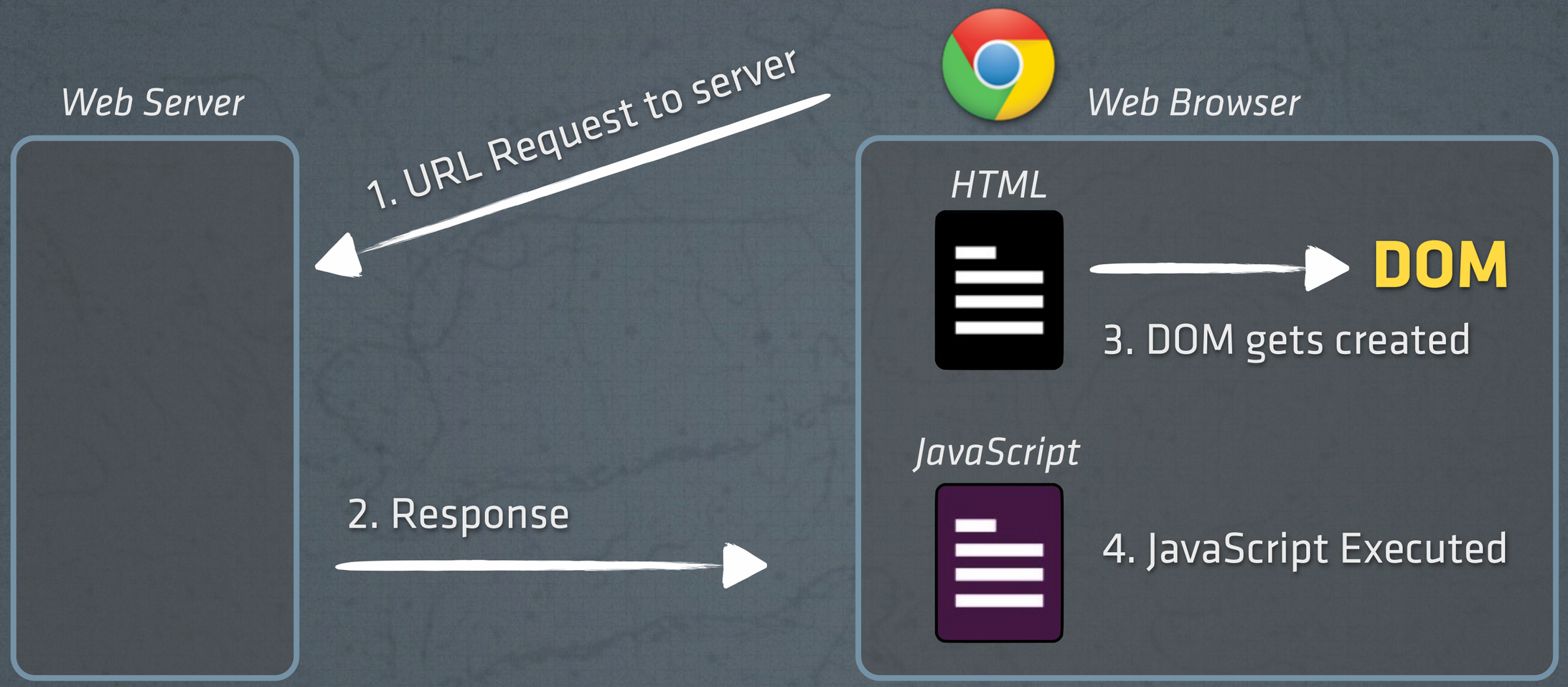
Ajax Basics

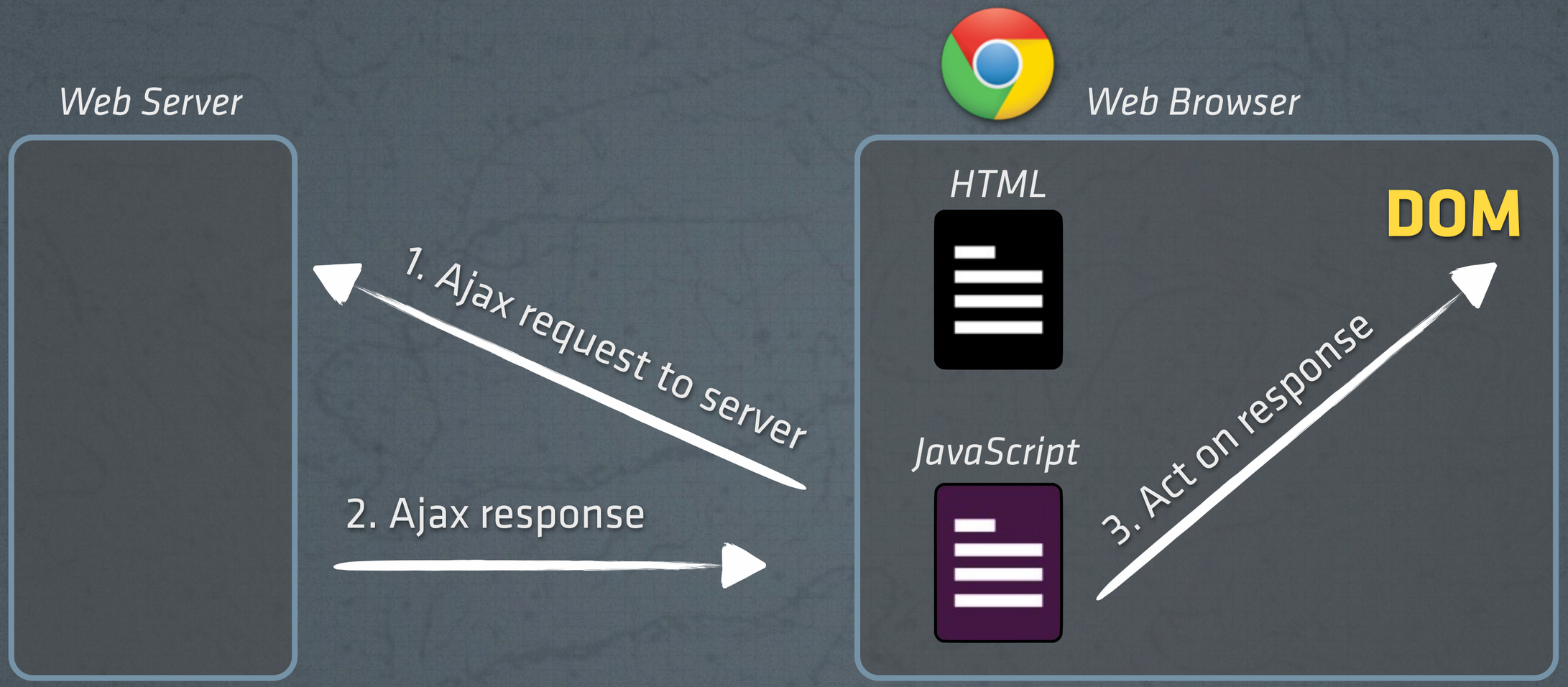
Typical Web Request



Asynchronous JavaScript And XML "Ajax"

A client side technique for communicating with a web server.

Ajax Communication



Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Hawaiian Vacation

Paid $399.99 on January 14, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

Previous Code

index.html

```
<li class='confirmation'>
  <h3>Hawaiian Vacation</h3>
  <button>Flight details</button>
  <div class='ticket'>
    <a class='view-boarding-pass'>View Boarding Pass</a>
    <img src='/ticket-14836.png'/>
  </div>
</li>
```

How would we load the ticket from the server when the button is clicked?

