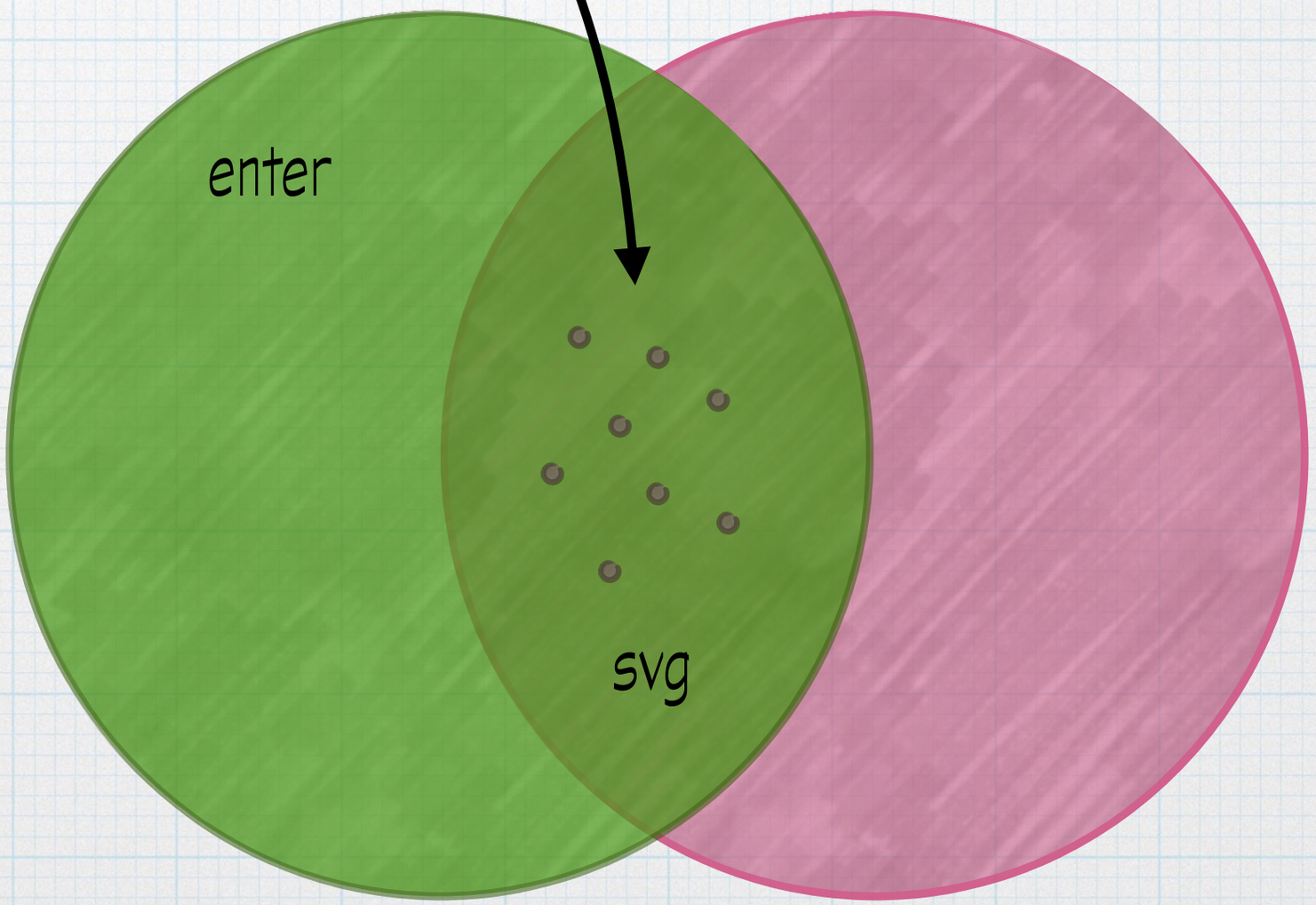


EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

Again, I start by doing a **selectAll** on all the circles that I currently have.

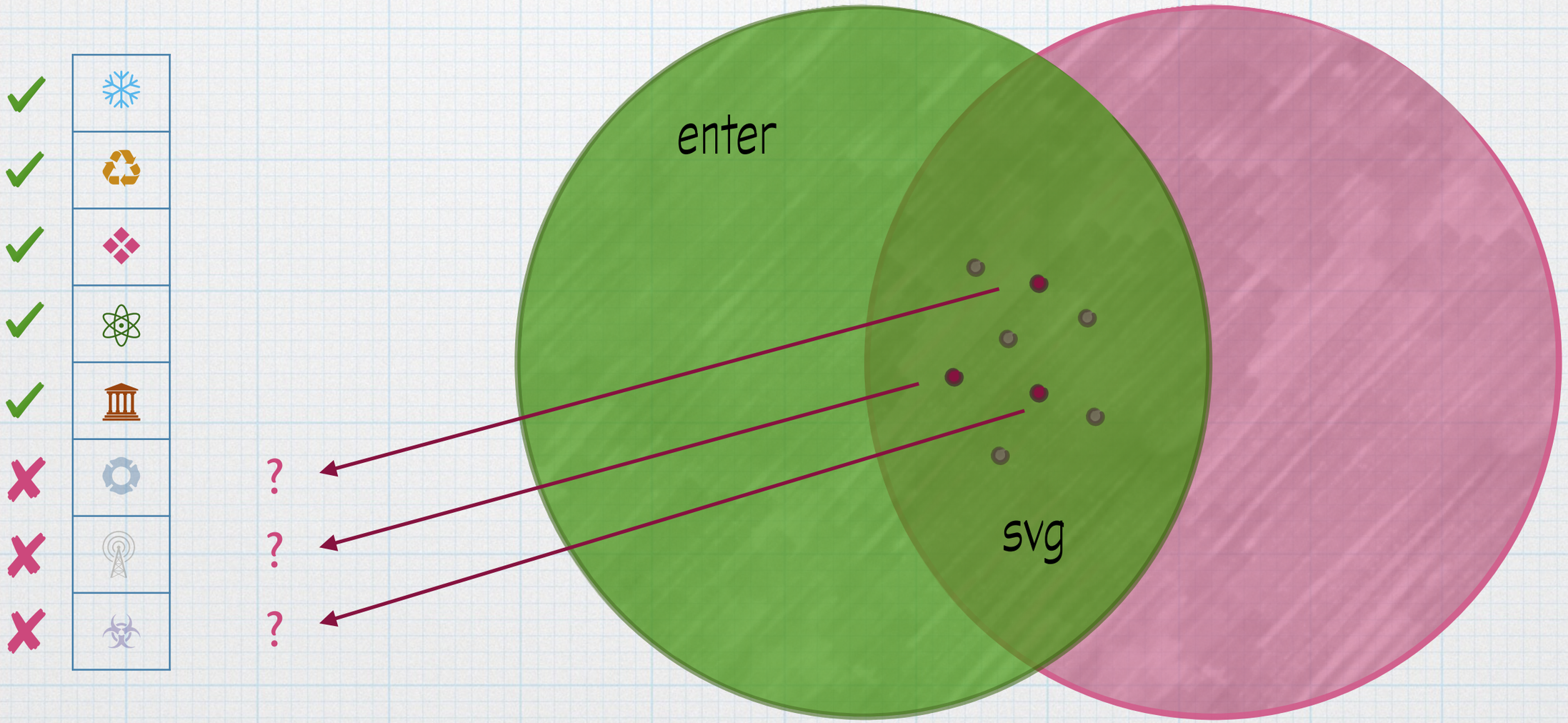




EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

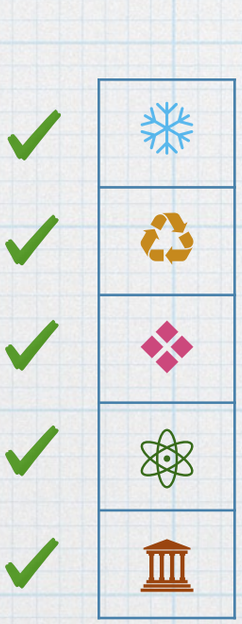
But now when I bind my data, things are different. D3 recognizes that five elements have bindings, but D3 also learns that three elements no longer have data bindings.

