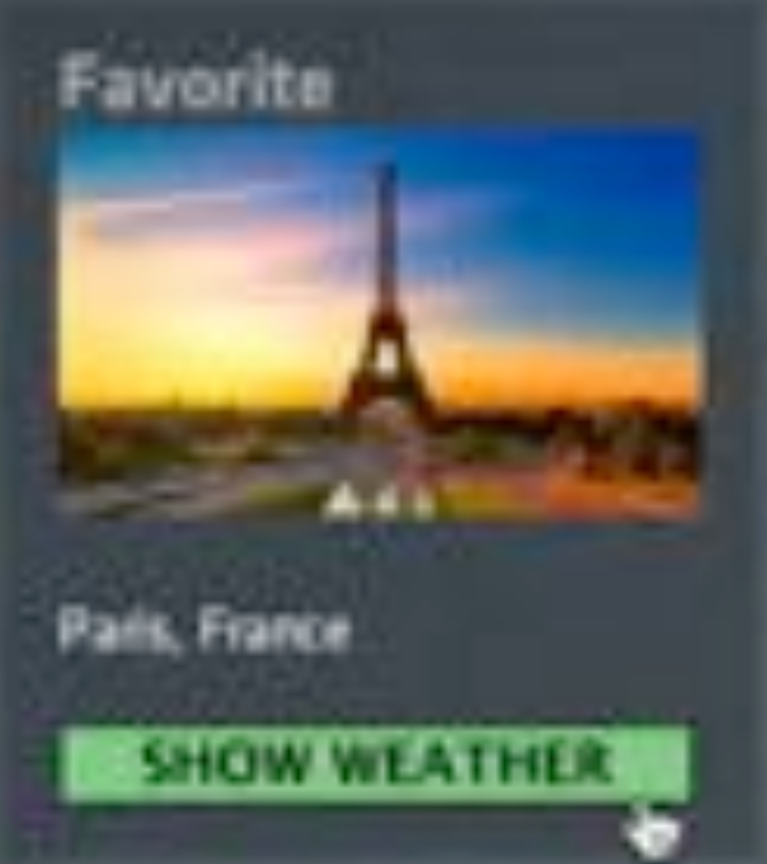
Promises


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"
}
```

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

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>
```

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);
  });
});
```

On Ajax success, done will be called on the promise

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

    return 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);
  });
});
```

Calling resolve will trigger our done callback

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

    });

    return promise;
  }
}
```

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);
  });
});
```

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

Calls the fail callback

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

```
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;
}
```

.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);
    });
 });
```

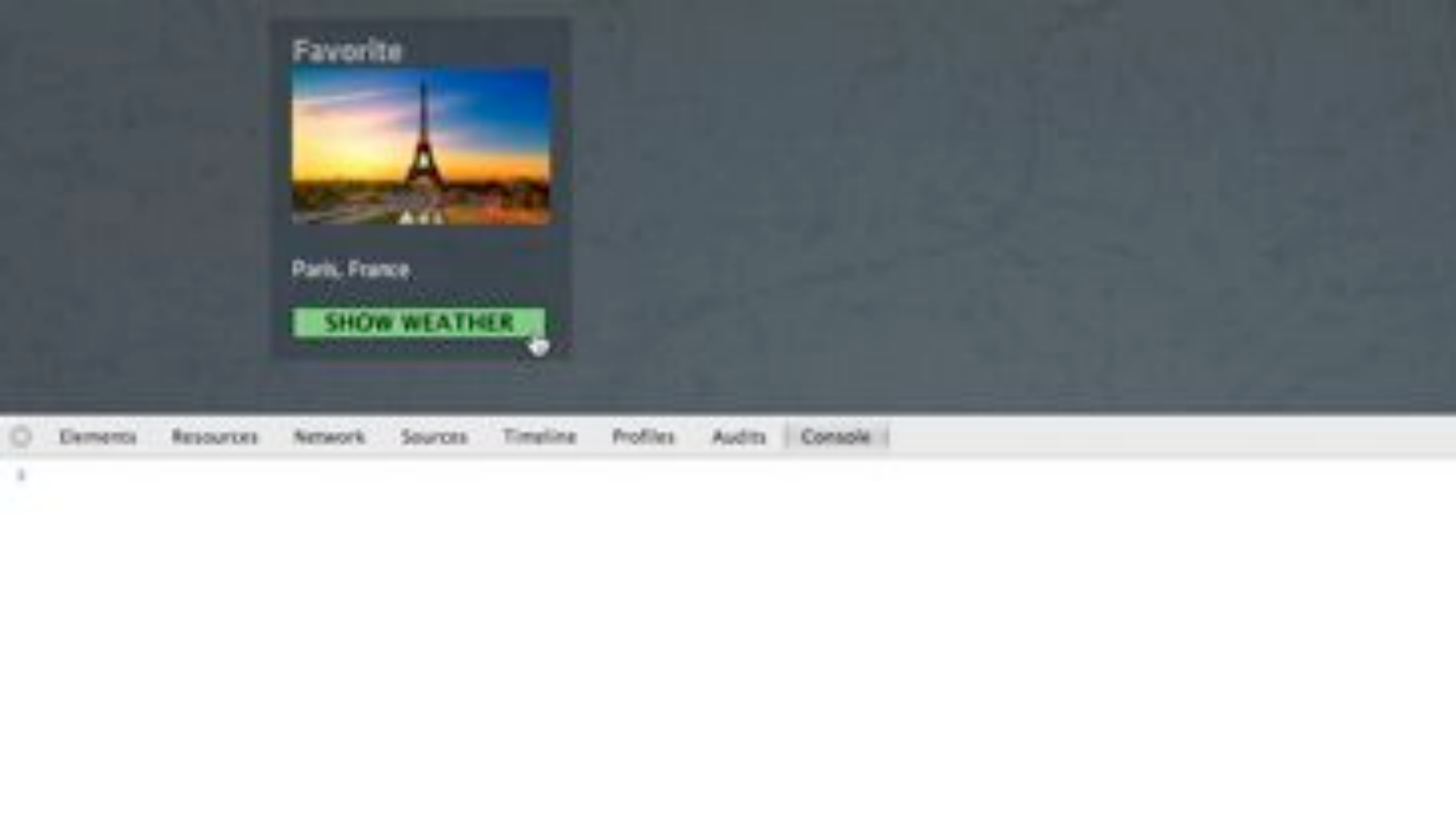
```
today: function(location){
  var promise = $.Deferred();
  $.ajax('/weather', {
    data: {q: location},
    success: function(result){
      promise.resolve(result.weather);
    },
    error: function(){
      promise.reject(error);
      var error = 'invalid location';
    }
  });