Shows the hidden element when the button is clicked

application.js

```
$('.confirmation').on('click', 'button', function(){
  $(this).closest('.confirmation').find('.ticket').slideDown();
});
$('.confirmation .view-boarding-pass').on('click', function() {
  $(this).closest('.ticket').find('img').show();
});
```

Show boarding pass when link is clicked

Making our first AJAX call

application.js

```
$('.confirmation').on('click', 'button', function(){
  $.ajax('http://example.org/confirmation.html', {
    success: function(response) {
      $('.ticket').html(response).slideDown();
    }
  });
});
```

Ajax Method

```
$.ajax(url[, settings])
```

Runs only when the server returns a successful response

The response which gets returned. Not a full webpage.

http://example.org/confirmation.html

```
<div> ...
  <strong>Boarding Pass: </strong>
  <a href='#' class='view-boarding-pass'>View Boarding Pass</a>
  <img src='ticket.png' alt='Your boarding pass' class='boarding-pass' />
</div>
```

Using a relative URL for AJAX calls

application.js

```
$('.confirmation').on('click', 'button', function(){
  $.ajax('confirmation.html', {
    success: function(response) {
      $('.ticket').html(response).slideDown();
    }
  });
});
```

Can be a relative URL

If you were on http://example.org/index.html, these requests would go to the same URL

```
$.ajax('http://example.org/confirmation.html', ...);
```

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Hawaiian Vacation

Paid $399.99 on January 14, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

Elements Resources Network Sources Timeline Profiles Audits Console

<top frame> <page context> All Errors Warnings Logs Debug

Shorthand with $.get

application.js

```
$.ajax('confirmation.html', {
  success: function(response) {
    $('.ticket').html(response).slideDown();
  }
});
```

Equivalent to $.ajax call above

```
$.get('confirmation.html', function(response) {
  $('.ticket').html(response).slideDown();
});
```

Shorthand Method

```
$.get(url, success)
```

Sending parameters with requests

Always fetching the same confirmation, no matter what trip

application.js

```
$.ajax('confirmation.html', {
  success: function(response) {
    $('.ticket').html(response).slideDown();
  }
});
```

How would we do a request with a Confirmation Number?

```
confirmation.html?confNum=1234
```

Sending parameters with requests

application.js

```
$.ajax('confirmation.html?confNum=1234', {
  success: function(response) {
    $('.ticket').html(response).slideDown();
  }
});
```

Equivalent to $.ajax call above

```
$.ajax('confirmation.html', {
  success: function(response) {
    $('.ticket').html(response).slideDown();
  },
  data: { "confNum": 1234 }
  }
});
```

JavaScript Object

Often the data in the request is dynamic

The flight number could be pulled from the HTML

```
<div class='ticket' data-confNum='1234'>
```

```
$.ajax('confirmation.html', {
  success: function(response) {
    $('.ticket').html(response).slideDown();
  },
  data: { "confNum": $(".ticket").data("confNum")
  }
});
```

Still results in the same URL request

```
confirmation.html?confNum=1234
```

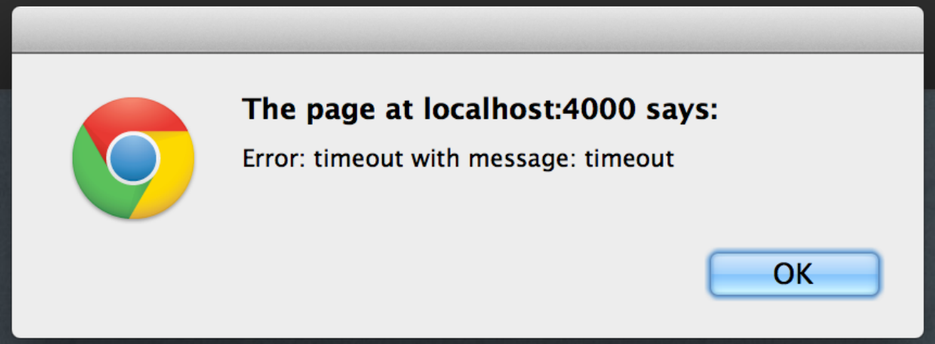
Ajax Options

Handling failed AJAX requests

application.js

```
$('.confirmation').on('click', 'button', function(){
  $.ajax('confirmation.html', {
    success: function(response) {
      $('.ticket').html(response).slideDown();
    },
    error: function(request, errorType, errorMessage) {
      alert('Error: ' + errorType + ' with message: ' + errorMessage);
    }
  });
});
```

Runs this callback if there is a timeout, abort, or server error



If the browser times out fetching this request, you might get this error

Setting the timeout speed

application.js

```
$('.confirmation').on('click', 'button', function(){
  $.ajax('confirmation.html', {
    success: function(response) { ... },
    error: function(request, errorType, errorMessage) { ... },
    timeout: 3000
  });
});
```

In ms, 1 second = 1000 ms

beforeSend and complete callbacks

application.js

```
$('.confirmation').on('click', 'button', function(){
  $.ajax('confirmation.html', {
    success: function(response) { ... },
    error: function(request, errorType, errorMessage) { ... },
    timeout: 3000,
    beforeSend: function() {
      $('.confirmation').addClass('is-loading');
    },
    complete: function() {
      $('.confirmation').removeClass('is-loading');
    }
  });
});
```

Runs before the Ajax request

Runs after both success and error

We can add a loading animation using CSS for this class

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Hawaiian Vacation

Paid $399.99 on January 14, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

Why didn't our normal click handler work?

http://example.org/confirmation.html

Our AJAX response

```
<div> ...
  <strong>Boarding Pass: </strong>
  <a href='#' class='view-boarding-pass'>View Boarding Pass</a>
  <img src='ticket.png' alt='Your boarding pass' class='boarding-pass' />
</div>
```

application.js

The Click handler that's not working

```
$('.confirmation .view-boarding-pass').on('click', function(){ ... });
```

This was run once when the page loaded.

But when the page loaded `.view-boarding-pass` *didn't exist.*

It only existed after our AJAX request.

Using Event Delegation

http://example.org/confirmation.html

Our AJAX response

```
<div> ...
  <strong>Boarding Pass: </strong>
  <a href='#' class='view-boarding-pass'>View Boarding Pass</a>
  <img src='ticket.png' alt='Your boarding pass' class='boarding-pass' />
</div>
```

application.js

```
$('.confirmation .view-boarding-pass').on('click', function(){ ... });
```

```
$('.confirmation').on('click', '.view-boarding-pass', function(){ ... });
```

Listen for click events inside `.confirmation`

When they happen, check if the target was `.view-boarding-pass`

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Hawaiian Vacation

Paid $399.99 on January 14, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

JavaScript Objects

application.js

```
$(document).ready(function() {
  $('.confirmation').on('click', 'button', function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      success: function(response) { $('.ticket').html(response).slideDown(); },
      error: function(request, errorType, errorMessage) {
        alert('Error: ' + errorType + ' with message: ' + errorMessage);
      },
      beforeSend: function() { $('.confirmation').addClass('is-loading'); },
      complete: function() { $('.confirmation').removeClass('is-loading'); }
    });
  });
  $('.confirmation').on('click', '.view-boarding-pass', function(event) {
    event.preventDefault();
    $('.view-boarding-pass').hide();
    $('.boarding-pass').show();
  });
});
```

that's a big document ready method!

How can we organize this?

Refactoring to Objects

application.js

```
var confirmation = {
  init: function() {
    // Our existing event handlers
  }
};
```

Within init, we can run all the code that was in our document ready function

```
$(document).ready(function() {
  confirmation.init();
});
```

Much easier to read what happens on load

Refactoring to Objects

application.js

```
var confirmation = {
  init: function() {
    $('.confirmation').on('click', 'button', function() {
      $.ajax('confirmation.html', { ... });
    });

    $('.confirmation').on('click', '.view-boarding-pass', function(event) { ... });
  }
};
$(document).ready(function() {
  confirmation.init();
});
```

Our init method creates our event handlers

Kick things off on page load

Separating Functions

We can pull this function out of init to make this easier to read

application.js

```
var confirmation = {
  init: function() {
    $('.confirmation').on('click', 'button', function() {
      $.ajax('confirmation.html', { ... });
    });

    $('.confirmation').on('click', '.view-boarding-pass', function(event) { ... });
  }
};
$(document).ready(function() {
  confirmation.init();
});
```

Separating Functions

Need to use 'this' to reference other keys of this object

application.js

```
var confirmation = {
  init: function() {
    $('.confirmation').on('click', 'button', this.loadConfirmation);
    $('.confirmation').on('click', '.view-boarding-pass',  function(event) { ... });
  },
  loadConfirmation: function() {
    $.ajax('confirmation.html', { ... });
  }
};

$(document).ready(function() {
  confirmation.init();
});
```

Refactoring our other event handler

application.js

```
var confirmation = {
  init: function() {
    $('.confirmation').on('click', 'button', this.loadConfirmation);
    $('.confirmation').on('click', '.view-boarding-pass', this.showBoardingPass);
  },
  loadConfirmation: function() {
    $.ajax('confirmation.html', { ... });
  },
  showBoardingPass: function(event) { ... }
};

$(document).ready(function() {
  confirmation.init();
});
```