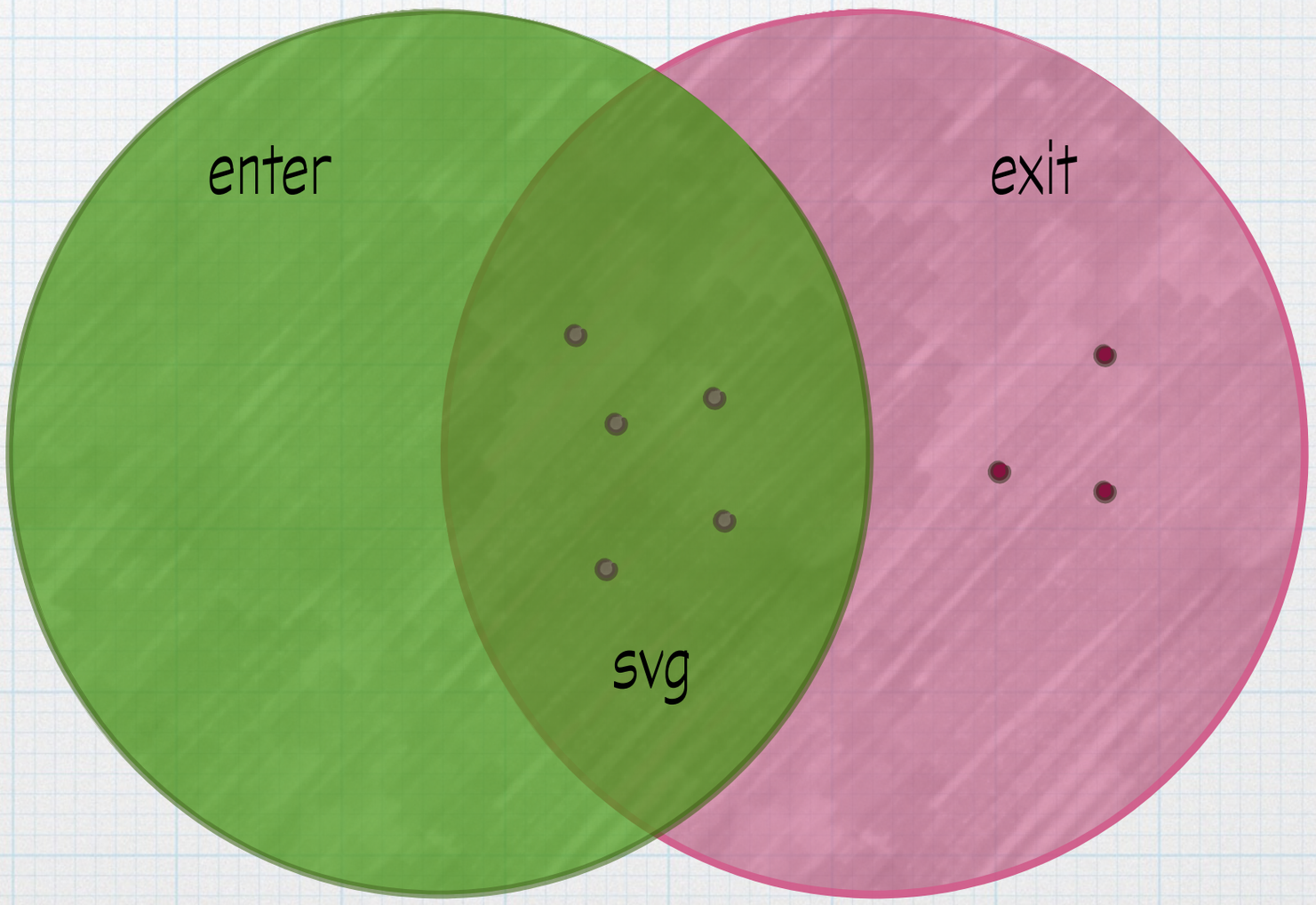


EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

So D3 puts these into the **exit()** set.



