



# Utility Methods

DISPLAY FAVORITE

Favorite

# Ajax & JSON Review

application.js

```
$( 'button' ).on( 'click', function() {  
    $.ajax( '/cities/favorite/1', {  
        contentType: 'application/json',  
        dataType: 'json',  
        success: function( result ) {  
            var favorite = $( '.favorite' );  
            favorite.find( 'p' ).html( result.name );  
            favorite.find( 'img' )  
                .attr( 'src', result.image );  
        }  
    } );  
});
```

*result will be a JSON object*

result

```
{  
    image: "images/paris.png",  
    name: "Paris, France",  
}
```

index.html

```
<div class='favorite'>  
    <h3>Favorite</h3>  
    <img src='' />  
    <p></p>  
    <button>Show Favorite</button>  
</div>
```

*What if we had multiple favorites?*

# Multiple Results from JSON

*Result is now an array of objects*

result

```
[
  {
    image: "images/paris.png",
    name: "Paris, France",
  },
  {
    image: "images/london.png",
    name: "London, UK"
  },
  {
    image: "images/madrid.png",
    name: "Madrid, Spain"
  }
]
```

application.js

```
success: function(result) {
  var favorite = $('.favorite');
  favorite.find('p').html(result.name);
  favorite.find('img')
    .attr('src', result.image);
}
```

*Need to 'loop' over each favorite*

# Use \$.each() to iterate through the array

application.js

```
success: function(result) {  
  $.each(result, function(index, city) {  
    var favorite = $('#favorite-' + index);  
    favorite.find('p').html(city.name);  
    favorite.find('img')  
      .attr('src', city.image);  
  });  
}
```



index.html

```
<div class='favorite-0'>  
  ...  
</div>  
  
<div class='favorite-1'>  
  ...  
</div>  
  
<div class='favorite-2'>  
  ...  
</div>
```

*Utility Method*

```
$.each(collection, function(<index>, <object>) {})
```



*Calls the function for each  
object in the collection*

DISPLAY FAVORITES

Favorite 0

Favorite 1

Favorite 2

**UPDATE FLIGHT**

JFK - New York, NY

Departing Location

SFO - San Francisco, CA


Destination Location



# Transforming an array of objects into html

application.js

```
$( '.update-status' ).on( 'click', function() {  
    $.ajax( '/status', {  
        contentType: 'application/json',  
        dataType: 'json',  
        success: function( result ) { ... }  
    } );  
});
```



```
$.getJSON( url, success );
```

*result will be an array of  
JavaScript Objects*

```
$.getJSON( '/status', function( result ) { } );
```

Result

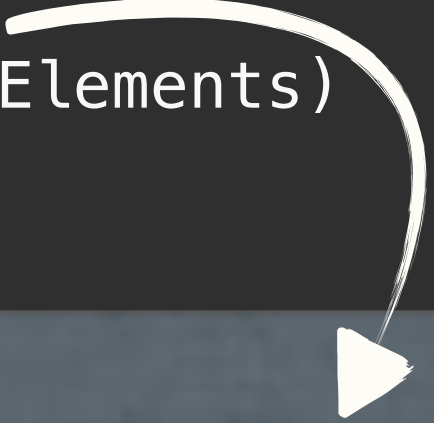
```
[  
  {  
    name: 'JFK – New York, NY',  
    status: 'Departing Location'  
  },  
  {  
    name: 'DEN – Denver, CO',  
    status: 'Connecting Flight'  
  },  
  {  
    name: 'SFO – San Francisco, CA',  
    status: 'Destination'  
  }  
]
```



# Transforming an array of objects into html

application.js


```
$( '.update-status' ).on( 'click', function() {  
    $.getJSON( '/status', function( result ) {  
        var statusElements = ???  
        $( '.status-list' ).html( statusElements )  
    });  
});
```



StatusElements

```
<li>  
    <h3>name</h3>  
    <p>status</p>  
</li>  
  
<li> ... </li>  
  
<li> ... </li>
```

*We need a good way to  
create an array of list  
item HTML elements  
from the result of the  
AJAX call.*



Result

```
[  
  {  
    name: 'JFK – New York, NY',  
    status: 'Departing Location'  
  },  
  {  
    name: 'DEN – Denver, CO',  
    status: 'Connecting Flight'  
  },  
  {  
    name: 'SFO – San Francisco, CA',  
    status: 'Destination'  
  }  
]
```

# An Array of Results with \$.map()

*You can think of a collection as an array*

```
$.map(collection, function(<item>, <index>){});
```

*Beware the different argument order!*

Map returns an array modified by what is returned in the function passed as an argument.

