Updating Data

Lars Verspohl
VISUALIZATION DESIGNER AND DEVELOPER

@lars_vers www.datamake.io

Overview

General update pattern

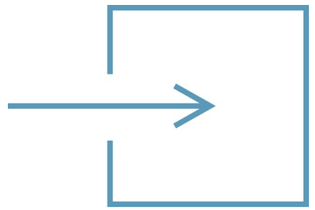
The `join` method

The `key` function

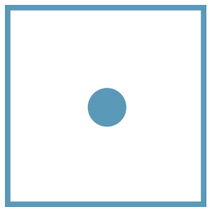
D3 transitions

The general update pattern allows you to keep updated data in sync with your visual elements on screen.

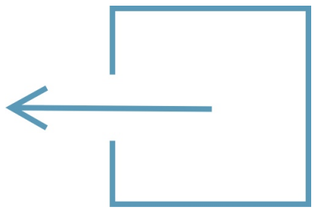
Data Change States



The **enter** state: new data rows



The **update** state: remaining data rows



The **exit** state: leaving data rows

Favorite Fruit Lists

Biff

1. Apples
2. Oranges
3. Lemons

Chip

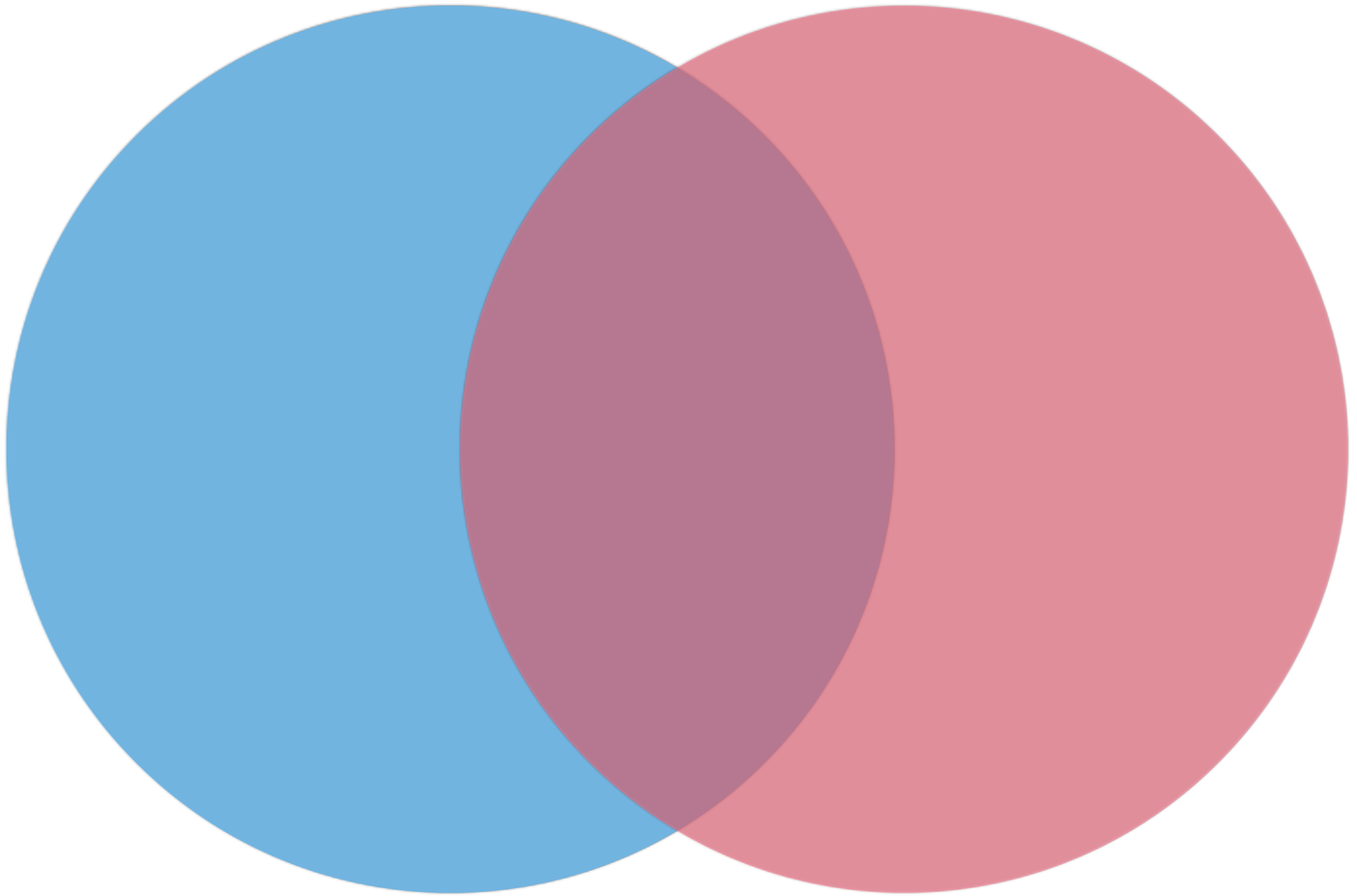
1. Apples
2. Oranges

Kipper

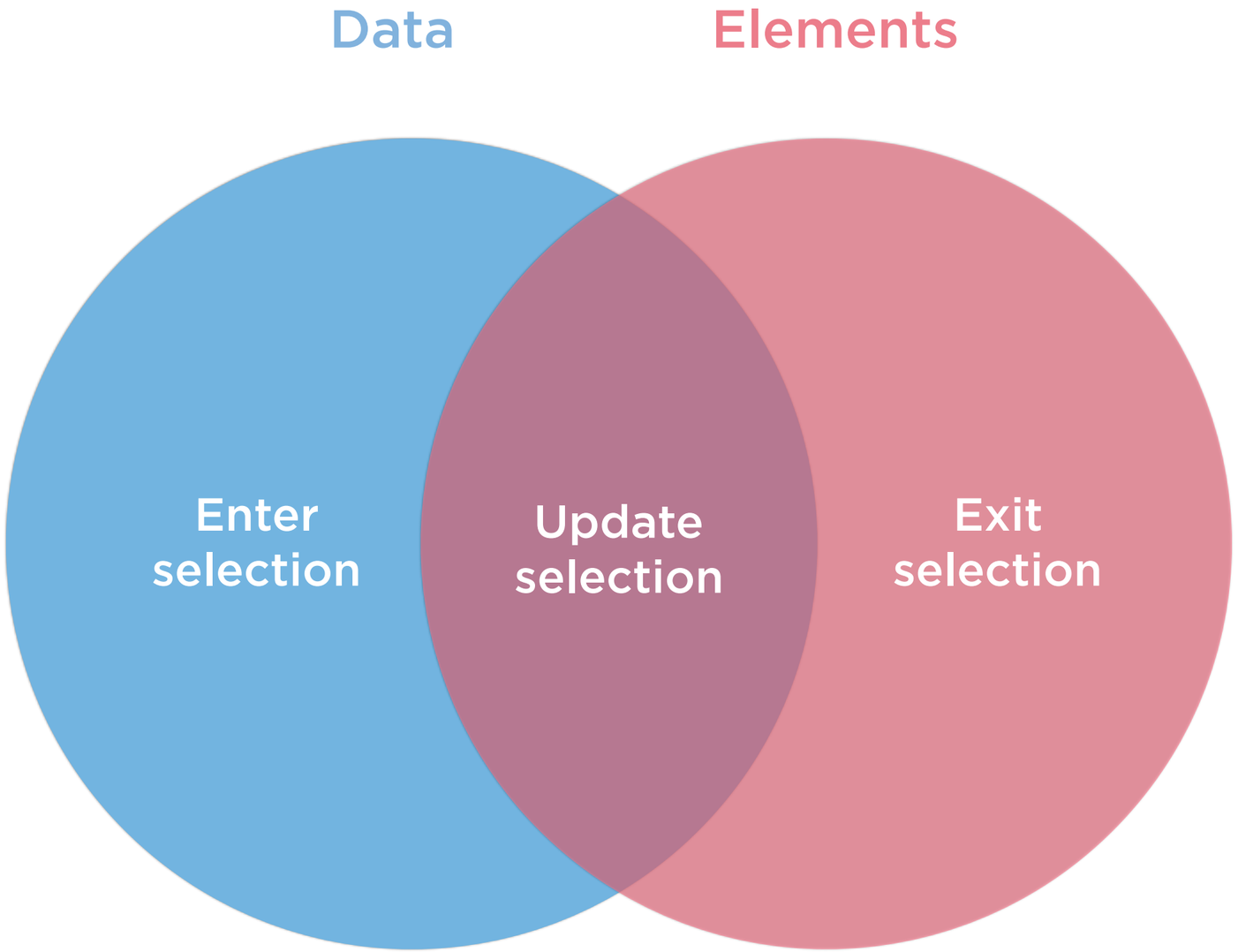
1. Apples
2. Cherries
3. Peaches
4. Oranges

Data

Elements



1. Enter
2. Update
3. Exit


Data
Elements
Enter selection
Update selection
Exit selection

Data

Apples
Oranges
Lemons

Elements

<text>Apples</text>

<text>Cherries</text>

<text>Peaches</text>

<text>Oranges</text>

Data  **Elements**

Apples
Oranges
Lemons

<text>Apples</text>

<text>Cherries</text>

<text>Peaches</text>

<text>Oranges</text>

Data

Apples
Oranges

Elements

<text>Apples</text>

<text>Cherries</text>

<text>Peaches</text>

<text>Oranges</text>

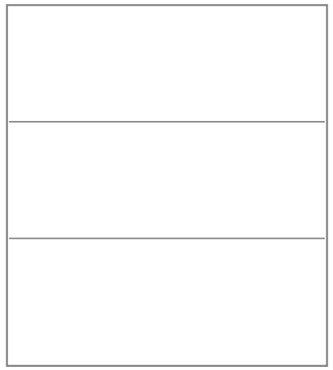
Enter selection

Update selection

Exit selection

Lemons

Data



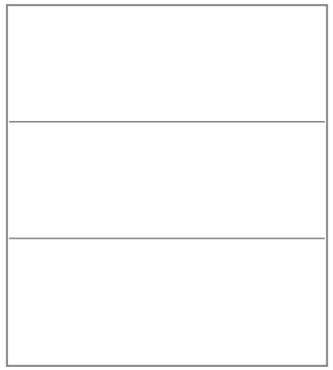
Elements

<text>Cherries</text>

<text>Peaches</text>