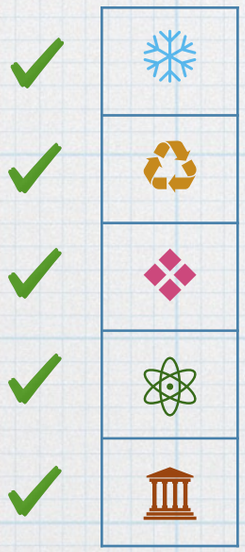


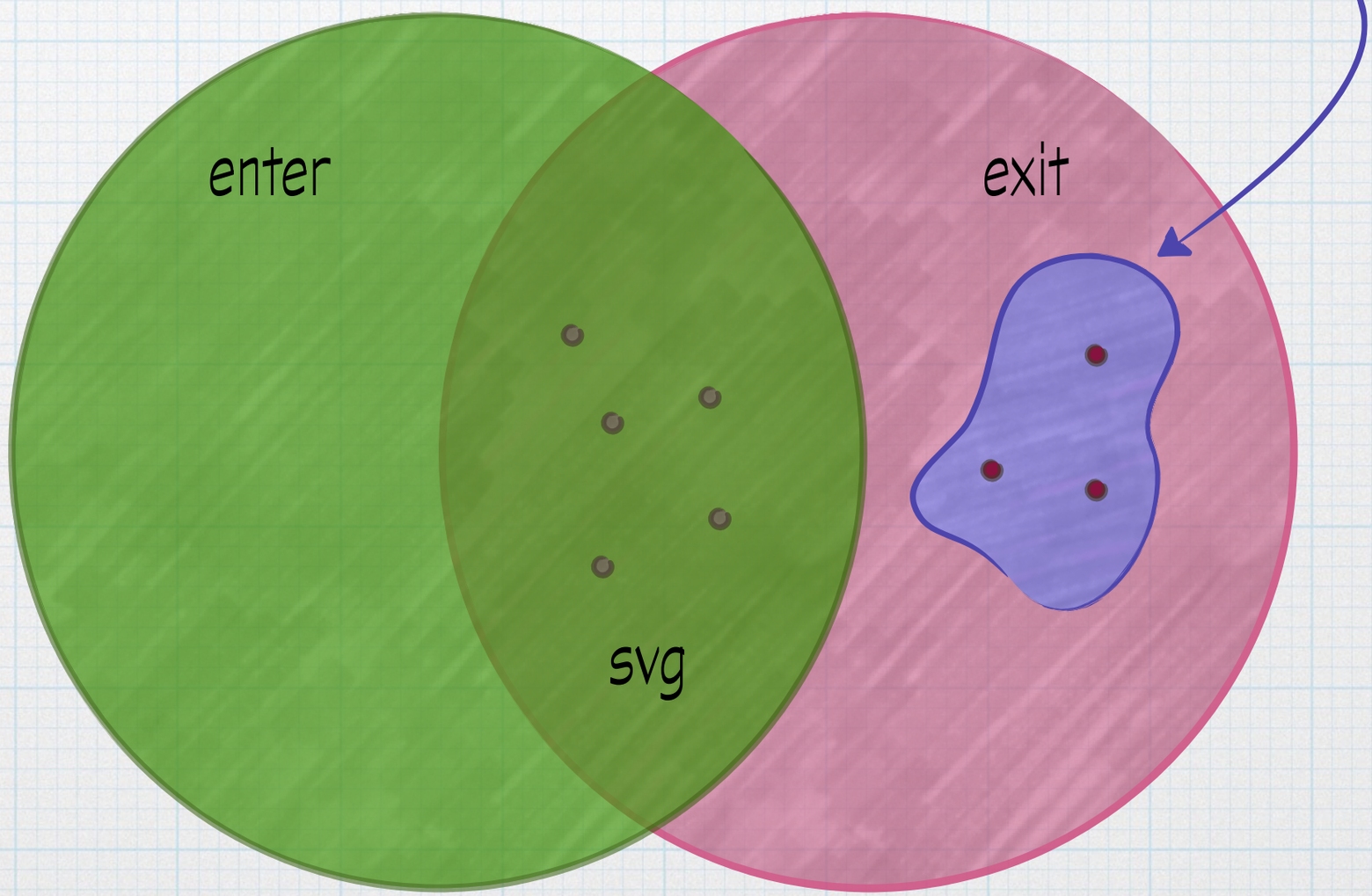
EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

And now I tell D3 that all further commands in the chain should be applied to the set of items currently residing in **exit()**.

Use these, please!

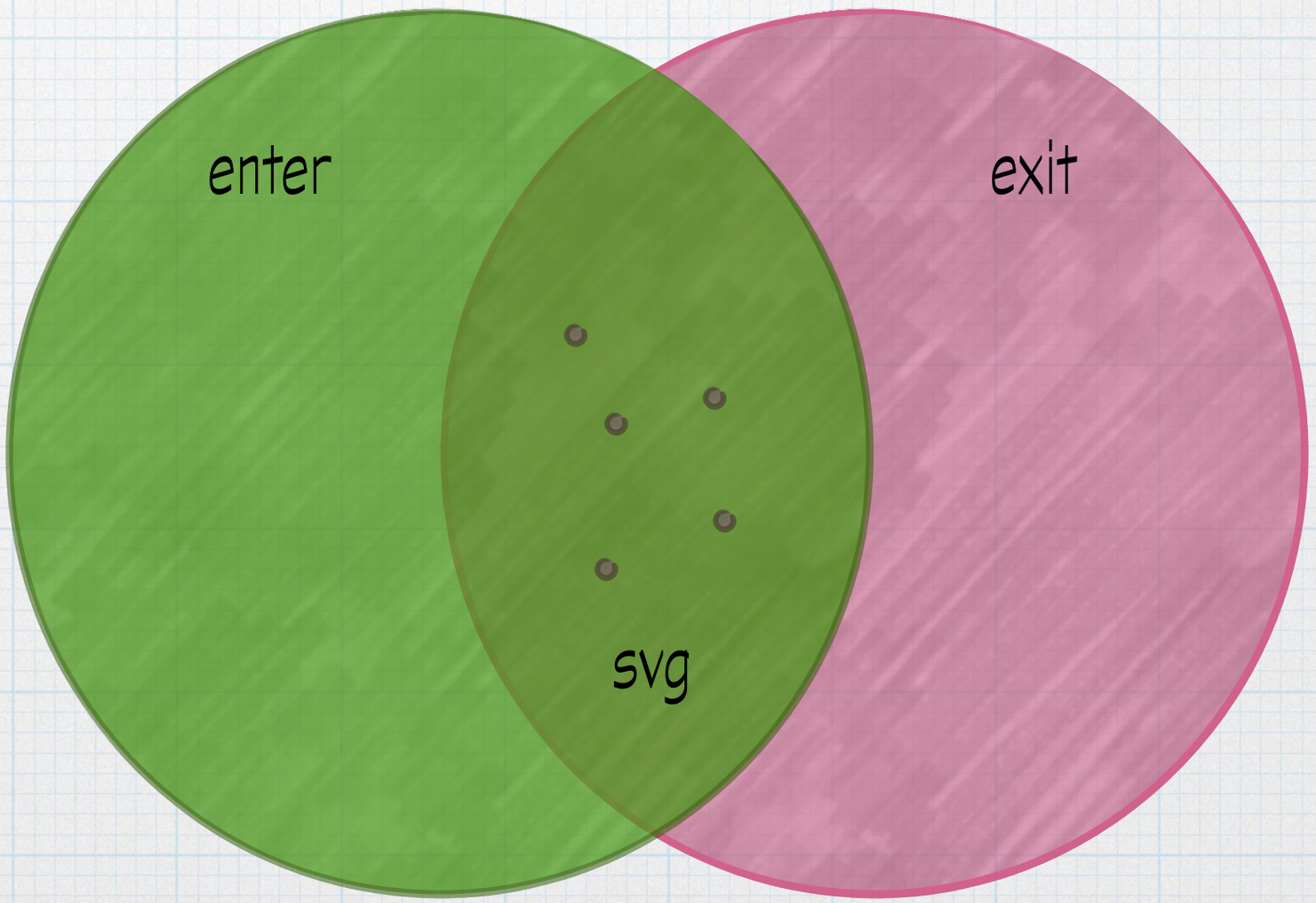




EXIT WORKFLOW

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```

And I have D3 **remove** them. The SVG elements have been deleted and no longer exist.



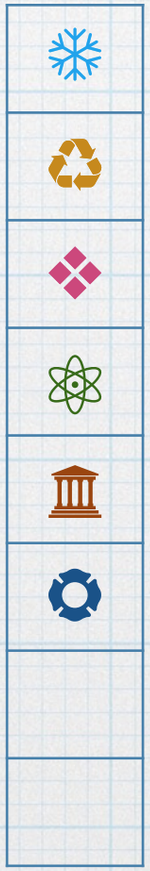
Scenario four

In reality, D3 just calculates all three sets at the same time

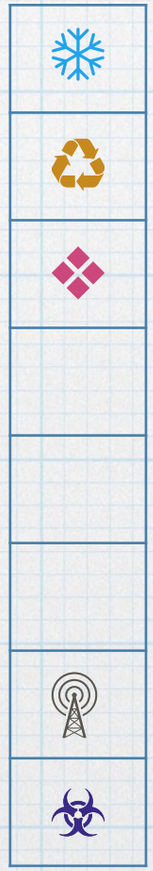
THE UPDATE

The first thing we need D3 to do is called an "update." D3 needs to compare the old dataset, **.selectAll()**, to the new dataset, **.data()**, so that it can calculate all the relevant differences between the two. These differences will be sorted into three distinct groups: stuff that stays the same (selected), stuff that's being added (enter) , and stuff that's being deleted (exit).