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Hawaiian Vacation

Paid $399.99 on January 14, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

JavaScript Functions

Object vs Function

```
var vacation = {
  init: function() {
    // init vacation
  }
};

$(document).ready(function() {
  vacation.init();
});
```

Limited to 1 confirmation on the page

Notice it's capitalized!

```
function Vacation(destination) {
  // init vacation to destination
}

$(document).ready(function() {
  var paris = new Vacation('Paris');
  var london = new Vacation('London');
});
```

Can have multiple vacations

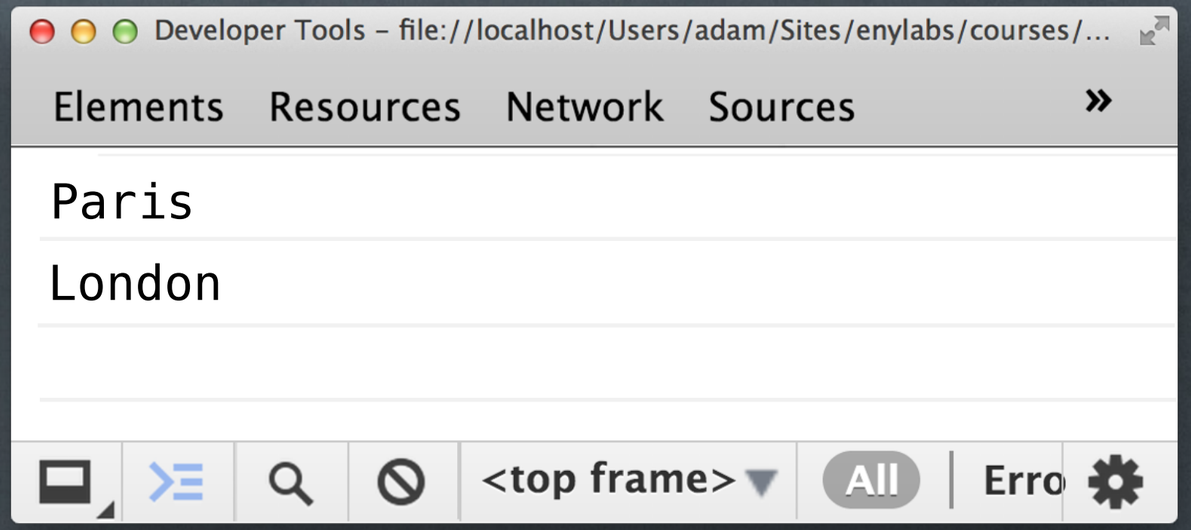
Functions as Objects Example

```
function Vacation(destination) {
  this.details = function() {
    console.log(destination);
  }
}

$(document).ready(function() {
  var paris = new Vacation('Paris');
  paris.details();

  var london = new Vacation('London');
  london.details();
});
```

paris and london both have their own scoped variables

Browser Logging

```
console.log(message);
```



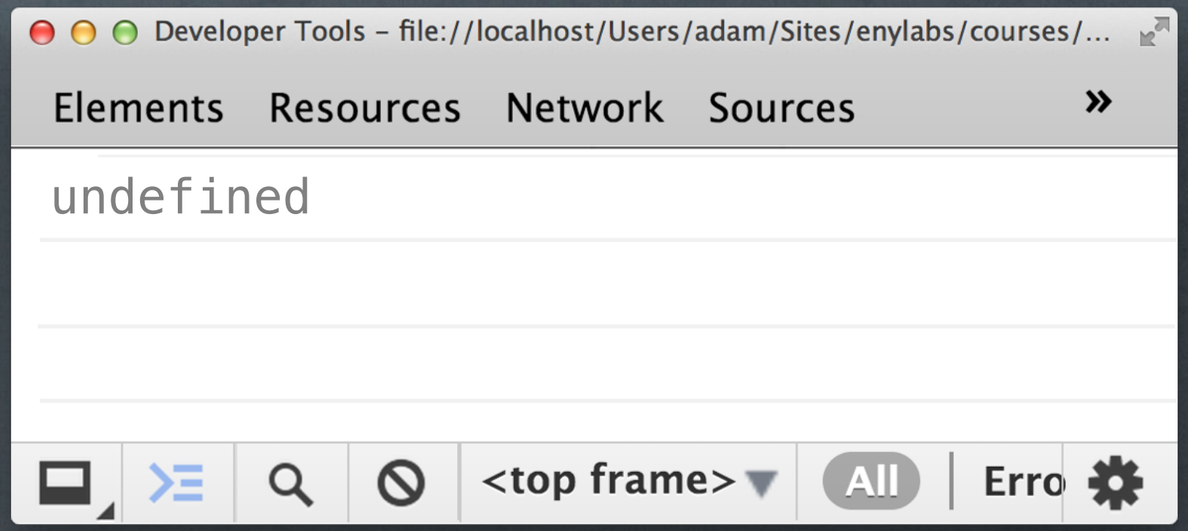
Functions as Objects Example

```
function Vacation(destination) {
  this.details = function() {
    console.log(destination);
  }
}

$(document).ready(function() {
  var paris = new Vacation('Paris');
  console.log(paris.destination);
});
```

Can't access locally scoped variables

Browser Logging

```
console.log(message);
```



Lets Move this Object into a Reusable Function

application.js

```
var confirmation = {
  init: function() {
    $('.confirmation').on('click', 'button', this.loadConfirmation);
    $('.confirmation').on('click', '.view-boarding-pass', this.showBoardingPass);
  },
  loadConfirmation: function() { ... }
  showBoardingPass: function(event) { ... }
};

$(document).ready(function() {
  confirmation.init();
});
```

Refactoring to Functions

application.js

el refers to the element

```
function Confirmation(el) {
  this.el = el;
  // helper methods go here
  // event handlers go here
}


$(document).ready(function() {
  var paris = new Confirmation($('#paris'));
  var london = new Confirmation($('#london'));
});
```

Save a reference to the passed in element

Create a new confirmation for each ticket

Adding the methods and event listeners

application.js

```
function Confirmation(el) {
  this.el = el;

  this.loadConfirmation = function() { ... }
  this.showBoardingPass = function(event) { ... }

  this.el.on('click', 'button', this.loadConfirmation);
  this.el.on('click', '.view-boarding-pass', this.showBoardingPass);
}

$(document).ready(function() {
  var paris = new Confirmation($('#paris'));
  var london = new Confirmation($('#london'));
});
```

Fixing the target element

application.js

```
function Confirmation(el) {
  this.el = el;

  this.loadConfirmation = function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      success: function(response) {
        $('.ticket').html(response).slideDown();
      }
    });
  }

  this.el.on('click', 'button', this.loadConfirmation);
  this.el.on('click', '.view-boarding-pass', this.showBoardingPass);
}
```

Will target all tickets, not just the one that was clicked

Fixing the target element

application.js

```
function Confirmation(el) {
  this.el = el;
  this.ticket = this.el.find('.ticket');

  this.loadConfirmation = function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      success: function(response) {
        this.ticket.html(response).slideDown();
      }
    });
  }

  this.el.on('click', 'button', this.loadConfirmation);
  this.el.on('click', '.view-boarding-pass', this.showBoardingPass);
}
```

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Paris Vacation

Paid $1399.99 on January 14, 2013.

FLIGHT DETAILS

London Vacation

Paid $1199.99 on December 28, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

We're referencing the wrong 'this'

application.js

```
function Confirmation(el) {
  this.el = el;
  this.ticket = this.el.find('.ticket');

  this.loadConfirmation = function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      success: function(response) {
        this.ticket.html(response).slideDown();
      }
    });
  }
}
```

This refers to ajax settings and not our function's el

Sometimes jQuery changes the value of 'this'

Inside AJAX callbacks, this is set to the AJAX settings.

Setting the proper 'this'

application.js

```
function Confirmation(el) {
  this.el = el;
  this.ticket = this.el.find('.ticket');
  var confirmation = this;

  this.loadConfirmation = function() {
    $.ajax('confirmation.html', {
      timeout: 3000,
      context: confirmation,
      success: function(response) {
        this.ticket.html(response).slideDown();
      }
    });
  }
}
```

Allows us to set the value of "this" inside our callbacks.