Enter selection	Update selection	Exit selection
Lemons	<text>Apples</text>	
	<text>Oranges</text>	

Data



Elements

Enter selection	Update selection	Exit selection
Lemons	<text>Apples</text>	<text>Cherries</text>
	<text>Oranges</text>	<text>Peaches</text>

Enter selection	Update selection	Exit selection
Lemons	<text>Apples</text>	<text>Cherries</text>
	<text>Oranges</text>	<text>Peaches</text>

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
	<text>Oranges</text>	<text>Peaches</text>

Create as text elements

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
	<text>Oranges</text>	<text>Peaches</text>

Create as text elements

Fill blue

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
	<text>Oranges</text>	<text>Peaches</text>

Create as text elements

Fill blue

Change to gray

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
		<text>Peaches</text>
	<text>Oranges</text>	

Create as text elements

Fill blue

Change to gray

Reposition

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
		<text>Peaches</text>
	<text>Oranges</text>	

Create as text elements	Change to gray	Fill red
Fill blue	Reposition	

Enter selection	Update selection	Exit selection
<text>Lemons</text>	<text>Apples</text>	<text>Cherries</text>
		<text>Peaches</text>
	<text>Oranges</text>	

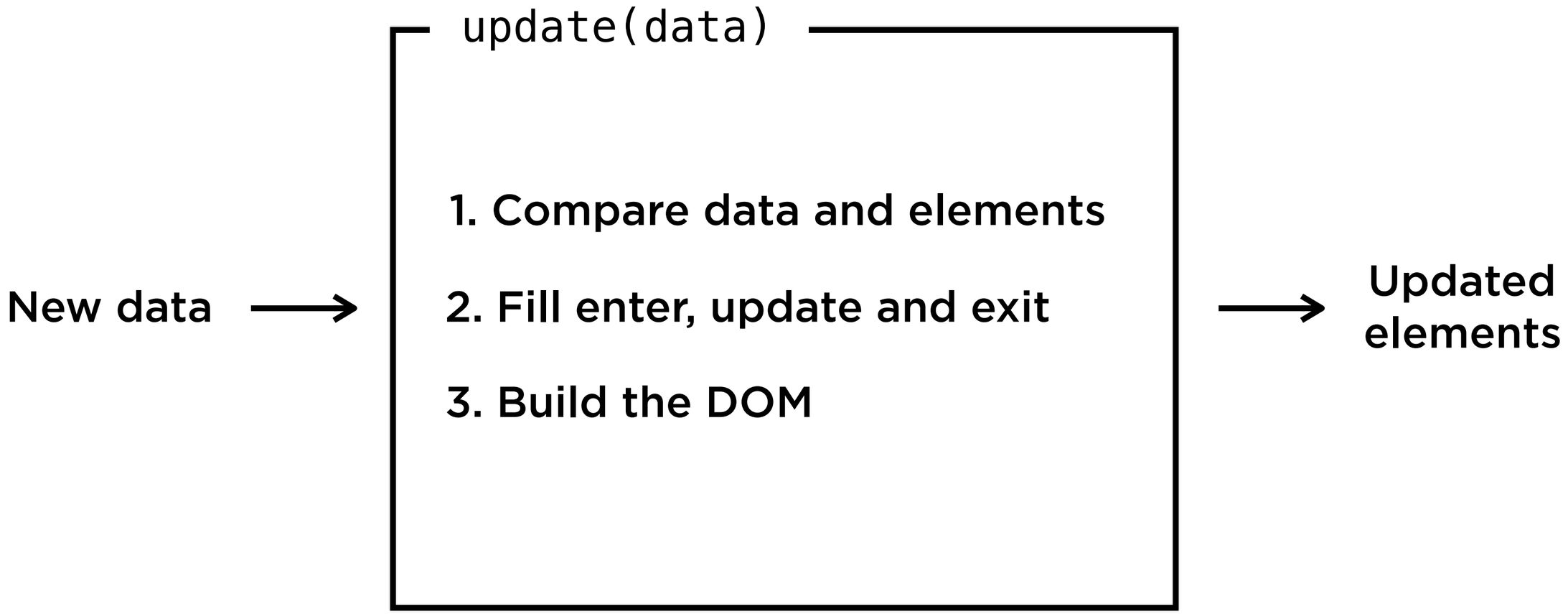
Enter selection	Update selection	Exit selection
Create as text elements	Change to gray	Fill red
Fill blue	Reposition	Remove