`.data()`



`.selectAll()`

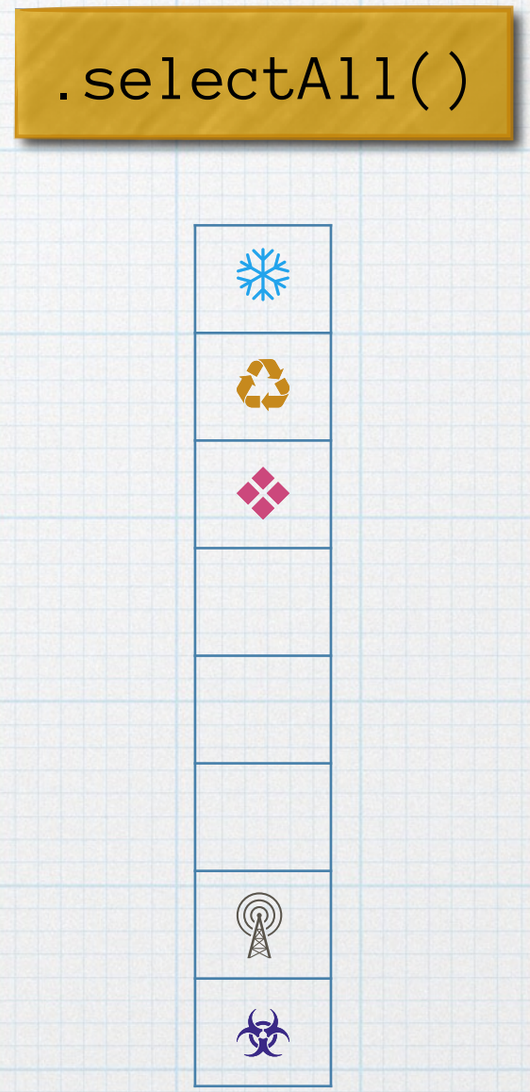


SELECTING OLD DATA

D3 makes no assumptions about the old dataset. After all, the dataviz viewing window might have been created by merging many different datasets that are each represented by many different kinds of SVG elements. D3 can't read our minds to know which of these elements we want to work on. Maybe we want to modify all the circles. Or add new squares. Or delete all the lines.

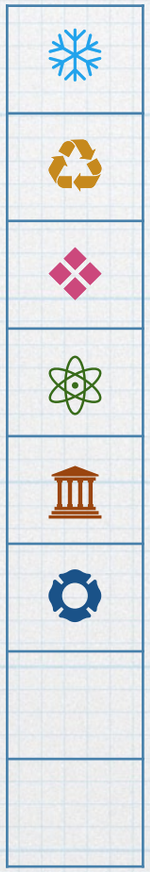
So D3 doesn't guess. It gives us complete control over selecting precisely which elements we want to work on.

Selecting the proper elements is, of course, our responsibility. In giving us more control over the process, D3 expects we'll "level up" on the learning curve.



NEW DATA

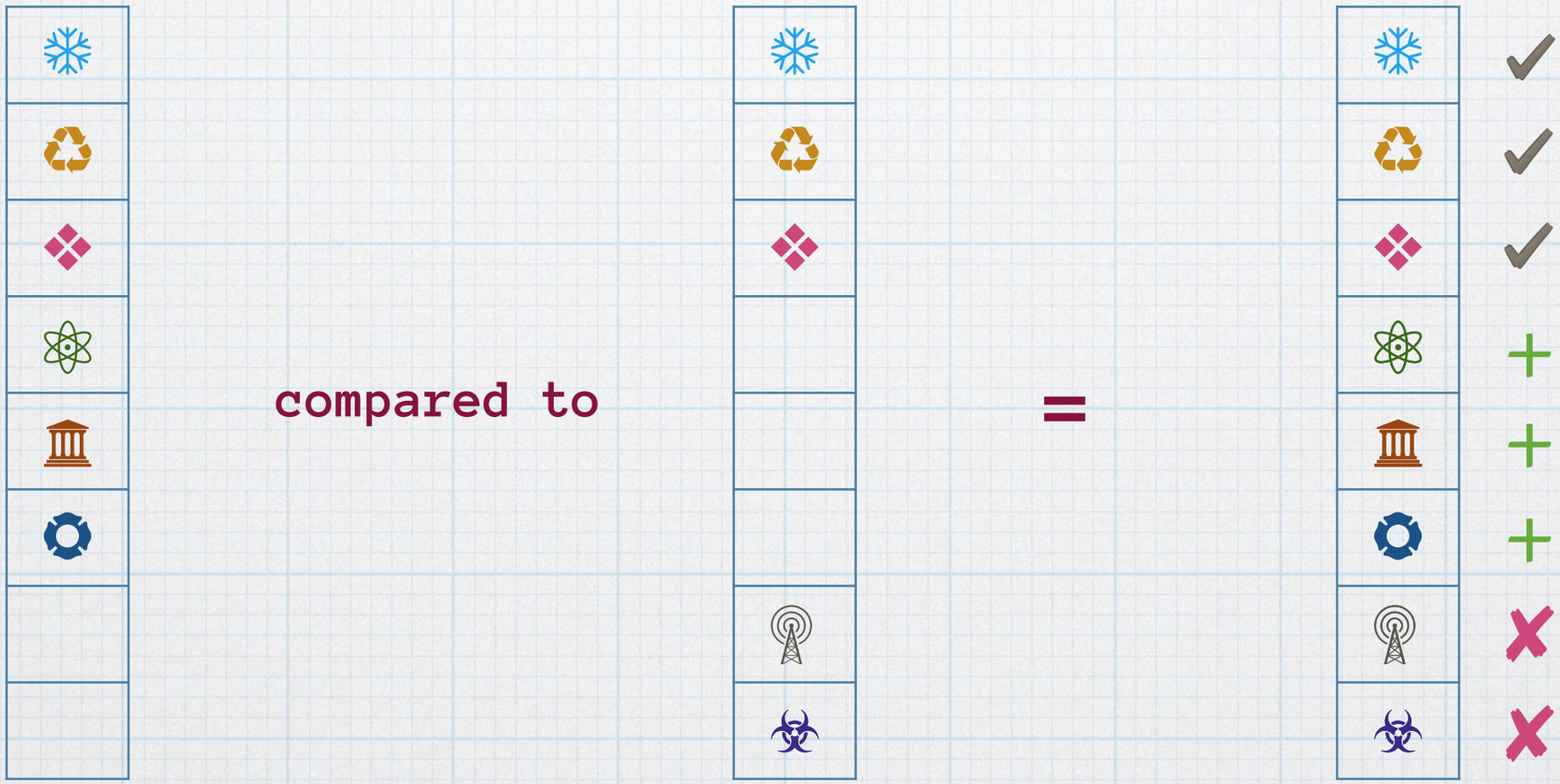
`.data()`



Likewise, D3 assumes that we will already have taken the responsibility of modifying the new dataset so that it is current, up-to-date, and correct. If it isn't, that's our fault. D3 helps us with some fancy data loading functions like **d3.csv()** and **d3.json()**, but we are free to acquire this data any way we wish.