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Paris Vacation

Paid $1399.99 on January 14, 2013.

FLIGHT DETAILS

London Vacation

Paid $1199.99 on December 28, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

Refactoring hardcoded DOM Elements

application.js

Run when the link is clicked

```
this.showBoardingPass = function(event) {
  event.preventDefault();
  $('.view-boarding-pass').hide();
  $('.boarding-pass').show();

  console.log(this);
  console.log(confirmation);
}
```

In the context of an event callback 'this' is set to the dom object

References the confirmation object in the parent scope

Developer Tools – http://localhost:4000/

Elements Resources Network Sources Timeline »

```
<a href="#" class="view-boarding-pass" style="display: none;">View Boarding Pass</a>

Confirmation {loadConfirmation: function, showBoardingPass: function, el:b.fn.b.init[1], ticket: b.fn.b.init[1]}
```

<top frame>

Refactoring to Functions

application.js

```
this.showBoardingPass = function(event) {
  event.preventDefault();
  $(this).hide();
  confirmation.el.find('.boarding-pass').show();
}
```

Remember, this will be the link that was clicked

Able to use variables from the parent scope

Rendered HTML:

jQuery Travels - Vacation Confirmation

Confirmations

Paris Vacation

Paid $1399.99 on January 14, 2013.

FLIGHT DETAILS

London Vacation

Paid $1199.99 on December 28, 2013.

FLIGHT DETAILS

Call us at 555-25937 to make a reservation today!

Ajax Forms

$.ajax with POST

jQuery Travels - Trip Booker

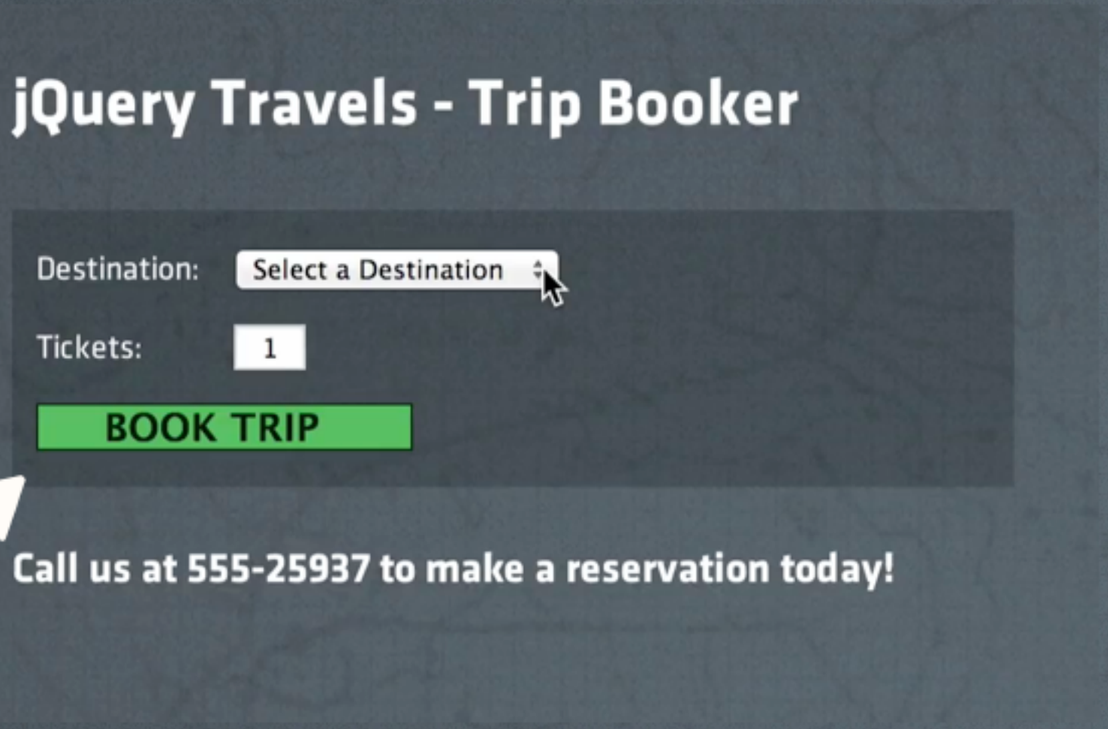
Destination: Select a Destination

Tickets: 1

BOOK TRIP

Call us at 555-25937 to make a reservation today!

$.ajax with POST



What if we want to submit this form via ajax?

application.js

```
$('form').on('submit', function(e) {
  $.ajax('/book', {
    type: 'POST'
  });
});
```

Will do a POST request to the server, which is used for forms

jQuery Travels - Trip Booker

Destination: Select a Destination

Tickets: 1

BOOK TRIP

Call us at 555-25937 to make a reservation today!

Preventing Double Submission

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST'
  });
});
```

Prevents browser from submitting

This would do the AJAX request, but it's not sending any form data over.

index.html

```
<form action='/book'>
  <select id='destination' name='destination'>
    ...
  </select>
  <input type='text' id='quantity' name='quantity' value='1' />
</form>
```

How would we pass the values of this form via $.ajax?

We could use the data option and grab all the values

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST',
    data: { "destination": $('#destination').val(),
            "quantity": $('#quantity').val() }
  });
});
```

Works, but there's a better way

index.html

```
<form action='/book'>
  <select id='destination' name='destination'>
    ...
  </select>
  <input type='text' id='quantity' name='quantity' value='1' />
</form>
```

Serialize packages all the values for us

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST',
    data: $('form').serialize()
  });
});
```

jQuery Object Methods

```
serialize()
```

Used to merge all form fields for submission

index.html

```
<form action='/book'>
  <select id='destination' name='destination'>
    ...
  </select>
  <input type='text' id='quantity' name='quantity' value='1' />
</form>
```

Add AJAX callback (just like you saw in level 1)

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST',
    data: $('form').serialize(),
    success: function(result) {  }
  });
});
```

Like our previous $.ajax request, success is called with the HTML result

result

Your vacation to Paris, France has been booked for $2,196.00 for 4 nights. Confirmation #345feab.

$.ajax success callback

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST',
    data: $('form').serialize(),
    success: function(result) {
      $('form').remove();
      $('#vacation').hide().html(result).fadeIn();
    }
  });
});
```

There's duplication of DOM lookups

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  $.ajax('/book', {
    type: 'POST',
    data: $('form').serialize(),
    success: function(result) {
      $('form').remove();
      $('#vacation').hide().html(result).fadeIn();
    }
  });
});
```

Eliminating duplication of DOM lookups

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  var form = $(this);
  $.ajax('/book', {
    type: 'POST',
    data: form.serialize(),
    success: function(result) {
      form.remove();
      $('#vacation').hide().html(result).fadeIn();
    }
  });
});
```

Save a reference to the form to reduce DOM queries

jQuery Travels - Trip Booker

Destination: Select a Destination

Tickets: 1

BOOK TRIP

Call us at 555-25937 to make a reservation today!

Ajax with JSON

JavaScript Object Notation

A standard way to format data

"HTML"

```
<p>Your vacation to Paris,
France has been booked for
$2,196.00 for 4 nights.
Confirmation #345feab.</p>
```

Server decides how the HTML is formatted

"JSON"

```
{
  totalPrice: 2196.00,
  nights: 4,
  location: 'Paris, France',
  confirmation: '345feab'
}
```

Server returns details, we can use it as we wish

Our old code wouldn't work

application.js

```
$('form').on('submit', function(event) {
  event.preventDefault();
  var form = $(this);
  $.ajax('/book', {
    type: 'POST',
    data: form.serialize(),
    success: function(result) {
      form.remove();
      $('#vacation').hide().html(result).fadeIn();
    }
  });
});
```

This code is expecting HTML to get returned, and injecting the result right into the page.

$.ajax with POST and JSON

```
var form = $(this);
$.ajax('/book', {
  type: 'POST',
  data: form.serialize(),
  dataType: 'json',
  success: function(result) {
    form.remove();
    $('#vacation').hide().html(result).fadeIn();
  },
  contentType: 'application/json'
});
```

Parse the response as JSON

Result isn't HTML, we'll need to create a DOM node here

Ask the server to respond with JSON

Reading the JSON results and creating HTML