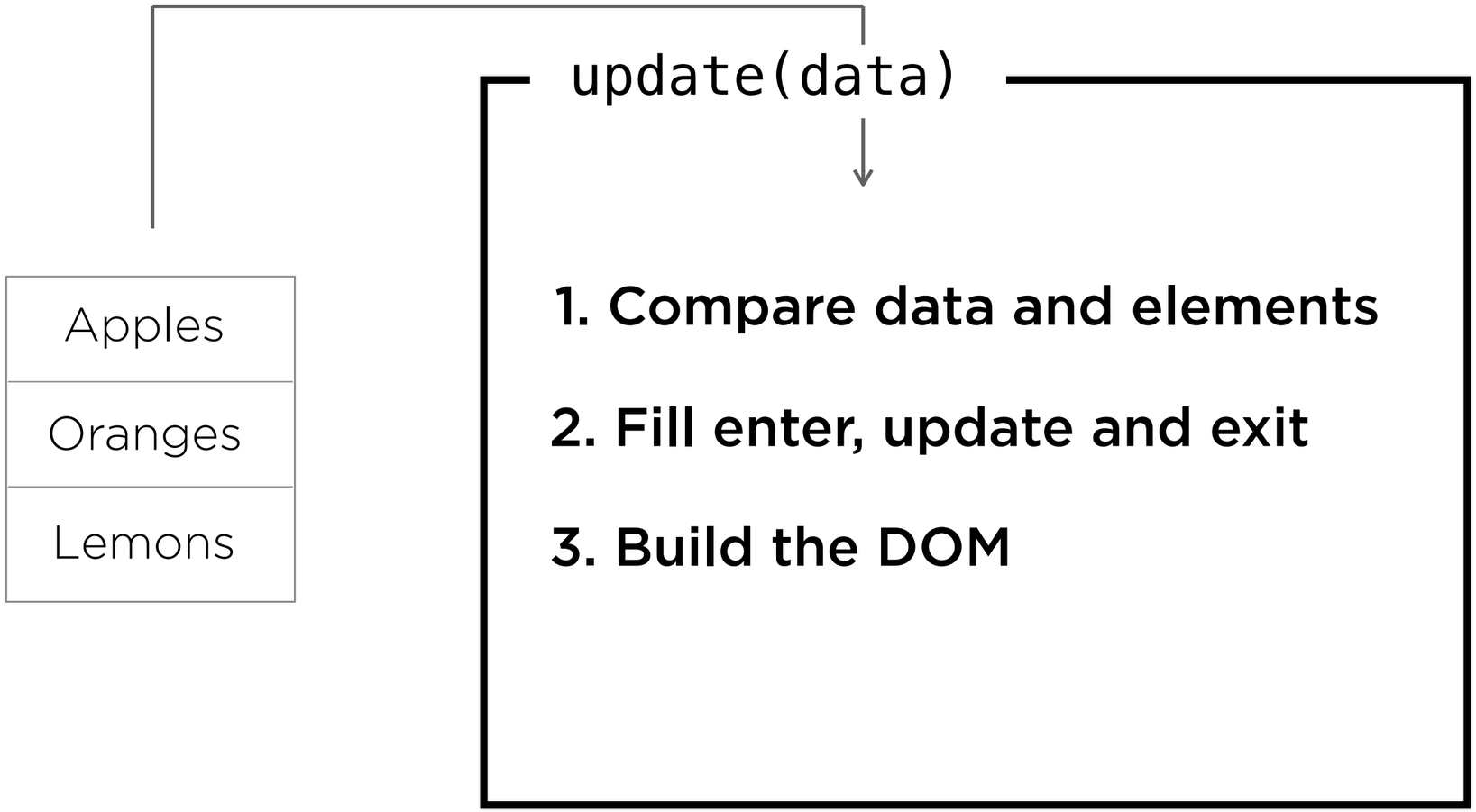
function

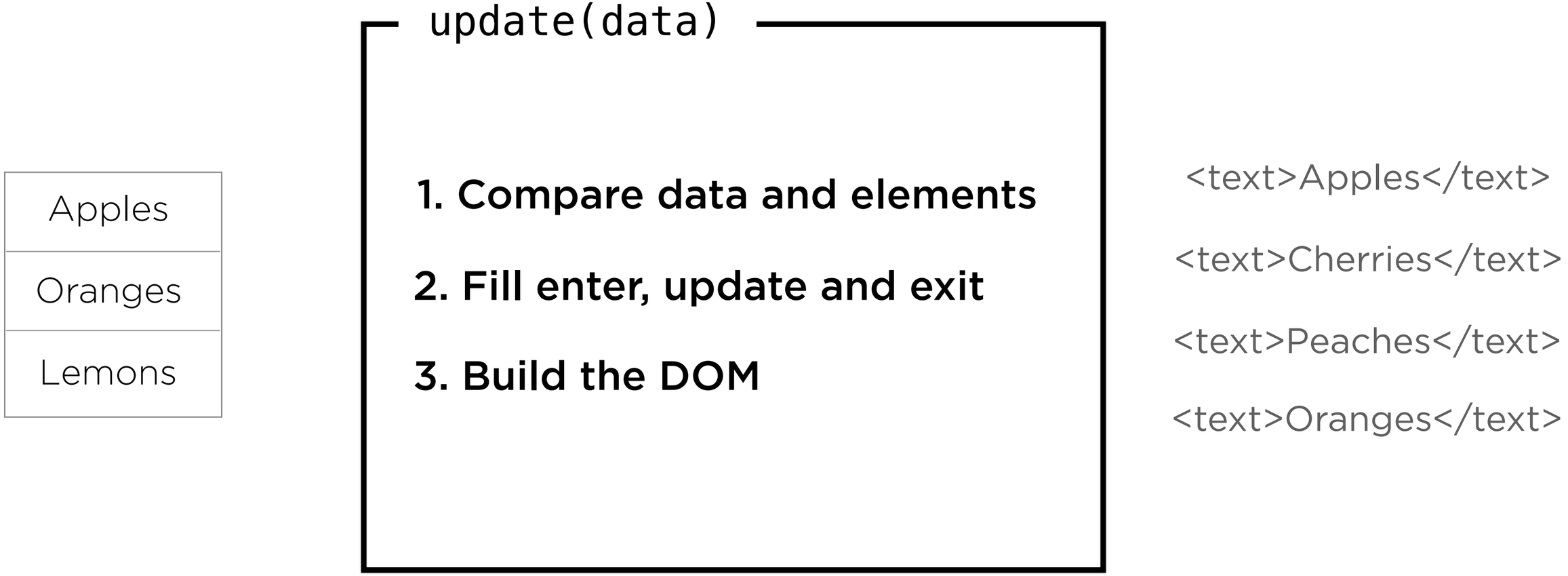
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM

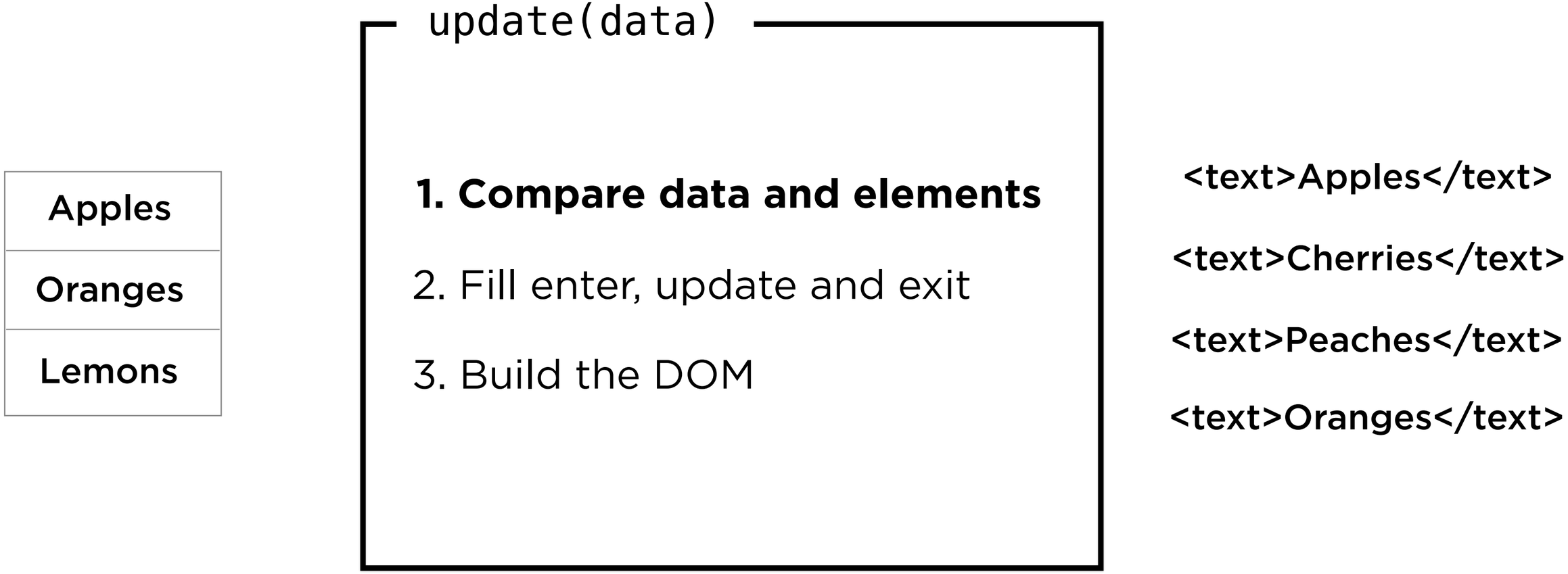
`update()`

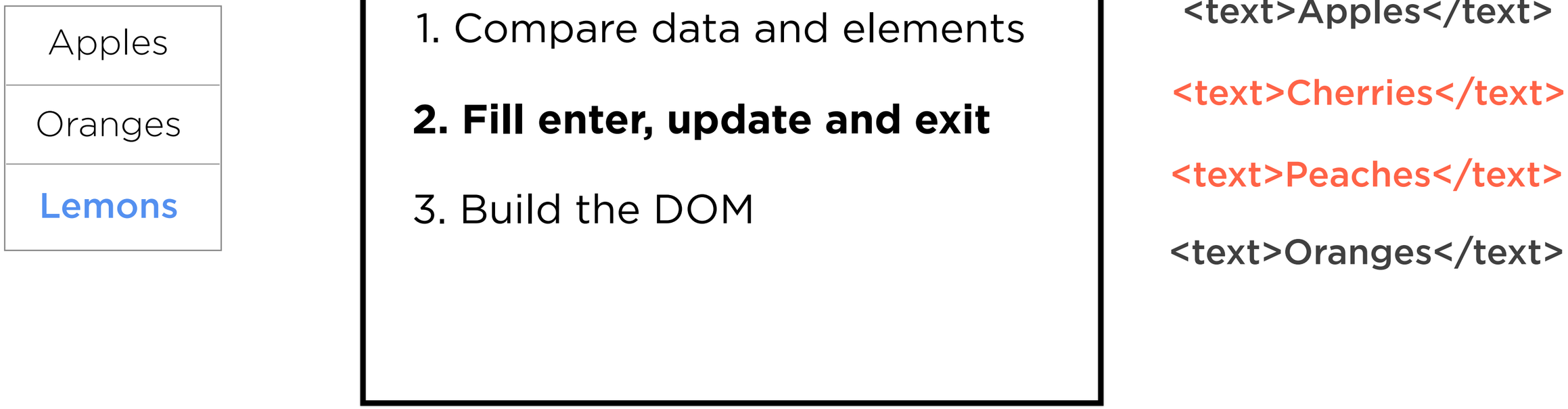
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM

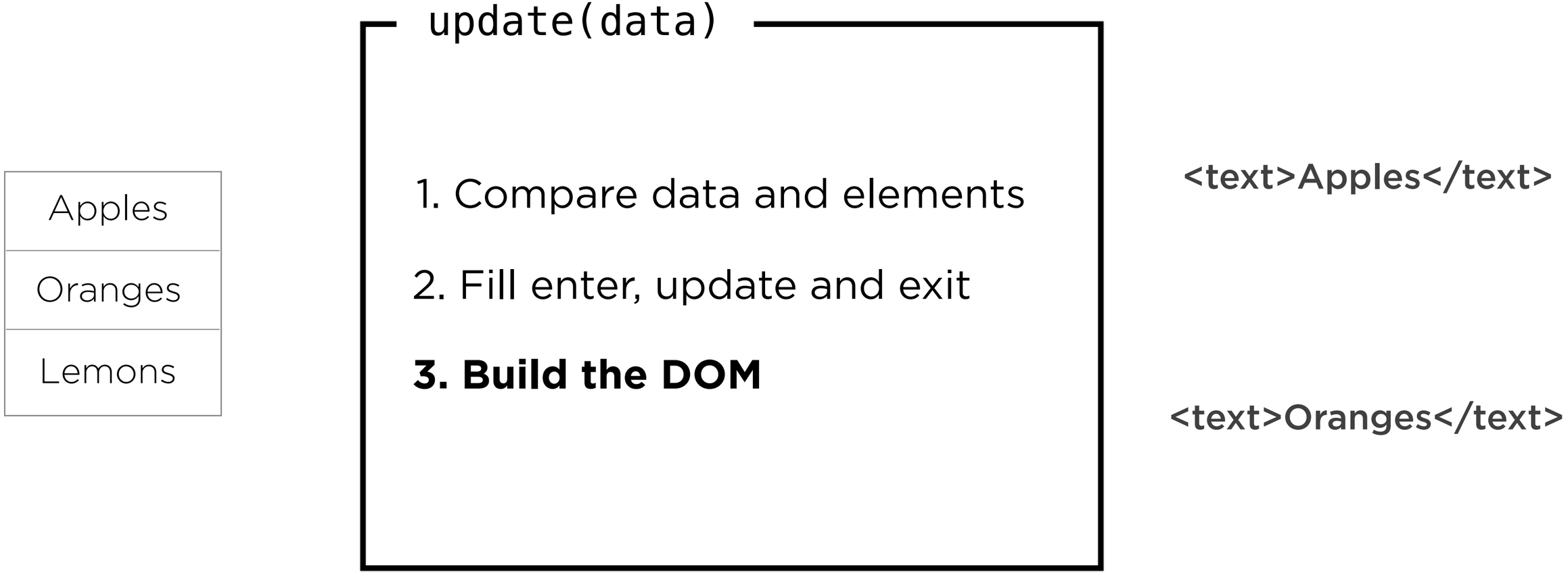

update(data)
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
New data
Updated elements

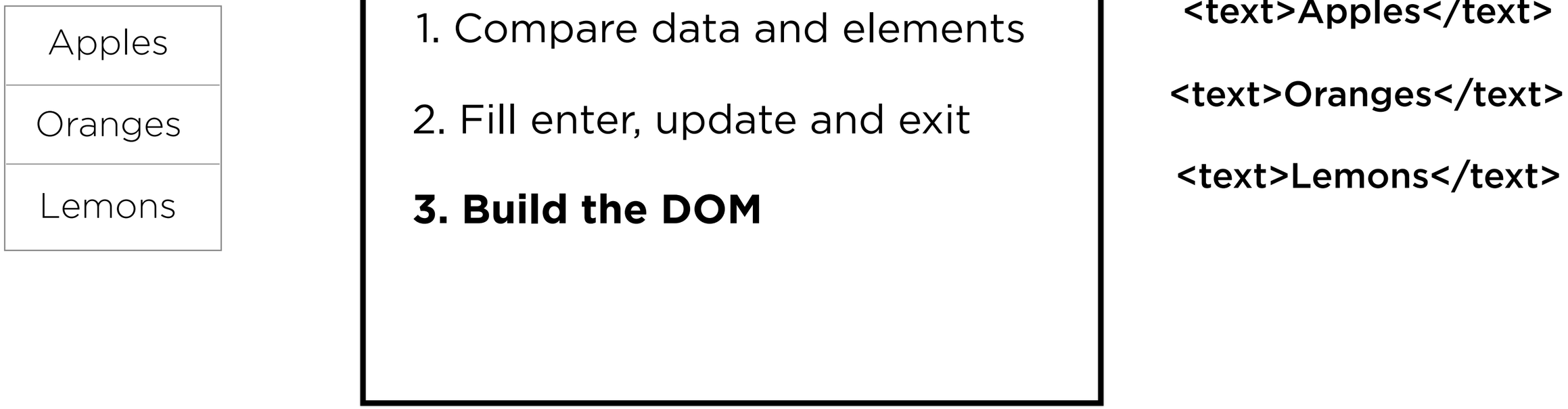

update(data)
Apples
Oranges
Lemons
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM


Apples
Oranges
Lemons
update(data)
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
<text>Apples</text>
<text>Cherries</text>
<text>Peaches</text>
<text>Oranges</text>


Apples
Oranges
Lemons
update(data)
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
<text>Apples</text>
<text>Cherries</text>
<text>Peaches</text>
<text>Oranges</text>


Apples
Oranges
Lemons
update(data)
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
<text>Apples</text>
<text>Cherries</text>
<text>Peaches</text>
<text>Oranges</text>


update(data)
Apples
Oranges
Lemons
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
<text>Apples</text>
<text>Oranges</text>


Apples
Oranges
Lemons
update(data)
1. Compare data and elements
2. Fill enter, update and exit
3. Build the DOM
<text>Apples</text>
<text>Oranges</text>
<text>Lemons</text>