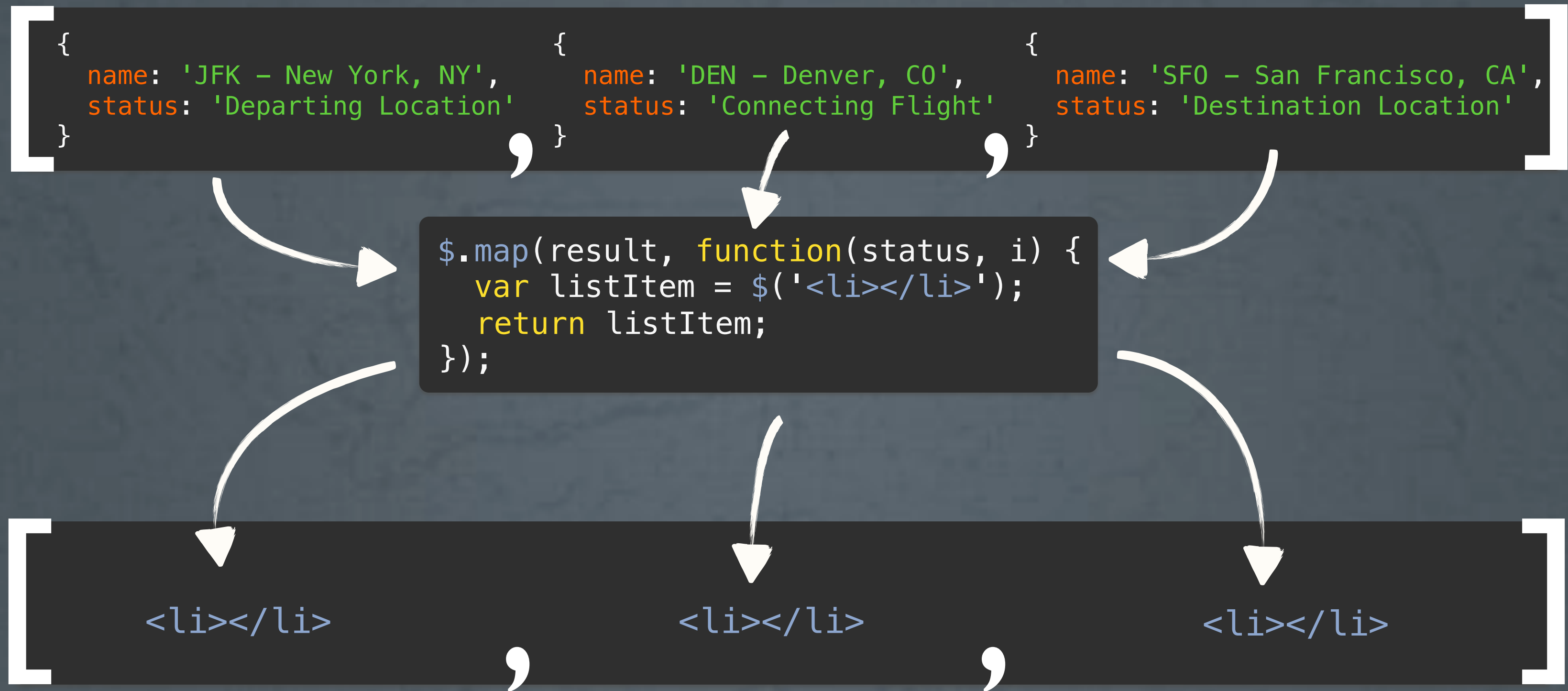
```
var myNumbers = [1,2,3,4];
```

```
var newNumbers = $.map(myNumbers, function(item, index){ return item + 1 });
```

myNumbers → [1,2,3,4]

newNumbers → [2,3,4,5]