THE UPDATE

By comparing the two datasets, D3 knows what's happening to every single data item and D3 can successfully update all three subsets of data.



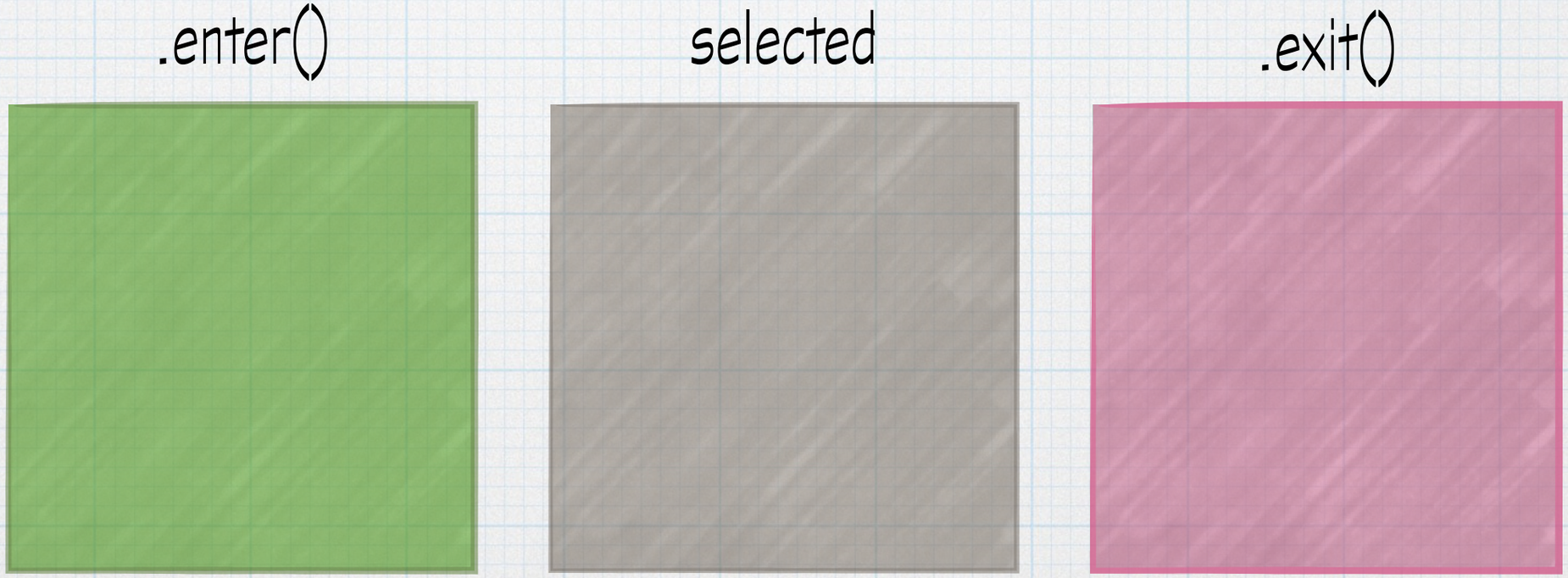
THE UPDATE

As we have seen, the three different groups are

1) the set of items that we have **selected**. (*Note: there may be more unselected items remaining in the SVG container!)*

2) a set of data items that have been added: **enter()**.

3) a set of items that have been deleted: **exit()**.

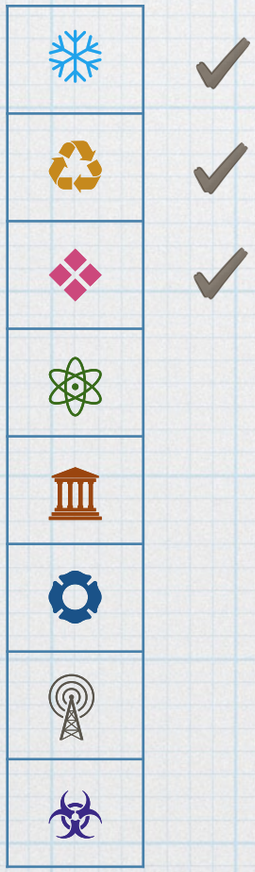


THE UPDATE

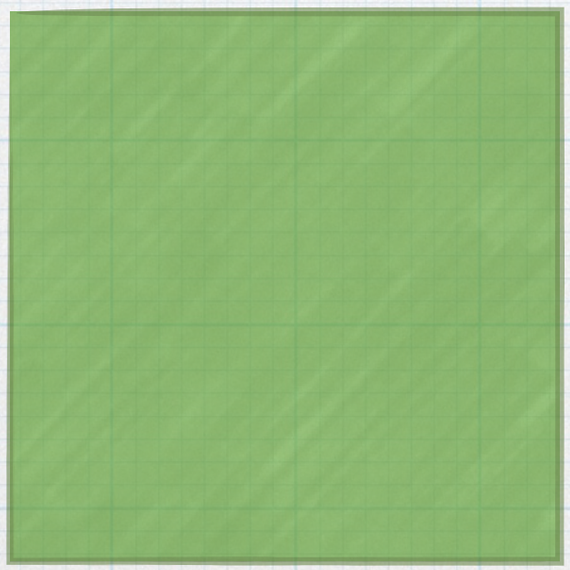
Based on the result that D3 calculates:

1) items in both sets remain in **selected**.

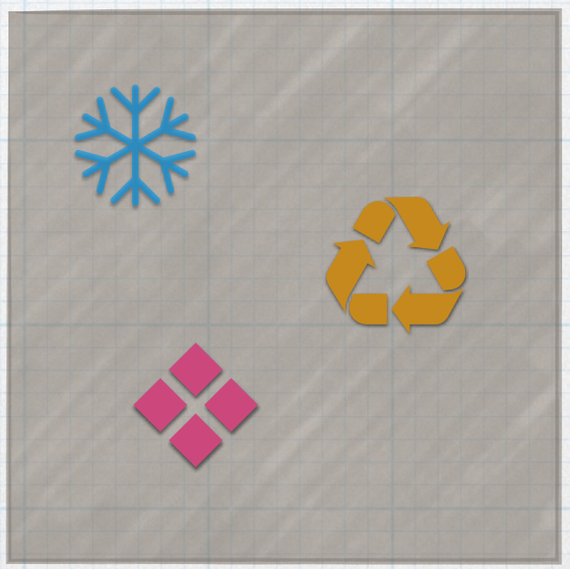
result



.enter()



selected



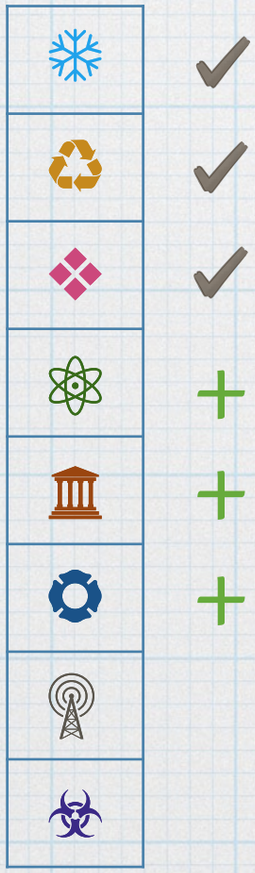
.exit()