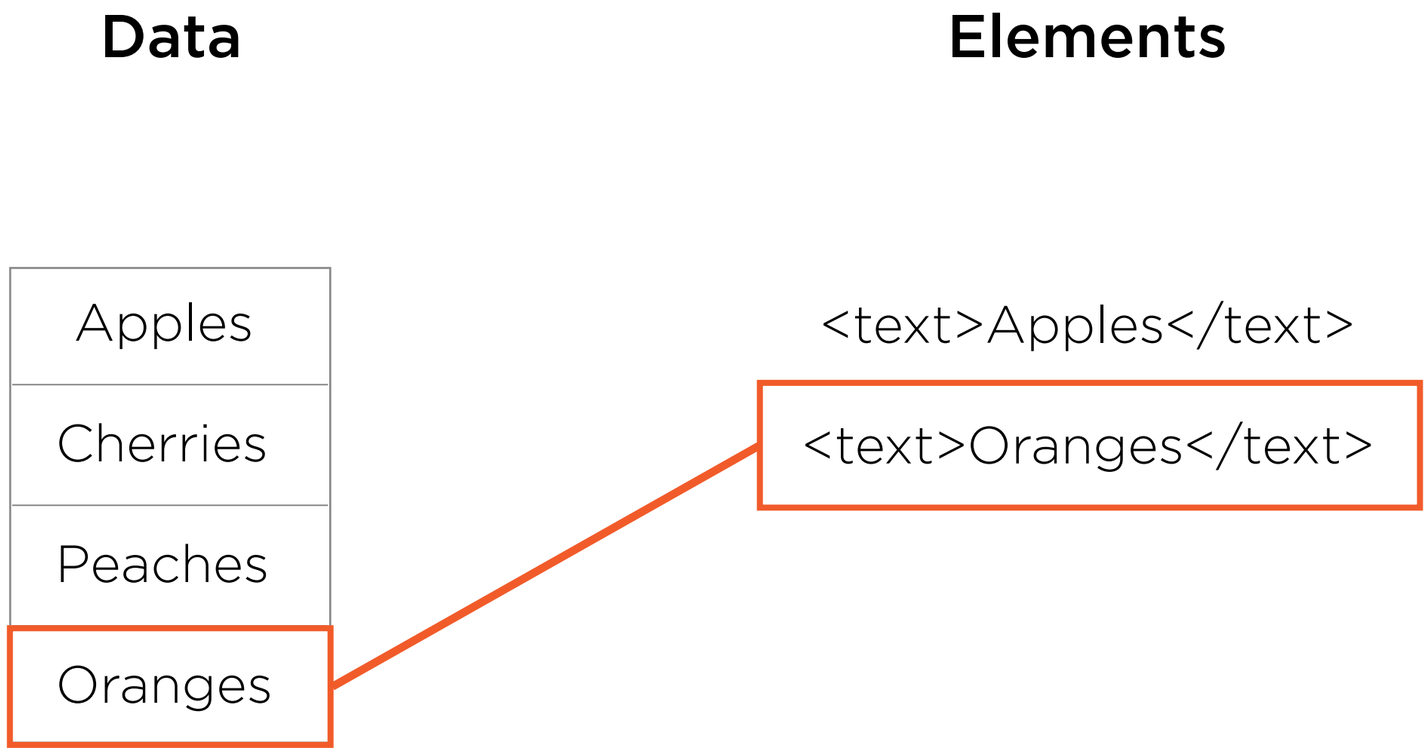
Data

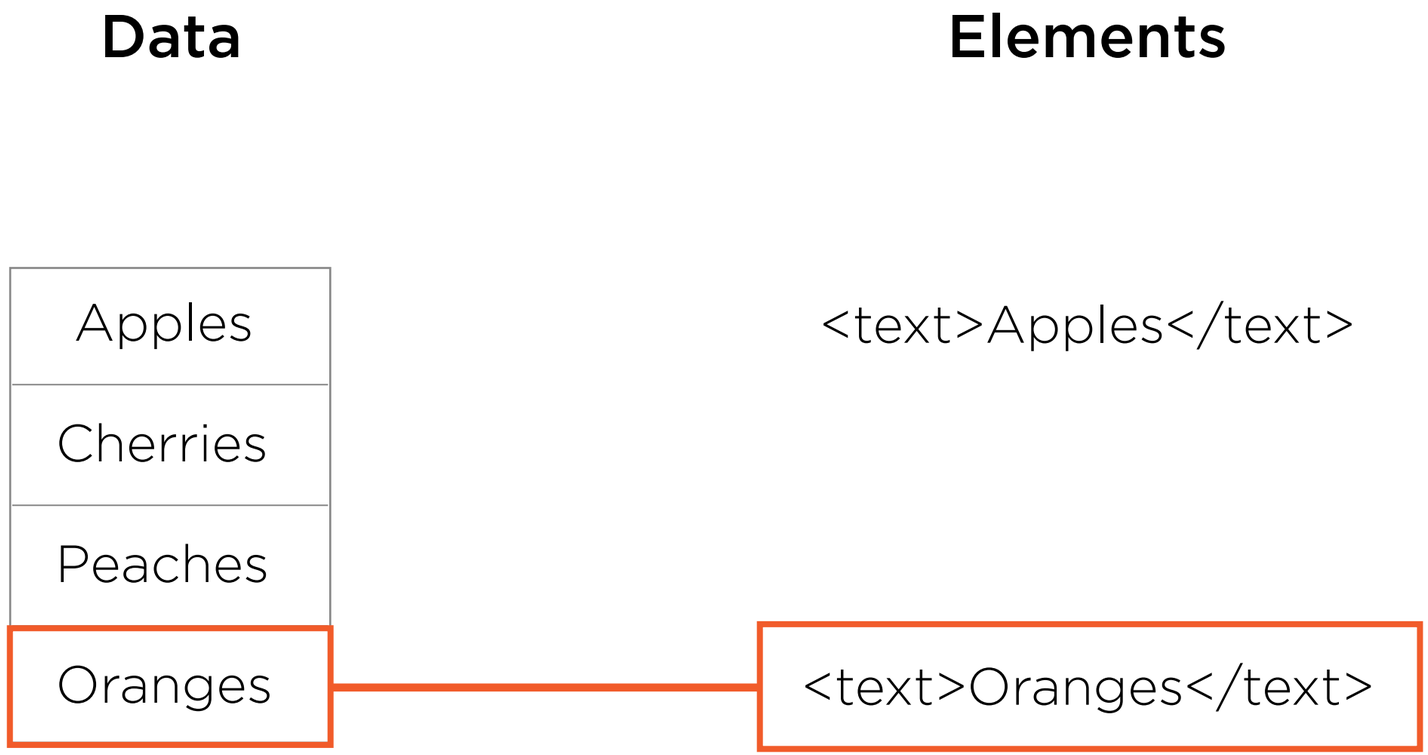
Apples
Oranges

Elements

<text>Apples</text>

<text>Oranges</text>


Data
Elements
Apples
Cherries
Peaches
Oranges
<text>Apples</text>
<text>Oranges</text>


Data
Elements
Apples
Cherries
Peaches
Oranges
<text>Apples</text>
<text>Oranges</text>

Data

index: 0
index: 1

Elements

<text> *index: 0* </text>

<text> *index: 1* </text>

Chip

Data

Apples	*index: 0*
Cherries	*index: 1*
Peaches	*index: 2*
Oranges	*Index: 3*

Elements

<text> *index: 0* </text> Apples

<text> *index: 1* </text> Oranges

<text> *index: 2* </text> Peaches

<text> *index: 3* </text> Oranges

Key Function

```
update(data)

  svg
   .selectAll(‘text’)
   .data(data)
   .join(
    enter,
    update,
    exit
   );
```