```
success: function(result) {
  form.remove();
  var msg = $("<p></p>");
  msg.append("Destination: " + result.location + ". ");
  msg.append("Price: " + result.totalPrice + ". ");
  msg.append("Nights: " + result.nights + ". ");
  msg.append("Confirmation: " + result.confirmation+ ".");

  $('#vacation').hide().html(msg).fadeIn();
}
```

Create and add a DOM node

result *JSON is converted into a JavaScript Object*

```
{
  location: 'Paris',
  totalPrice: 1196.00,
  nights: 4,
  confirmation: '345feab'
}
```

Destination: Paris. Price: $2196.00. Nights: 4. Confirmation: 345feab.

Final event handler with POST and JSON

```
$('form').on('submit', function(e) {
  event.preventDefault();
  var form = $(this);
  $.ajax('/book', {
    type: 'POST',
    contentType: 'application/json',
    dataType: 'json',
    data: form.serialize(),
    success: function(result) {
      form.remove();
      var msg = $("<p></p>");
      msg.append("Destination: " + result.location + ". ");
      msg.append("Price: " + result.totalPrice + ". ");
      msg.append("Nights: " + result.nights + ". ");
      msg.append("Confirmation: " + result.confirmation+ ".");
      $('#vacation').hide().html(msg).fadeIn();
    },
    contentType: 'application/json'
  });
});
```

Don't Repeat Yourself

application.js

```
$.ajax('/book', { ... });
```

index.html

```
<form action='/book' method='POST'>
  ...
</form>
```

Form URL is in both HTML and JavaScript

application.js

```
$.ajax($('form').attr('action'),{ ... });
```

Will use the action from the form HTML

jQuery Object Methods

```
attr(<attribute>)
```

```
attr(<attribute>, <value>)
```

Final event handler with POST and JSON

```
$('form').on('submit', function(e) {
  event.preventDefault();
  var form = $(this);
  $.ajax($('form').attr('action'),{
    type: 'POST',
    contentType: 'application/json',
    dataType: 'json',
    data: form.serialize(),
    success: function(result) {
      form.remove();
      var msg = $("<p></p>");
      msg.append("Destination: " + result.location + ". ");
      msg.append("Price: " + result.totalPrice + ". ");
      msg.append("Nights: " + result.nights + ". ");
      msg.append("Confirmation: " + result.confirmation+ ".");
      $('#vacation').hide().html(msg).fadeIn();
    }
  });
});
```

jQuery Travels - Trip Booker

Destination: Select a Destination

Tickets: 1

BOOK TRIP

Call us at 555-25937 to make a reservation today!

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)
  });
});
```

Result

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

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.

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

```
[
  {
    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>');
  return listItem;
});
```

```
[
  <li></li>,
  <li></li>,
  <li></li>
]
```

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?

$.each vs $.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 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

Advanced Events

Managing Events

Both functions will be run when the button is clicked

```
function picture() { console.log('Show Plane'); }
function status() { console.log('In Service'); }

$(document).ready(function() {
  $('button').on('click', picture);
  $('button').on('click', status);
});
```

```
<button>Boeing 777</button>
```

```
click:picture
```

```
click:status
```

But what if later we wanted to turn off these click event handlers?

Removing event handlers

```
function picture() { console.log('Show Plane'); }
function status() { console.log('In Service'); }

$(document).ready(function() {
  $('button').on('click', picture);
  $('button').on('click', status);
  ...
  $('button').off('click');
});
```

Removes all of the callbacks

```
<button>Boeing 777</button>
```

~~click:picture~~

~~click:status~~

jQuery Object Methods

`off(<event name>)` *Stops watching for the given event*

But what if we only want to turn off the status handler?

Namespacing Events

```
function picture() { console.log('Show Plane'); }
function status() { console.log('In Service'); }

$(document).ready(function() {
  $('button').on('click.image', picture);
  $('button').on('click.details', status);
  ...
  $('button').off('click.image');
});
```

```
<button>Boeing 777</button>
```

~~click.image:picture~~

click.details:status

Only removes event handlers called click.image

Would do the same thing

```
  $('button').off('.image');
```

Removes all types of event handlers that end in .image

Removing Namespaced Events

```
function picture() { console.log('Show Plane'); }
function status() { console.log('In Service'); }
function zoom() { console.log('Zoom Picture'); }

$(document).ready(function() {
  $('button').on('click.image', picture);
  $('button').on('click.details', status);
  $('button').on('mouseover.image', zoom);

  ...
  $('button').off('.image');
});
```

Removes all types of event handlers that end in .image

```
<button>Boeing 777</button>
```

~~click.image:picture~~

click.details:status

~~mouseover.image:zoom~~

Triggering Events

```
function picture() { console.log('Show Plane'); }
function status() { console.log('In Service'); }
function zoom() { console.log('Zoom Picture'); }

$(document).ready(function() {
  $('button').on('click.image', picture);
  $('button').on('click.details', status);
  $('button').on('mouseover.image', zoom);
  ...
  $('button').trigger('click');
});
```

Similar to if the user clicked on the button

jQuery Object Methods

`trigger(<event name>)`

Will trigger the given event on the target

```
click.image:picture
```

```
click.details:status
```

```
mouseover.image:zoom
```

We can also trigger individual namespaced events

```
$('button').trigger('click.details');
```

Triggers the status function

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Call us at 555-25937 to make a reservation today!

Implementing our vacation pricing

```
<li class='vacation' data-price='399'>
  <h3>Hawaiian Vacation</h3>
  <button>Show Price</button>
</li>

<li class='vacation' data-price='749'>
  <h3>European Vacation</h3>
  <button>Show Price</button>
</li>

<a href='#' class='show-prices'>Show All Prices</a>
```

Need Event Handlers

First lets define the handler on the Show Price button

```
$('.vacation').on('click.price', 'button', showPrice)
```

We are namespacing the click event, since it is part of the price responsibility

Also using the proper delegation syntax, not

```
$('.vacation button').on(...
```

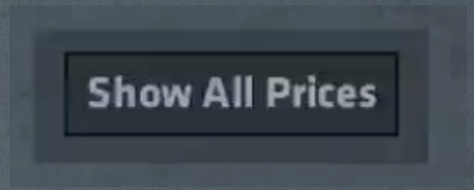
Implementing our showPrice method

```
<li class='vacation' data-price='399'>
  <h3>Hawaiian Vacation</h3>
  <button>Show Price</button>
</li>
```

```
var showPrice = function() {
   var vacation = $(this).closest('.vacation');
   var price = vacation.data('price');
   var details = $('<p>Book 3 days for $'+(3 * price)+'</p>');
   vacation.append(details);
};
$('.vacation').on('click.price', 'button', showPrice)
```

How do we implement show all prizes?

```
<li class='vacation' data-price='399'>
  <h3>Hawaiian Vacation</h3>
  <button>Show Price</button>
</li>

<a href='#' class='show-prices'>Show All Prices</a>
```



We need to call showPrice on each of the vacation list items

```
var showPrice = function() {
   var vacation = $(this).closest('.vacation');
   var price = vacation.data('price');
   var details = $('<p>Book 3 days for $'+(3 * price)+'</p>');
   vacation.append(details);
};
```

We could create a custom event

```
<a href='#' class='show-prices'>Show All Prices</a>
```

Show All Prices

Our typical event handler syntax

```
$(<dom element>).on("<event>.<namespace>", <method>)
```

The <event> can also be custom (instead of click, hover, etc)

```
$('.vacation').on('show.price', showPrice);
```

To trigger this event on all vacations

```
$('.vacation').trigger('show.price');
```

To trigger this event on a single vacation

```
$('.vacation:last').trigger('show.price');
```

Our completed jQuery with custom events

```
<a href='#' class='show-prices'>Show All Prices</a>
```

```
var showPrice = function() { ... };

$('.vacation').on('click.price', 'button', showPrice)
$('.vacation').on('show.price', showPrice);

$('.show-prices').on('click', function(event) {
    event.preventDefault();
    $('.vacation').trigger('show.price');
};
```

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

jQuery Plugins

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

Not Interested

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Setting up a Plugin - HTML

index.html

```
<li class='vacation' data-price='399'>
  <h3>Hawaiian Vacation</h3>
  <button>Show Price</button>
  <a href='#' class='not-interested'>Not Interested</a>
</li>
```