THE UPDATE

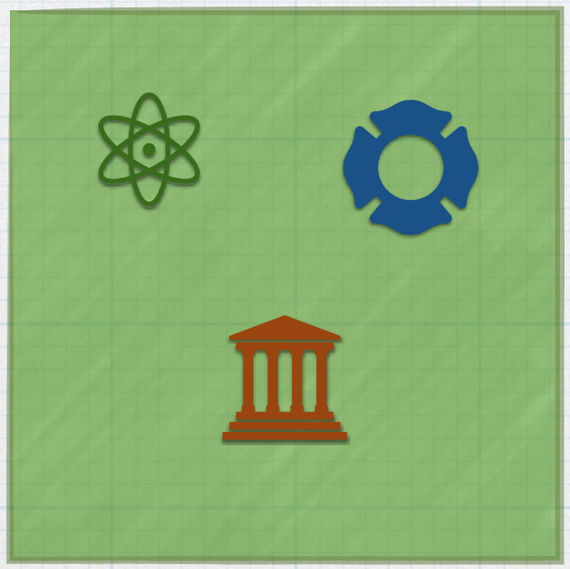
Based on the result that D3 calculates:

1) items in both sets remain in **selected**.
2) new dataset items not in **selected** go into **enter()**.

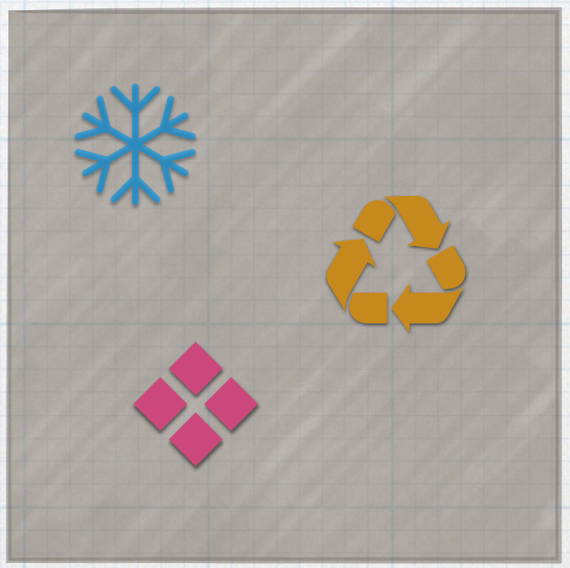
result



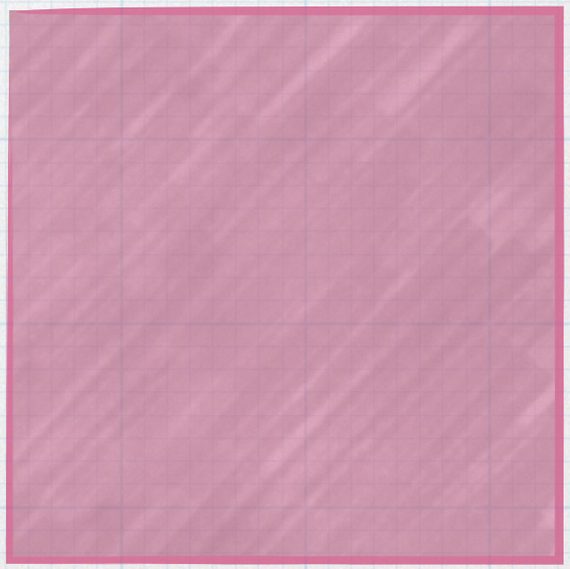
.enter()



selected



.exit()

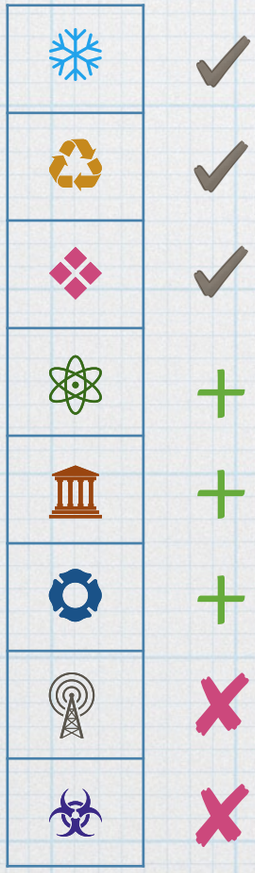


THE UPDATE

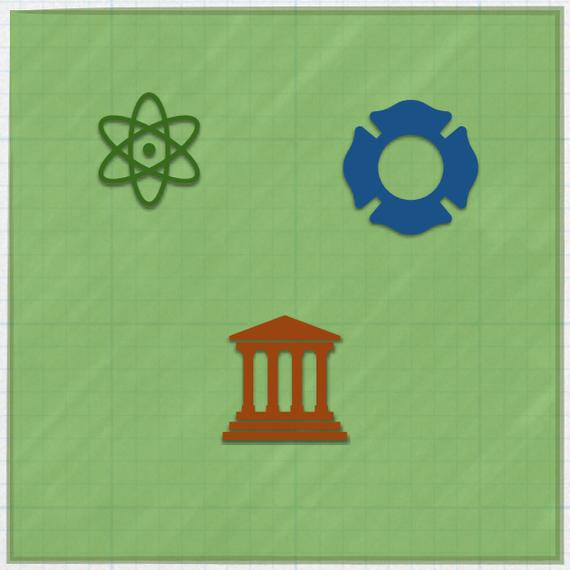
Based on the result that D3 calculates:

1) items in both sets remain in **selected**.
2) new dataset items not in **selected** go into **enter()**.
3) items still in **selected** that are not in the new dataset go into **exit()**.