Key Function

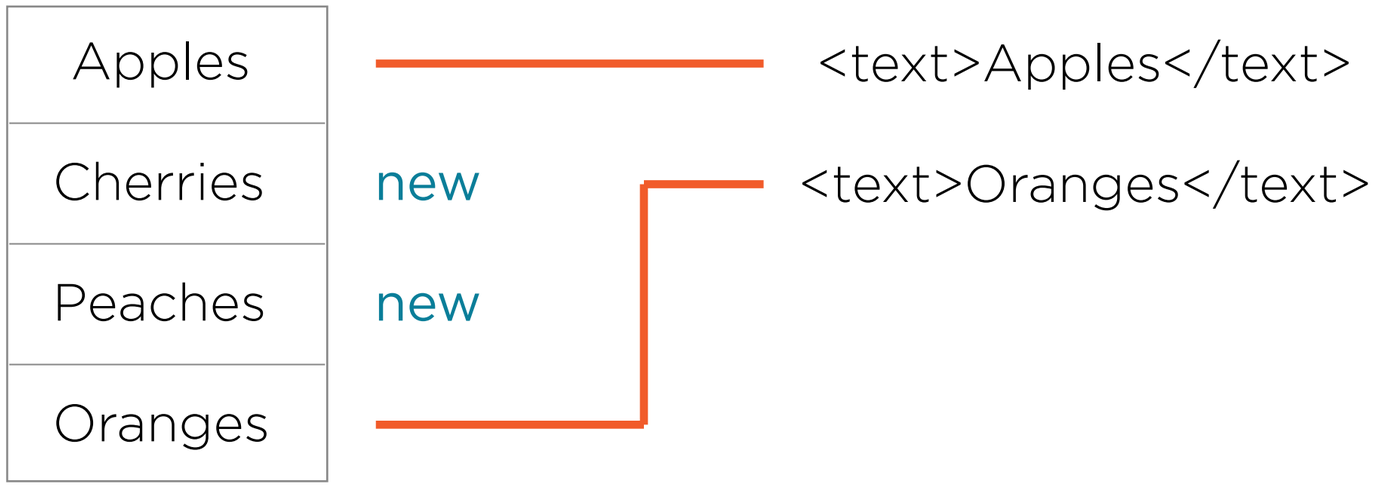
```
update(data)

  svg
   .selectAll('text')
   .data(data, d => d)
   .join(
    enter,
    update,
    exit
   );
```