When clicked, we'll add a <p> with the price

Clicking "Not Interested" will hide the vacation

Setting up a Plugin

application.js

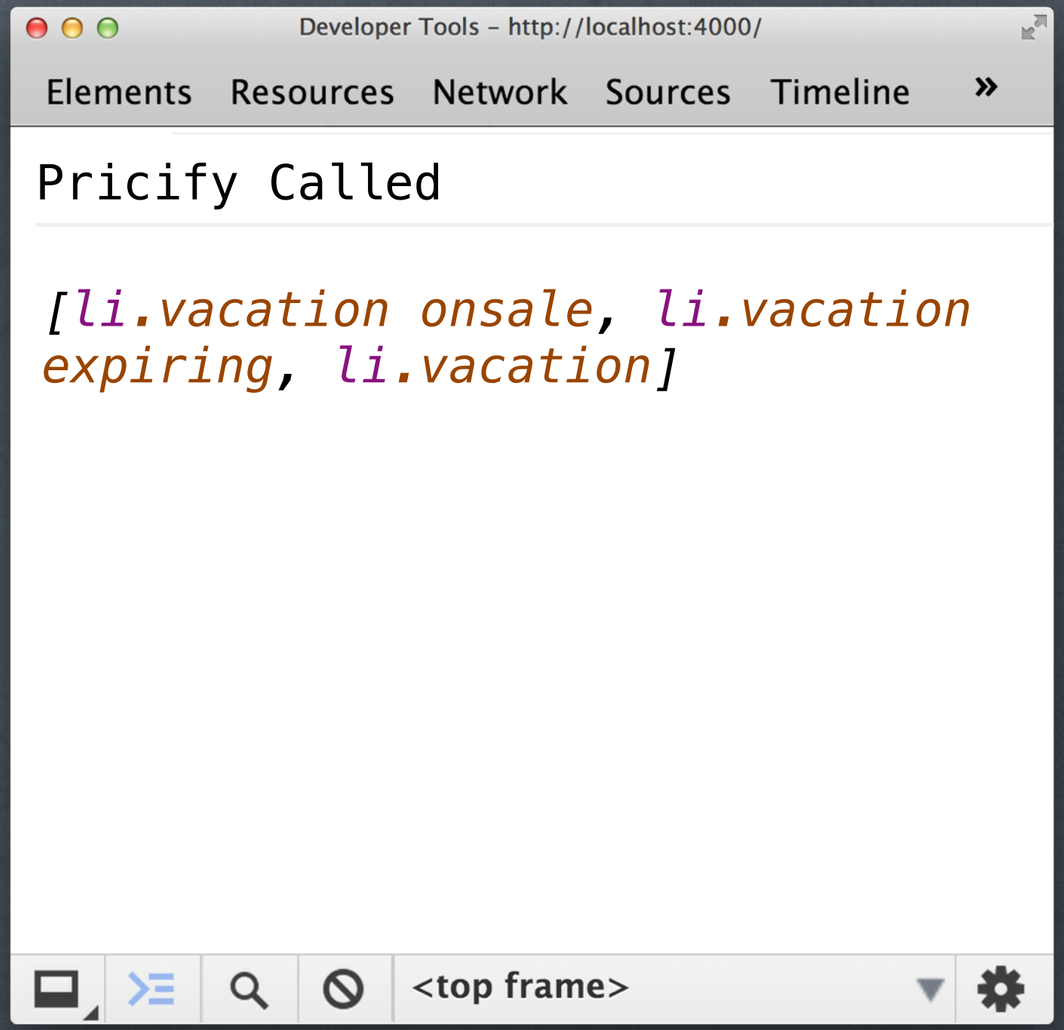
Makes the priceify method available

```
$.fn.priceify = function() {
  console.log('Pricify Called');
  console.log(this);
};
```

Within the plugin, 'this' will be the jQuery object the plugin was called on

```
$('.vacation').priceify();
```

We can 'priceify' all vacations at once



Adding a click handler to append price details

application.js

```
$.fn.priceify = function() {
  var vacation = this;
  vacation.on('click.priceify', 'button', function() {
    var price = vacation.data('price');
    var details = $('<p>Book 3 days for $'+(3 * price)+'</p>');
    $(this).hide();
    vacation.append(details);
  });
};

$(document).ready(function() {
  $('.vacation').priceify();
});
```

'this' will be $('.vacation')

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

Not Interested

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Fixing the vacation plugin

application.js

'this' will point to all .vacation elements

```
$.fn.priceify = function() {
  var vacation = this;
  vacation.on('click.priceify', 'button', function() {
    var price = vacation.data('price');
    var details = $('<p>Book 3 days for $'+(3 * price)+'</p>');
    $(this).hide();
    vacation.append(details);
  });
};

$(document).ready(function() {
  $('.vacation').priceify();
});
```

When we call append, it'll append it in multiple places!

Adding Iteration

application.js

Loop over each .vacation element setting each one to 'this'

```
$.fn.priceify = function() {
  this.each(function() {
    var vacation = this;
    vacation.on('click.priceify', 'button', function() {
      var price = vacation.data('price');
      var details = $('<p>Book 3 days for $'+(3 * price)+'</p>');
      $(this).hide();
      vacation.append(details);
    });
  });
};
```

Each vacation has its own click handler and local variable scope

jQuery Object Method

```
each(<function>)
```

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

Not Interested

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Plugins with Parameters

Plugin Use With Options

```
$('.vacation').priceify({ days: 5 });
```

Let's make our plugin accept an argument

```
$.fn.priceify = function(options) {
  ...
  var price = vacation.data('price');
  var details = $('<p>Book ' + options.days + ' days for $' +
                  (options.days * price) + '</p>');
```

But what if we wanted to make it an optional parameter?

Default it to 3 days if not passed in

Plugins with Options

Plugin Use With Options

```
$('.vacation').priceify({ days: 5 });
```

Plugin Use Without Options

```
$('.vacation').priceify();
```

Let's make our plugin accept an optional argument

```
$.fn.priceify = function(options) { ... };
```

Using $.extend

Utility Method

```
$.extend(target[, object1][, objectn])
```

Will combine all objects

```
$.extend({ days: 3 }, { price: 5 });
```

→

```
{ days: 3, price: 5 }
```

```
$.extend({ days: 3 }, {});
```

→

```
{ days: 3 }
```

```
$.extend({ days: 3 }, { days: 5 });
```

→

```
{ days: 5 }
```

That's how we can set a default value

Using $.extend in our plugin

Sets the default days to 3

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend({ days: 3 }, options);

    ...
    var vacation = this;
    var price = vacation.data('price');
```

Why not include the rest of these setting variables inside the settings object?

$.extend

application.js

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend({
      days: 3,
      vacation: this,
      price: $(this).data('price')
    }, options);
    vacation.on('click.priceify', 'button', function() { ... });
  });
};
```

Our event handler will need to be updated too

`settings.vacation` `settings.days` `settings.price`

$.extend

application.js

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend({
      days: 3,
      vacation: this,
      price: $(this).data('price')
    }, options);

    settings.vacation.on('click.priceify', 'button', function() {
      var details = $('<p>Book '+ settings.days +' days for $'+(settings.days *
settings.price +'</p>');
      $(this).hide();
      settings.vacation.append(details);
    });
  });
};
```

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

Not Interested

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Calling a Plugin from Outside

index.html

We could trigger('click.priceify') on all buttons

```
<a href='#' class='show-prices'>Show All Prices</a>
```

application.js

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend(...);
    settings.vacation.on('click.priceify', 'button', function() {
      var details = $('<p>Book '+ settings.days +' days for $'+
        (settings.days * settings.price)+'</p>');
      $(this).hide();
      settings.vacation.append(details);
    });
  });
};
```

... but we should create a new event and call this same function

Splitting out the show function

index.html

```
<a href='#' class='show-prices'>Show All Prices</a>
```

application.js

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend(...);
    var show = function() {
      var details = $('<p>Book '+ settings.days +' days for $'+
        (settings.days * settings.price)+'</p>');
      $(this).hide();
      settings.vacation.append(details);
    };
    settings.vacation.on('click.priceify', 'button', show);
  });
};
```

Split out the show function

Calling a Plugin from Outside

index.html