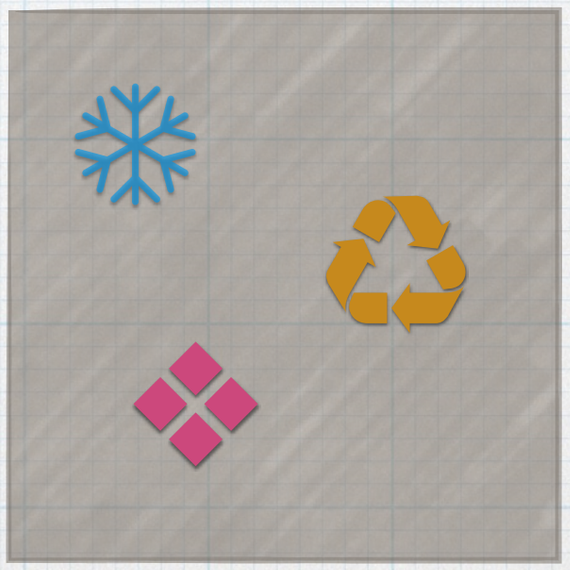
result



.enter()



selected

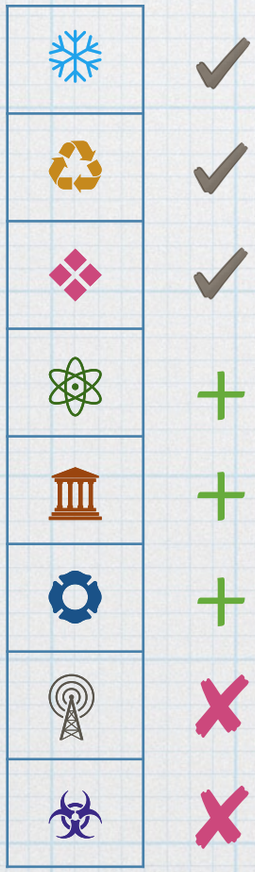


.exit()

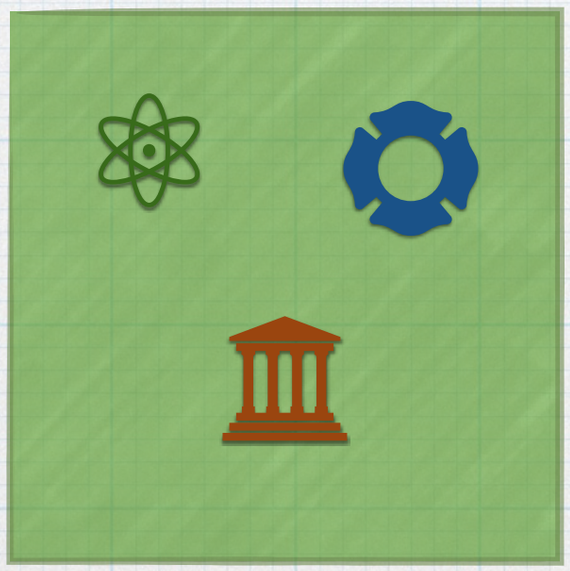
THE UPDATE

```
.selectAll()
.data()
```

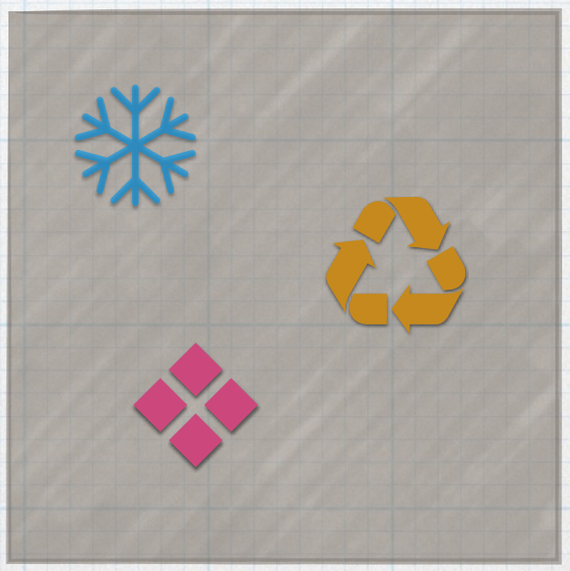
Executing these two lines in D3 creates all three of these sets. From here, we can now choose which sets to work with and what to do with them.



.enter()



selected

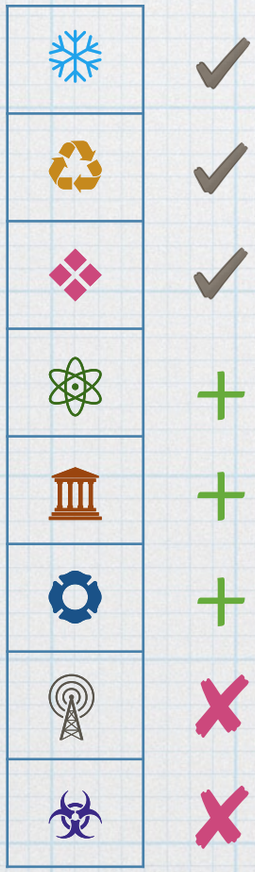


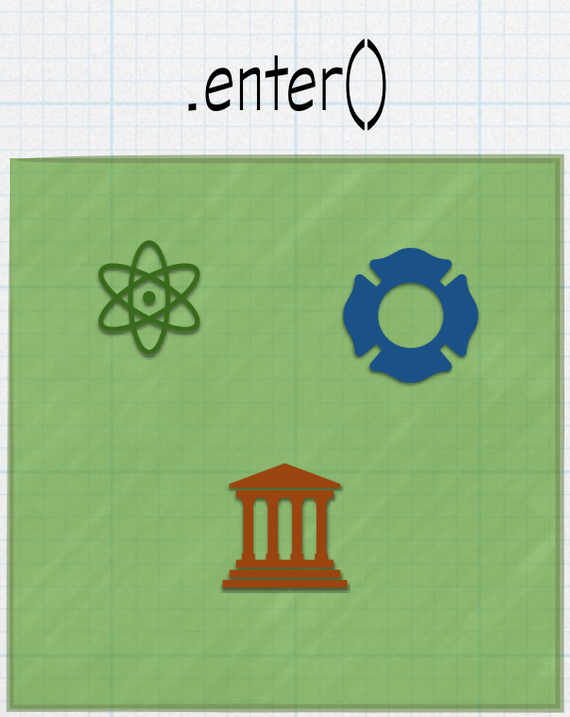
.exit()