  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.getWeather(loc)
    .done(function(weatherResult){
      resultDiv.append(weatherResult);
    });

  City.getCity(loc)
    .done(function(cityResult){
      resultDiv.append(cityResult.html);
    });
});
```

AJAX responses finish at different times

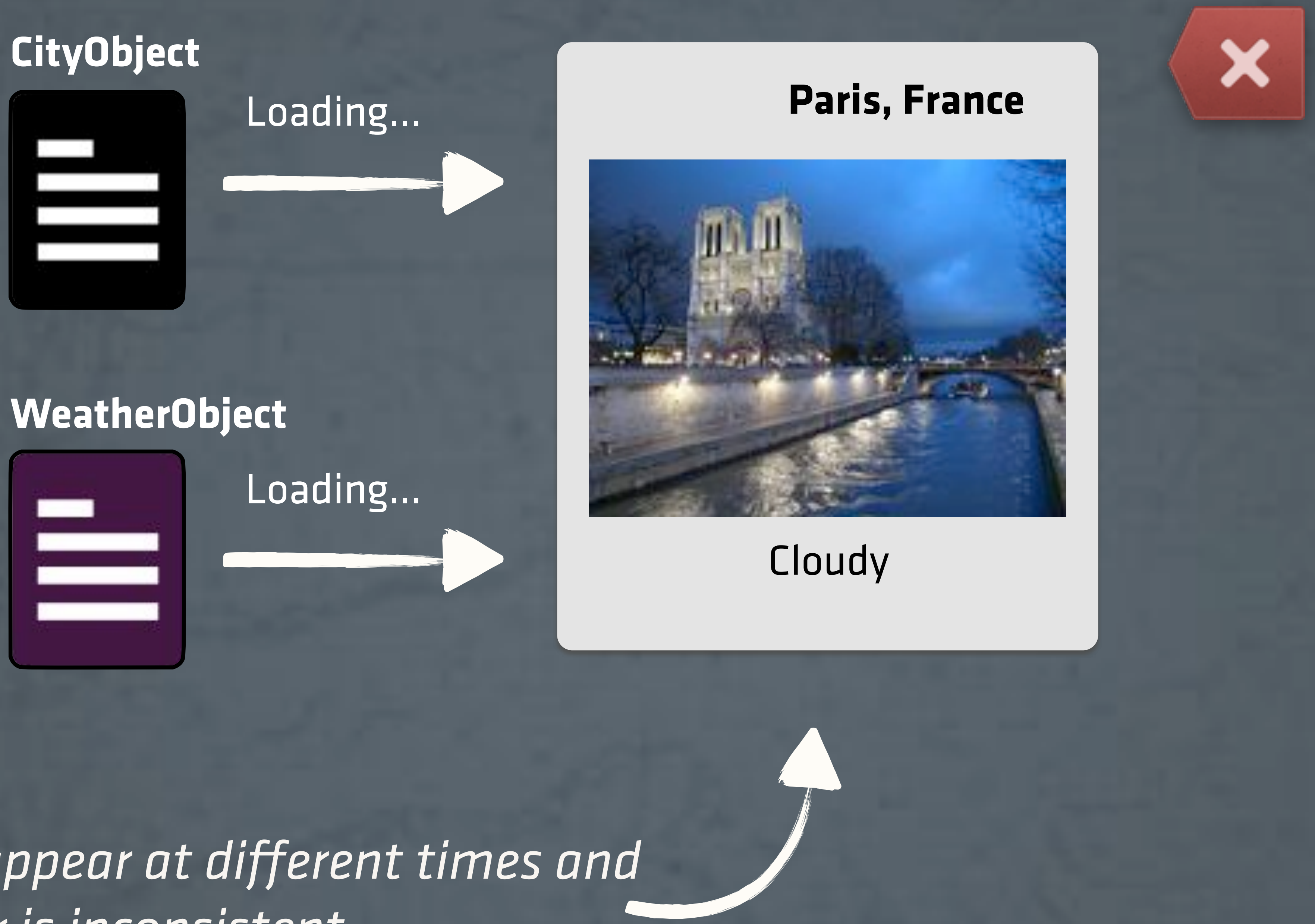
favorites.html