```
<a href='#' class='show-prices'>Show All Prices</a>
```

application.js

```
$.fn.priceify = function(options) {
  this.each(function() {
    var settings = $.extend(...);
    var show = function() {...};
    settings.vacation.on('click.priceify', 'button', show);
    settings.vacation.on('show.priceify', show);
  });
};
```

We can use the same show method

Custom Event

```
$('.show-prices').on('click', function(event) {
  event.preventDefault();
  $('.vacation').trigger('show.priceify');
});
```

Trigger an event on the vacation that the plugin watches for

TRY jQuery

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Removing behavior of a plugin

index.html

```
<li class="vacation" data-price='399'>
  ...
  <a href='#' class='remove-vacation'>Not Interested</a>
</li>
```

application.js

```
$.fn.priceify = function(options) {
  ...
  var remove = function() {
    settings.vacation.hide().off('.priceify');
  };
  settings.vacation.on('click.priceify', '.remove-vacation', remove);
};
```

*Removes all *.priceify event handlers*

jQuery Travels - Vacation Packages

Packages

Show All Prices

Hawaiian Vacation

SHOW PRICE

European Getaway

SHOW PRICE

Visit Japan

SHOW PRICE

Not Interested

Promises

Favorite

Paris, France

SHOW WEATHER

Our current code, not using a promise

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  $.ajax('/weather', {
    data: {q: location},
    success: function(result) {
      $('.weather').text(result.weather);
    }
  });
});
```

result will be a JSON object

result

```
{
  weather_code: 462,
  weather: "Partly Cloudy"
}
```

favorites.html

```
<div class='favorite'>
  <h3>Favorite</h3>
  <img src='/images/paris.png'/>
  <p class='loc'>Paris, France</p>
  <p class='weather'></p>
  <button>Show Weather</button>
</div>
```

What if we wanted to use this same ajax call on other pages?

Starting to build a promise object

application.js

```
var Weather = {
  today: function(location){
    var promise = $.ajax('/weather', {
      data: {q: location}
    });

    return promise;
  }
}
```

$.ajax() returns a promise object

Our current code

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  $.ajax('/weather', {
    data: {q: location},
    success: function(result) {
      $('.weather').text(result.weather);
    }
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.ajax('/weather', {
      data: {q: location}
    });

    return promise;
  }
}
```

Using the promise in our code

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.???(function(result) {
    $('.weather').text(result.weather);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.ajax('/weather', {
      data: {q: location}
    });

    return promise;
  }
}
```

Similar to AJAX success callback

```
promise.done(function(result){ ... });
```

Using the promise in our code

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result.weather);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.ajax('/weather', {
      data: {q: location}
    });

    return promise;
  }
}
```

On Ajax success, done will be called on the promise

Favorite

Paris, France

SHOW WEATHER

Using the promise in our code

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result.weather);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.ajax(
      '/weather', {
        data: {q: location}
      }
    );

    return promise;
  }
}
```

This code is a little brittle, because we need to know about result.weather, how can we hide this?

Our optimal .done method

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.ajax(
      '/weather', {
        data: {q: location}
      }
    );

    return promise;
  }
}
```

We need a way to intercept the result, inside our promise, and modify it before the .done method gets called.

Our optimal .done method

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.Deferred();
    $.ajax('/weather', {
      data: {q: location}
    });

    return promise;
  }
}
```

```
var promise = $.Deferred();
```

Create a new promise object. We'll need to tell it how to resolve

Our optimal .done method

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.Deferred();
    $.ajax('/weather', {
      data: {q: location},
      success: function(result) {
        promise.???(result.weather);
      }
    });

    return promise;
  }
}
```

Trigger our .done method and pass result.weather

```
promise.resolve(value);
```

Calls the done callback

```
promise.done(function(value){});
```

Using .resolve to trigger the .done callback

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise = Weather.today(location);
  todayPromise.done(function(result)
    $('.weather').text(result);
  });
});
```

```
var Weather = {
  today: function(location){
    var promise = $.Deferred();
    $.ajax('/weather', {
      data: {q: location}
      success: function(result) {
        promise.resolve(result.weather);
      }

    });

    return promise;
  }
}
```

Calling resolve will trigger our done callback

Custom promise with $.Deferred

```
var promise = $.Deferred();
```

jQuery method for creating a promise object

```
promise.resolve(value);
```

Calls the done callback

```
promise.done(function(value){});
```

```
promise.reject(value);
```

Calls the fail callback

```
promise.fail(function(value){});
```

Implementing the error callback

application.js

```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise =
    Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result);
  });
});
```

```
today: function(location){
  var promise = $.Deferred();
  $.ajax('/weather', {
    data: {q: location},
    success: function(result){
      promise.resolve(result.weather);
    },
    error: function(){
      var error = 'invalid location';
      promise.reject(error);
    }
  });

  return promise;
}
```

```
promise.reject(value);
```

Calls the fail callback

```
promise.fail(function(value){});
```

.reject triggers the .fail callback

application.js

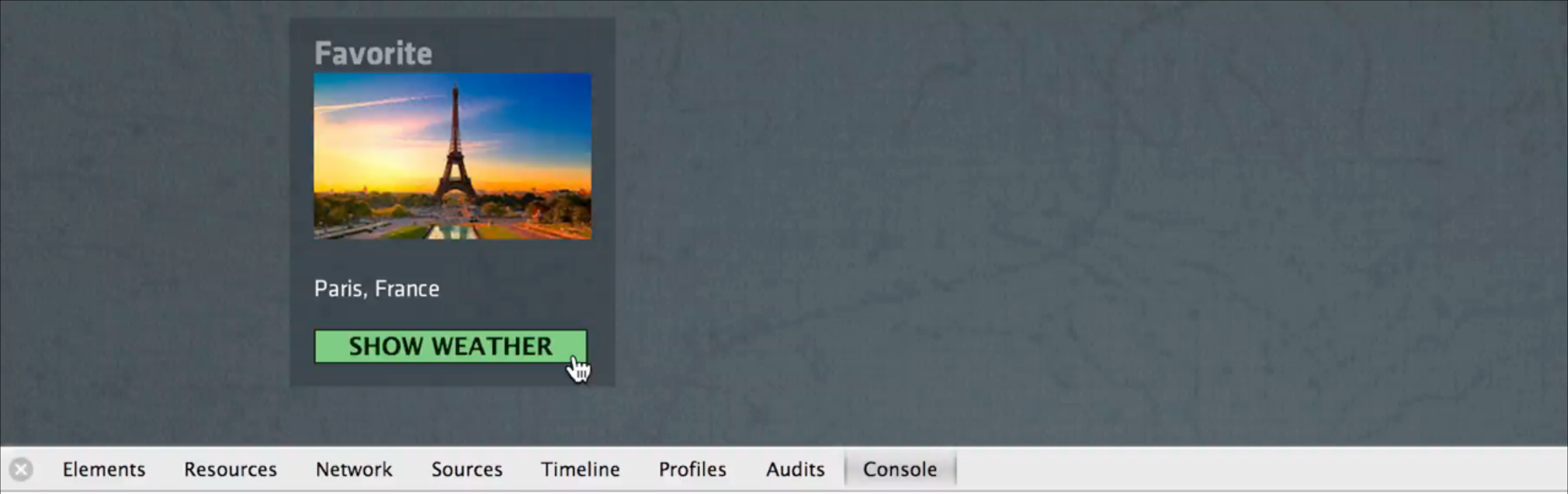
```
$('button').on('click', function() {
  var location = $('.loc').text();
  var todayPromise =
    Weather.today(location);
  todayPromise.done(function(result) {
    $('.weather').text(result);
  }).fail(function(error) {
        console.log(error);
    });
 });
```

```
today: function(location){
  var promise = $.Deferred();
  $.ajax('/weather', {
    data: {q: location},
    success: function(result){
      promise.resolve(result.weather);
    },
    error: function(){
      var error = 'invalid location';
      promise.reject(error);
    }
  });

  return promise;
}
```

```
promise.reject(value);
```

Calls the fail callback

```
promise.fail(function(value){});
```


Favorite
Paris, France
SHOW WEATHER
Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Console

Just a few more Promises

We'll be using weather and one more City object that returns a promise

city.js

```
var City = {
  find: function(location){
    var promise = $.ajax('/cities', {
      data: { loc: location }
    });

    return promise;
  }
}
```