join-by-identifier

Key Function

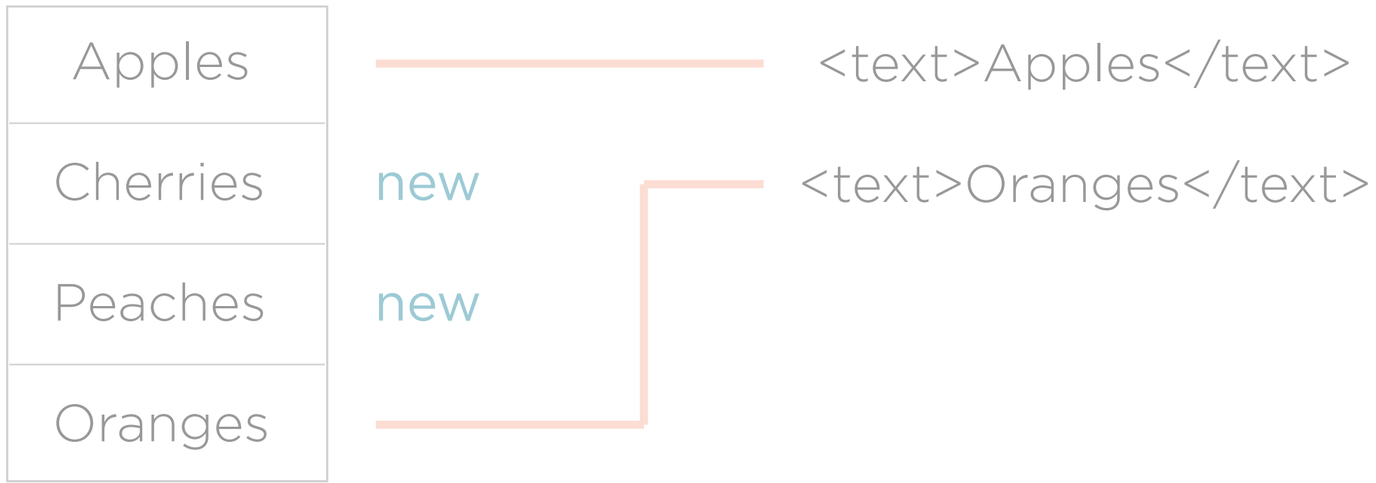
```
.data(data, d => d)
```



join-by-identifier

Key Function

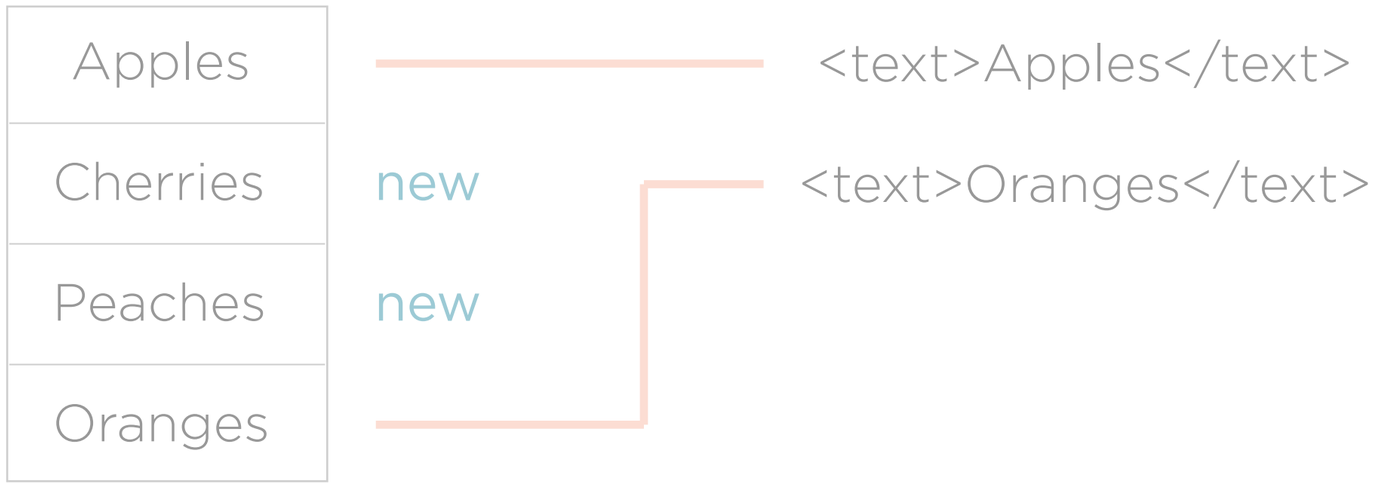
```
.data(data, d => d.fruitName)
```



join-by-identifier

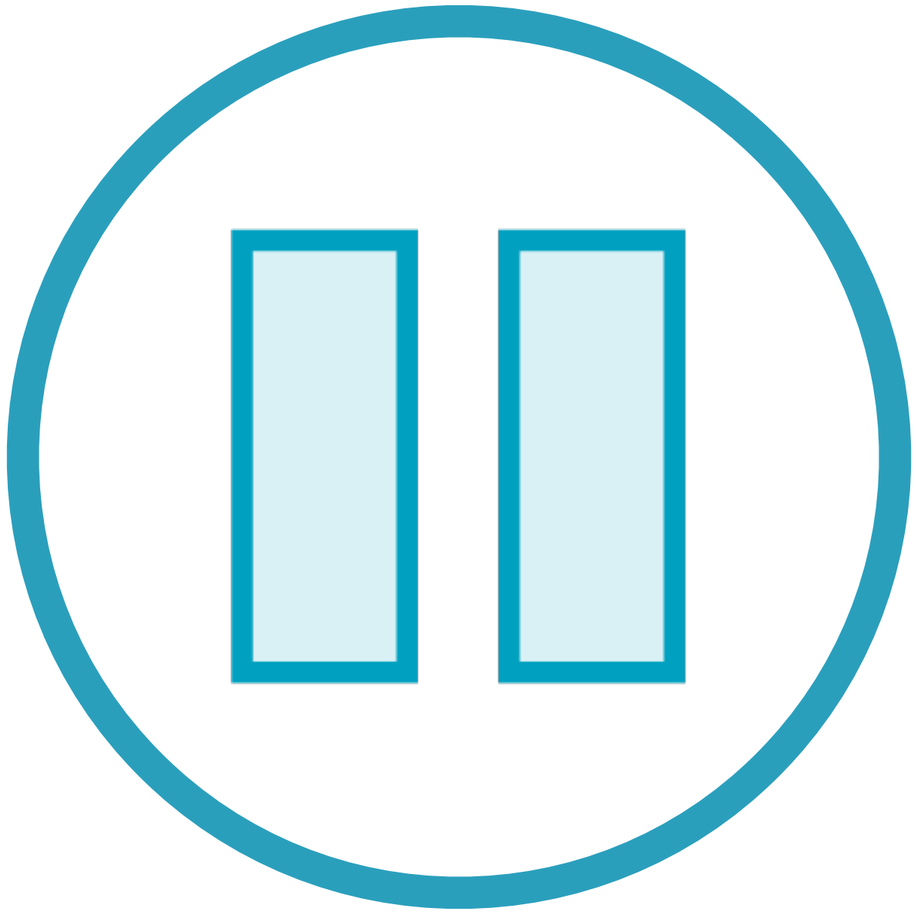
Key Function

```
.data(data, d => d.value)
```



join-by-identifier

Old update and simple join



Data Join with `.join()`

```
function update(data) {
  svg
    .selectAll('text')
    .data(data, d => d)
    .join(

      enter => enter
          .append('text')
          .text(d => d)
          .attr('x', 30)
          .attr('y', (d, i) => i * 30 + 50)
          .style('fill', 'dodgerblue'),

      update => update
          .style('fill', ‘gray')
          .attr('y', (d, i) => i * 30 + 50),

      exit => exit.remove()

    );
}
```

Data Join without `.join()`

```
function update(data) {
  // The data join.
  const fruitList = svg
     .selectAll(‘text')
     .data(data, d => d);

  // The update selection.
  fruitList
     .style('fill', ‘gray')
     .attr('y', (d, i) => i * 30 + 50);

  // The enter selection.
  fruitList
    .enter()
    .append('text')
    .text(d => d)
    .attr('x', 30)
    .attr('y', (d, i) => i * 30 + 50)
    .style('fill', 'dodgerblue');

  // The exit selection.
  fruitList.exit().remove();
}
```

Data Join with `.join()`


```
function update(data) {
  svg
    .selectAll('text')
    .data(data, d => d)
    .join(

      enter => enter
          .append('text')
          .text(d => d)
          .attr('x', 30)
          .attr('y', (d, i) => i * 30 + 50)
          .style('fill', 'dodgerblue'),

      update => update
          .style('fill', ‘gray')
          .attr('y', (d, i) => i * 30 + 50),

      exit => exit.remove()

    );
}
```

Simple `.join()`

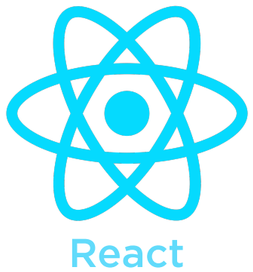
```
function update(data) {
  svg
    .selectAll('text')
    .data(data, d => d)
    .join('text')
    .attr('x', 30)
    .attr('y', (d, i) => i * 30 + 50)
    .style('fill', 'dodgerblue'),
}
```

Simple Join

```
function update(data) {
  svg
    .selectAll('text')
    .data(data, d => d)
    .enter()
    .append('text')
    .attr('x', 30)
    .attr('y', (d, i) => i * 30 + 50)
    .style('fill', 'dodgerblue'),
}
```

Transition intro

Alternatives

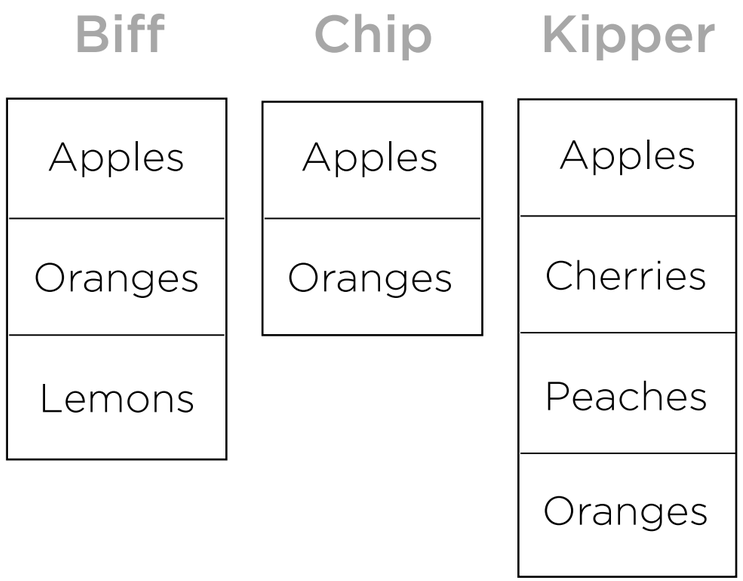


- Bind data to elements
- Update the DOM on data change
- Update internal in React and Vue
- D3 benefit: transitions

Transitions

- Object constancy
- Relationship between states
- Sometimes more efficient

Transitions



- Object constancy
- Relationship between states
- Sometimes more efficient
- Visually engaging
- Less cognitive burden
- D3’s in-built transitions

Summary

General update pattern

The `join` method

Control the join with the `key` function

D3 transition capabilities

Foray into interactions