```
<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.

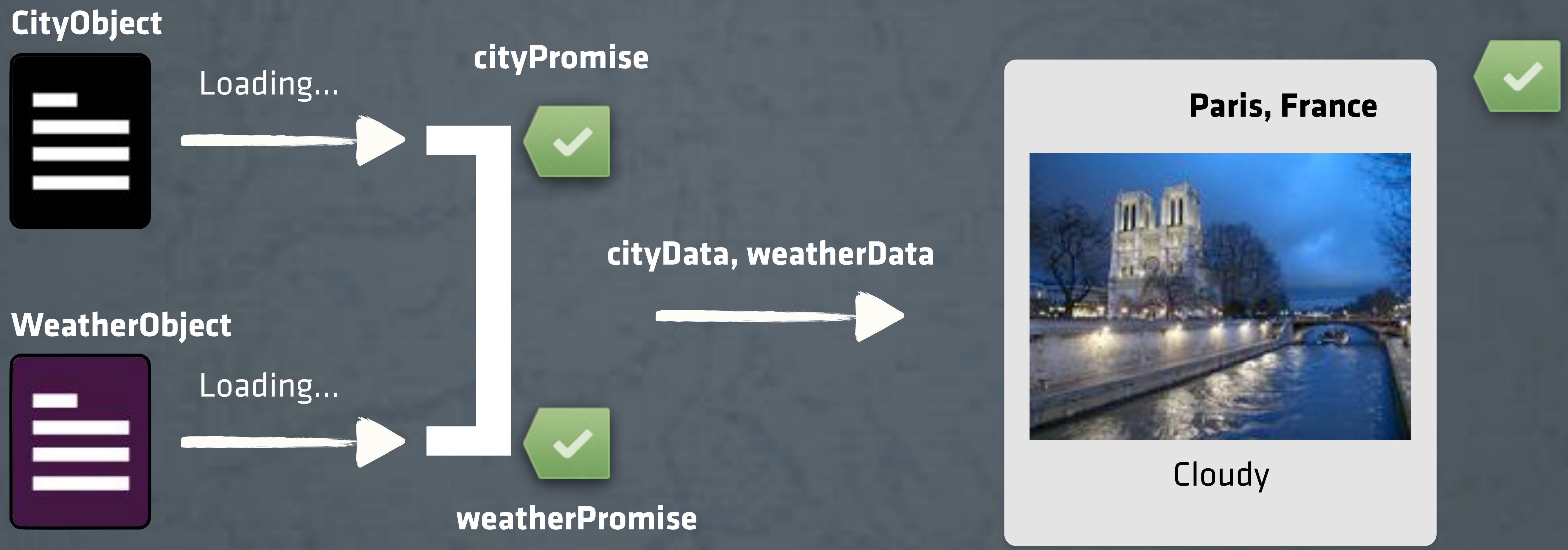
$.when() and then() to save the day

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

Callback data is in the same order as the promises

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

```
.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){
      resultDiv.append(weatherResult);
    });

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

Data is all rendered at the same time

London, UK

MORE INFO

Paris, France

MORE INFO

Madrid, Spain

MORE INFO