Same Weather object we just built

weather.js

```
var Weather = {
  today: function(location){
    var promise = $.Deferred();
    $.ajax('/weather', {
      data: {q: location},
      success: function(result){
        promise.resolve(result.weather);
      },
      error: function() {
        var error = 'invalid location';
        promise.reject(error);
      }
    });

    return promise;
  }
}
```

London, UK

MORE INFO

Paris, France

MORE INFO

Madrid, Spain

MORE INFO

Calling 2 promises one after the next

application.js

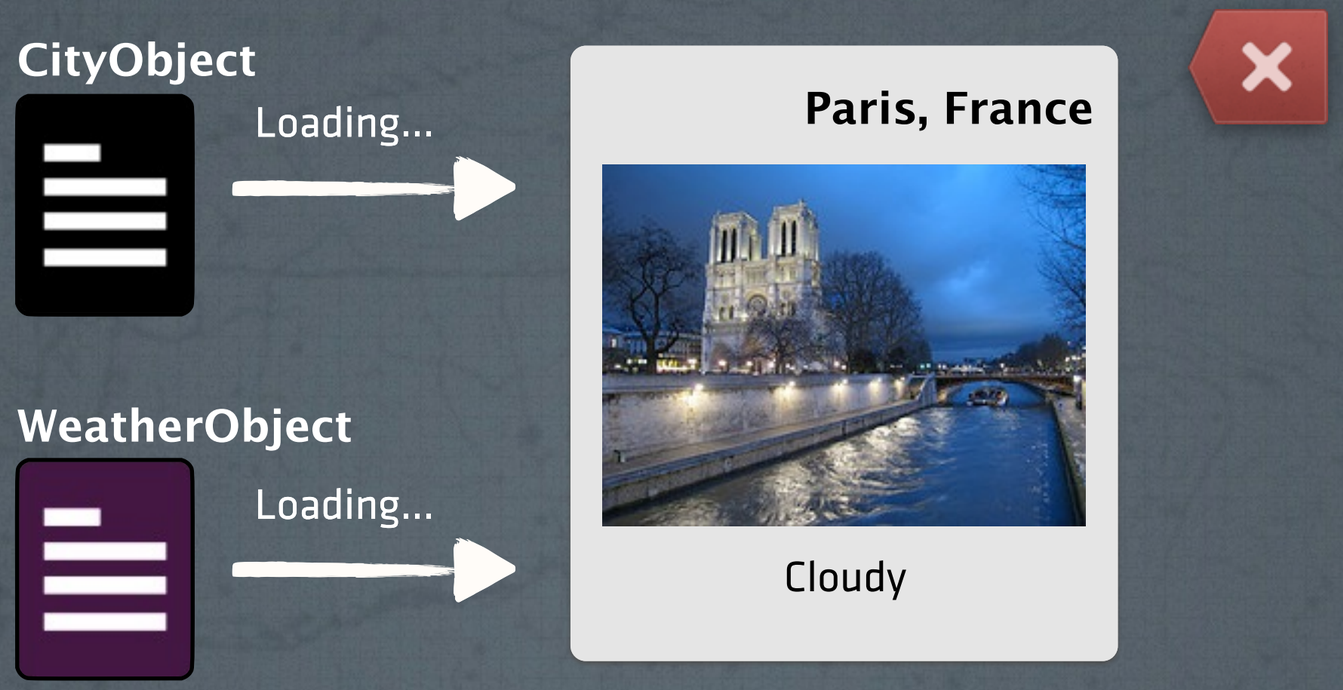
```
$('button').on('click', function() {
  var loc = $(this).parent().data('loc');
  var resultsDiv = $(this).parent()
                   .find('.results').empty();

  Weather.today(loc)
    .done(function(weatherResult){
      resultsDiv.append(weatherResult);
    });

  City.find(loc)
    .done(function(cityResult){
      resultsDiv.append(cityResult);
    });
});
```

AJAX responses finish at different times

favorites.html

```
<li data-loc='london,uk'>
  London, UK
  <button>More Info</button>
  <div class="results"></div>
</li>

<li data-loc='paris,france'>
  Paris, France
  <button>More Info</button>
  <div class="results"></div>
</li>

<li data-loc='madrid,spain'>
  Madrid, Spain
  <button>More Info</button>
  <div class="results"></div>
</li>
```

Ajax requests never load at the same time



Plus elements appear at different times and the layout order is inconsistent.

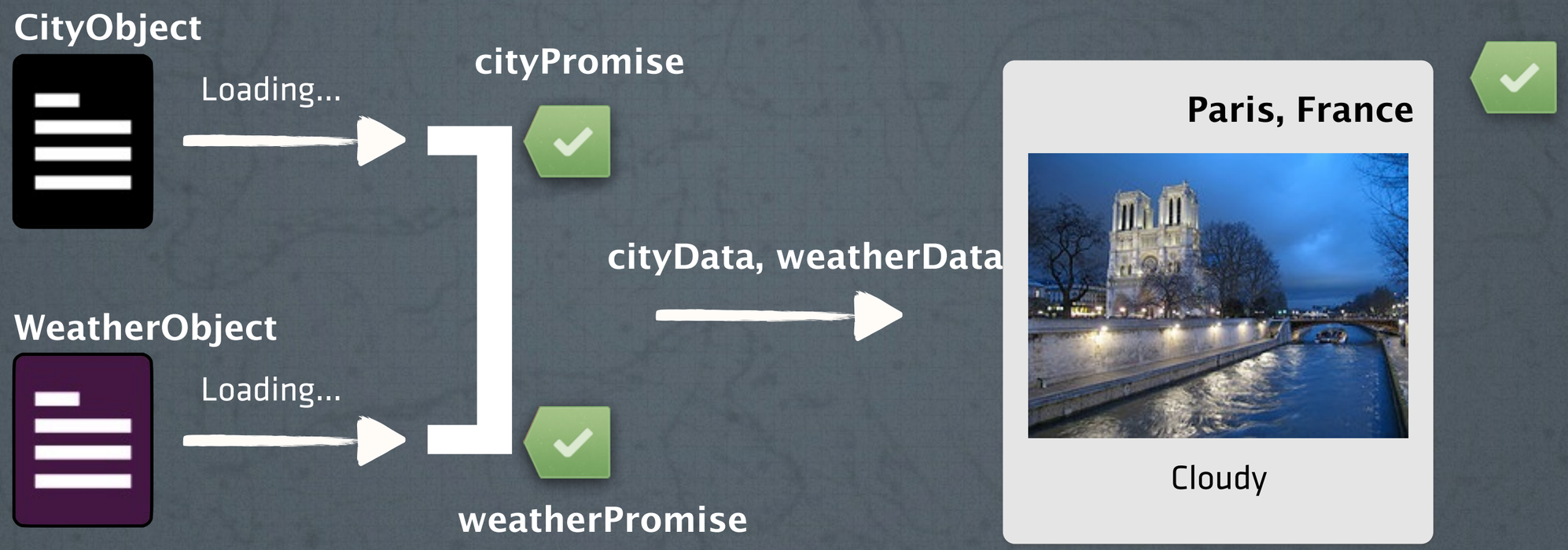
$.when() and then() to save the day

This can't be an array, only promises separated by comas.

```
$.when(<promise1>, <promise2>...)
```

Callback data is in the same order as the promises

```
.then(function(p1Data, p2Data){});
```



$.when() and then() to save the day

application.js

```
$('button').on('click', function() {
  var loc = $(this).parent().data('loc');
  var resultsDiv = $(this).parent()
                   .find('.results').empty();

  Weather.today(loc)
    .done(function(weatherResult){
      resultsDiv.append(weatherResult);
    });

  City.find(loc)
    .done(function(cityResult){
      resultsDiv.append(cityResult);
    });
});
```

Lets use $.when().then() to make these AJAX calls finish at the same time

$.when() and then() to save the day

application.js

```
$('button').on('click', function() {
  var loc = $(this).parent().data('loc');
  var resultsDiv = $(this).parent()
                   .find('.results').empty();

  $.when(
    Weather.today(loc),
    City.find(loc)
  ).then(function(weatherResult, cityResult){
    resultsDiv.append(cityResult);
    resultsDiv.append(weatherResult);
  });
});
```

Data is all rendered at the same time

London, UK

MORE INFO

Paris, France

MORE INFO

Madrid, Spain

MORE INFO