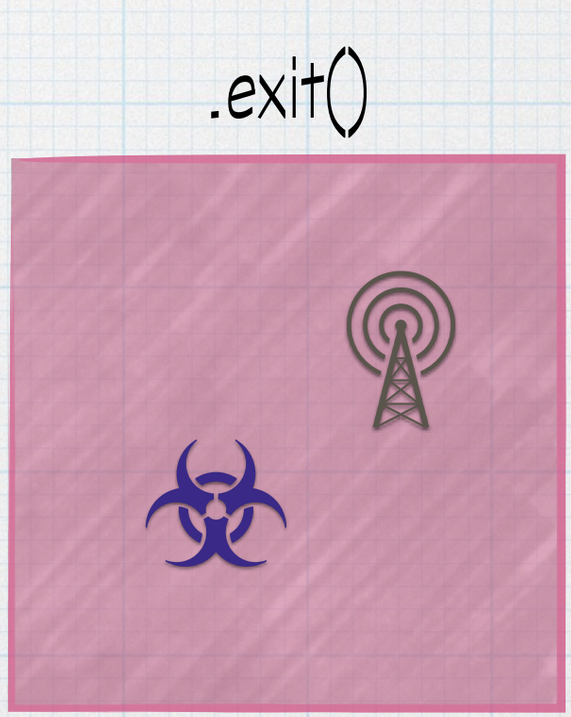
WORKING WITH SETS

```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

For example, we could work with the **.enter()** set and append an circle to the SVG container for each data item in the set.





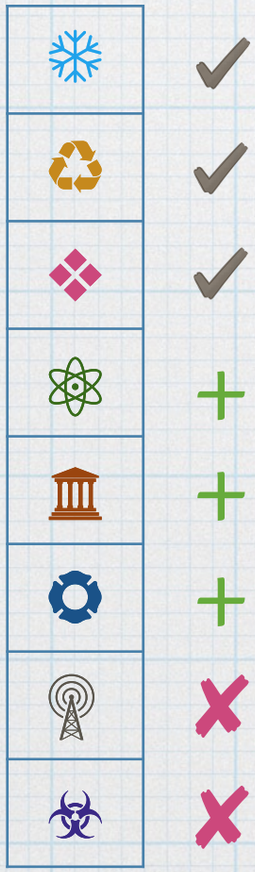


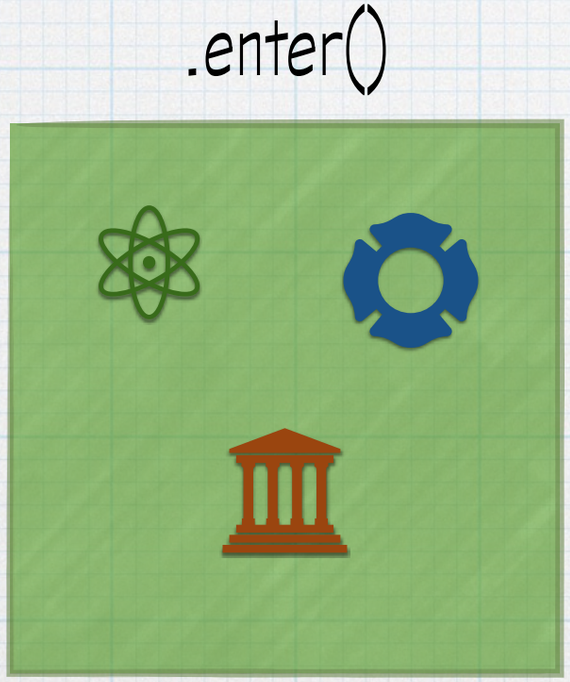
WORKING WITH SETS

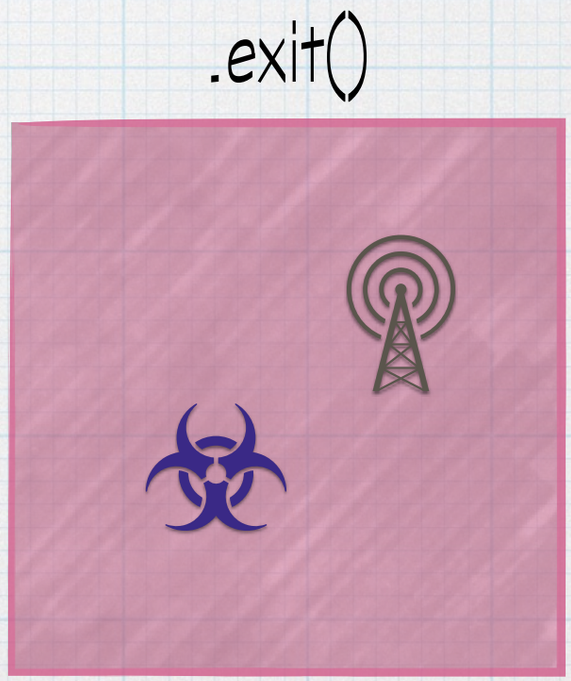
```
.selectAll('circle')
.data(my_data)
.enter()
.append('circle')
```

And/or we could choose to remove from the SVG container all the elements that have been moved to the **.exit()** dataset.

```
.selectAll('circle')
.data(my_data)
.exit()
.remove()
```



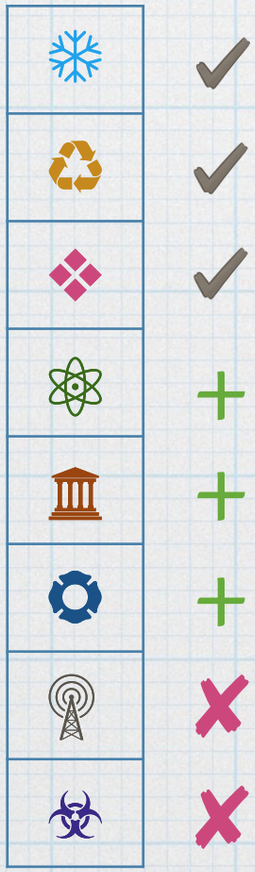


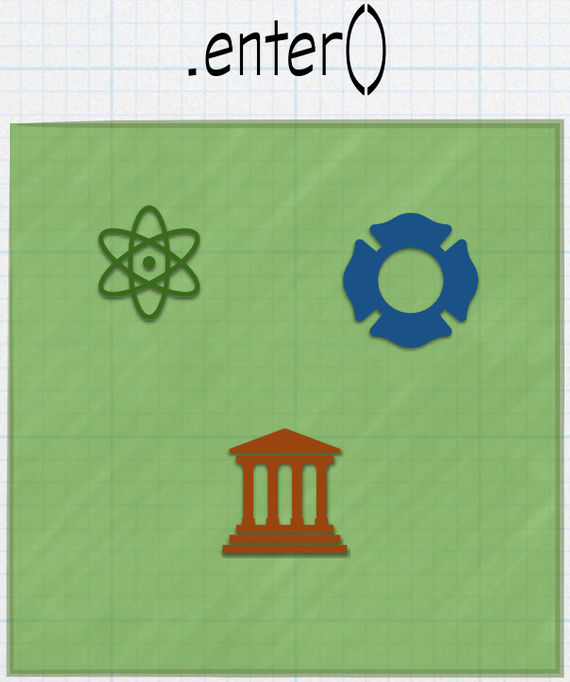


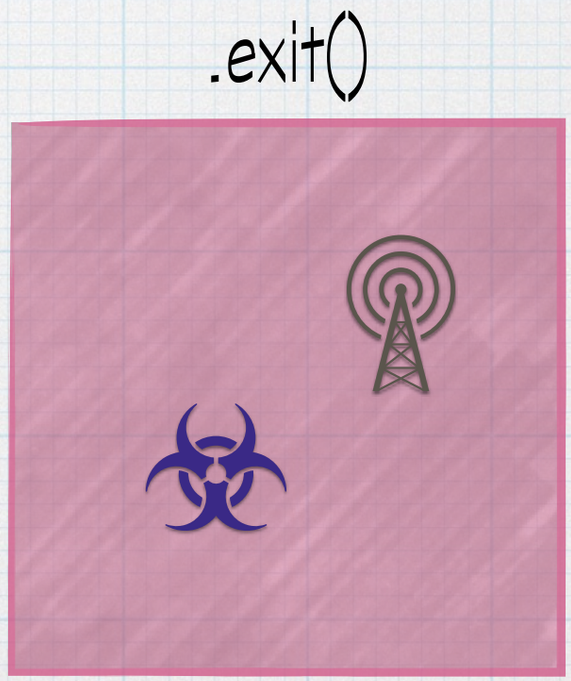
WORKING WITH SETS

Or we could modify all the elements in the selection that were also there previously.

```
.selectAll('circle')
.data(my_data)
.attr('r', blah)
.attr('fill', blah)
.attr('stroke', blah)
```







D3 Data Binding

select | enter | exit