# Transforming from JSON to HTML



# Transforming from JSON to HTML

```
[ { name: 'JFK - New York, NY', status: 'Departing Location' }, { name: 'DEN - Denver, CO', status: 'Connecting Flight' }, { name: 'SFO - San Francisco, CA', status: 'Destination Location' } ]
```

```
$.map(result, function(status, i) {  
  var listItem = $('<li></li>');  
  $('<h3>'+status.name+'</h3>').appendTo(listItem);  
  $('<p>'+status.status+'</p>').appendTo(listItem);  
  return listItem;  
});
```

```
[ <li>  
  <h3>JFK - New York, NY</h3>  
  <p>Departing Location</p>  
</li> , <li>  
  <h3>DEN - Denver, CO</h3>  
  <p>Connecting Flight</p>  
</li> , <li>  
  <h3>SFO - San Francisco, CA</h3>  
  <p>Destination Location</p>  
</li> ]
```

# Transforming from JSON to HTML

application.js

```
$( '.update-flight-status' ).on( 'click', function() {  
    $.getJSON( '/status', function( result ) {  
        var statusElements = $.map( result, function( status, i ) {  
            var listItem = $( '<li></li>' );  
            $( '<h3>' + status.name + '</h3>' ).appendTo( listItem );  
            $( '<p>' + status.status + '</p>' ).appendTo( listItem );  
            return listItem;  
        } );  
        $( '.status-list' ).html( statusElements );  
    } );  
} );
```



*Make sure to return the list item.*

*statusElements is an array list items*

**UPDATE FLIGHT**

JFK - New York, NY

Departing Location

SFO - San Francisco, CA

Destination Location



# `$.each` vs `$.map`

**So what's the difference  
between `$.each` and `$.map`?**




```
Developer Tools - http://localhost:4000/
Elements Resources Network Sources Timeline »

> var cities = ['Paris', 'London', 'Orlando'];
> $.each(cities, function(index, city) {
  var result = city + " " + index;
  console.log(result);
});
Paris 0
London 1
Orlando 2
< ["Paris", "London", "Orlando"]
> $.map(cities, function(city, index) {
  var result = city + " " + index;
  console.log(result);
  return result;
});
Paris 0
London 1
Orlando 2
< ["Paris 0", "London 1", "Orlando 2"]


<top frame>
```

# \$.each vs \$.map

*\$.each runs the function for each item in the array, but returns the original array unchanged.*



*\$.map runs the function for each item in the array and creates a new array from the returned results.*



# Give the DOM a break with .detach()

application.js

```
$( '.update-flight-status' ).on( 'click', function() {  
  $.getJSON( '/status', function( result ) {  
    var statusElements = $.map( result, function( status, index ) {  
      var listItem = $( '<li></li>' );  
      $( '<h3>' + status.name + '</h3>' ).appendTo( listItem );  
      $( '<p>' + status.status + '</p>' ).appendTo( listItem );  
      return listItem;  
    } );  
    $( '.status-list' ).html( statusElements );  
  } );  
});
```


.detach()

*.detach() removes an element from the DOM, preserving all data and events. This is useful to minimize DOM insertions with multiple html elements.*


# Give the DOM a break with .detach()

application.js

```
$( '.update-flight-status' ).on( 'click', function() {  
    $.getJSON( '/status', function( result ) {  
        var statusElements = $.map( result, function( status, index ) {  
            var listItem = $( '<li></li>' );  
            $( '<h3>' + status.name + '</h3>' ).appendTo( listItem );  
            $( '<p>' + status.status + '</p>' ).appendTo( listItem );  
            return listItem;  
        });  
        $( '.status-list' ).detach()  
        .html( statusElements )  
        .appendTo( '.status' );  
    });  
});
```



*.detach() removes the list from the DOM,  
then it can be modified and reinserted into  
the status element.*



UPDATE FLIGHT

JFK - New York, NY

Departing Location

SFO - San Francisco, CA

Destination Location