HTML5 and CSS3 for Mobile Web Applications

Prof. Paul Krause, University of Surrey

Promises

Objectives of this lecture

- What is a promise?
- Some simple examples of promises in action (including Angular.js)

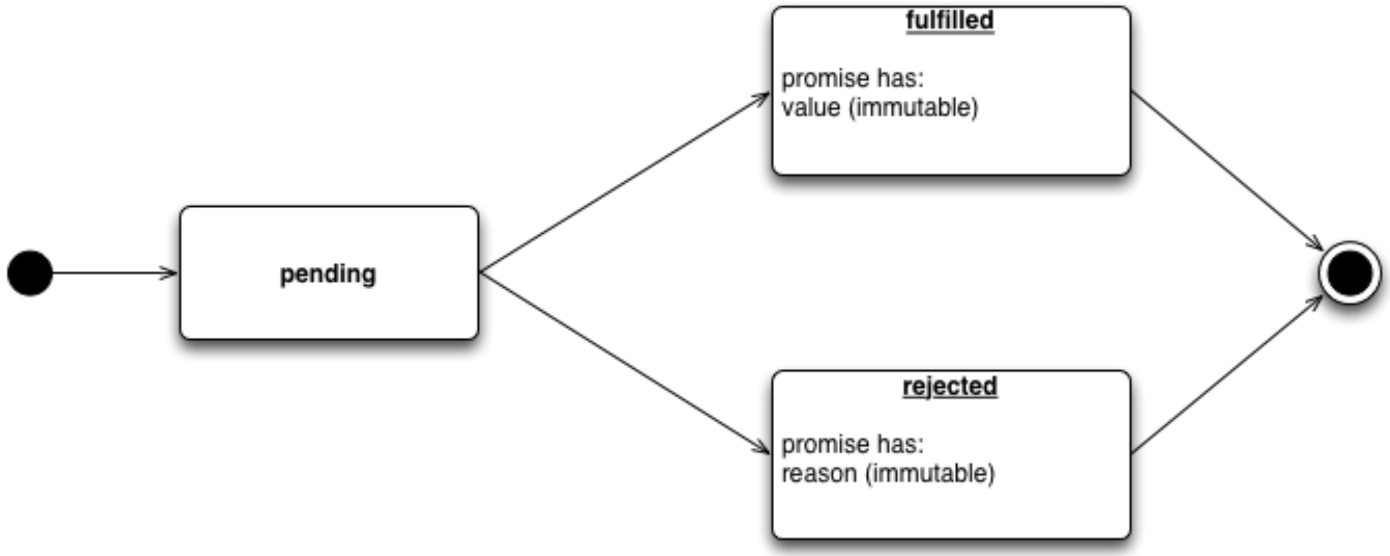
What is a promise?

- "A promise is the eventual result of an asynchronous operation"

Promises/A+ specification

- A *promise* must have a *then* method
- The *then* method registers callbacks to (eventually):
  - Receive the value of the *promise*, or
  - Receive the reason why the promise cannot be fulfilled

State model of a promise



How do we access the result?

- Through the promise’s *then* method:

- promise.then(onFulfilled, onRejected)

How do we access the result?

- Through the promise's *then* method:

- promise.then(onFulfilled, onRejected)

if not a function:
    it will be ignored
if a function:
    it must be called after the promise is <u>fulfilled</u>,
    with the promise's <u>value</u> as its first argument

How do we access the result?

- Through the promise's *then* method:

- promise.then(onFulfilled, onRejected)

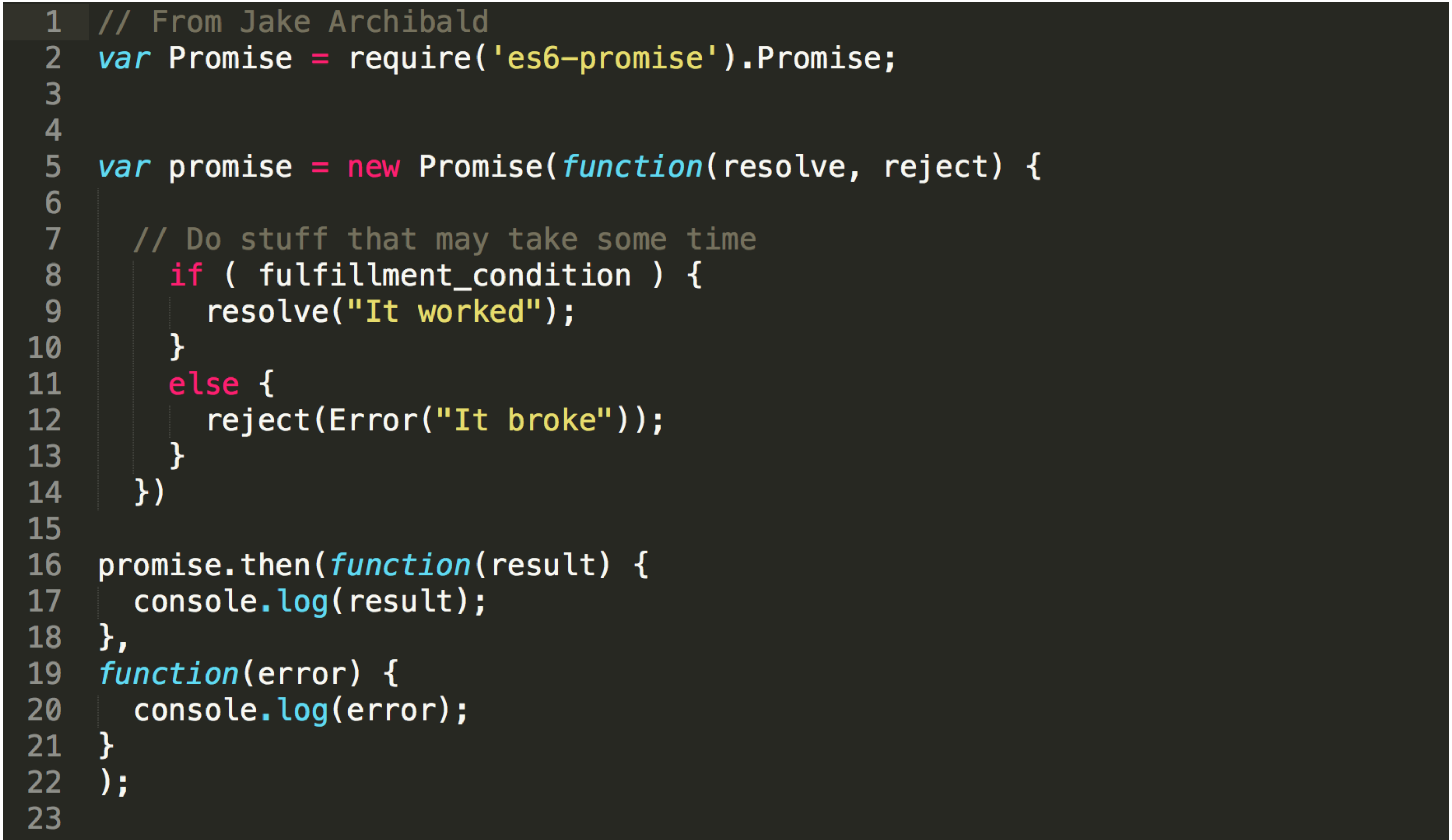
if not a function:
    it will be ignored
if a function:
    it must be called after the promise is rejected,
    with the promise's reason as its first argument

Promises support

- Enabled by default as of Chrome 32 and Opera 19
- Partial support in Firefox nightly
- Otherwise

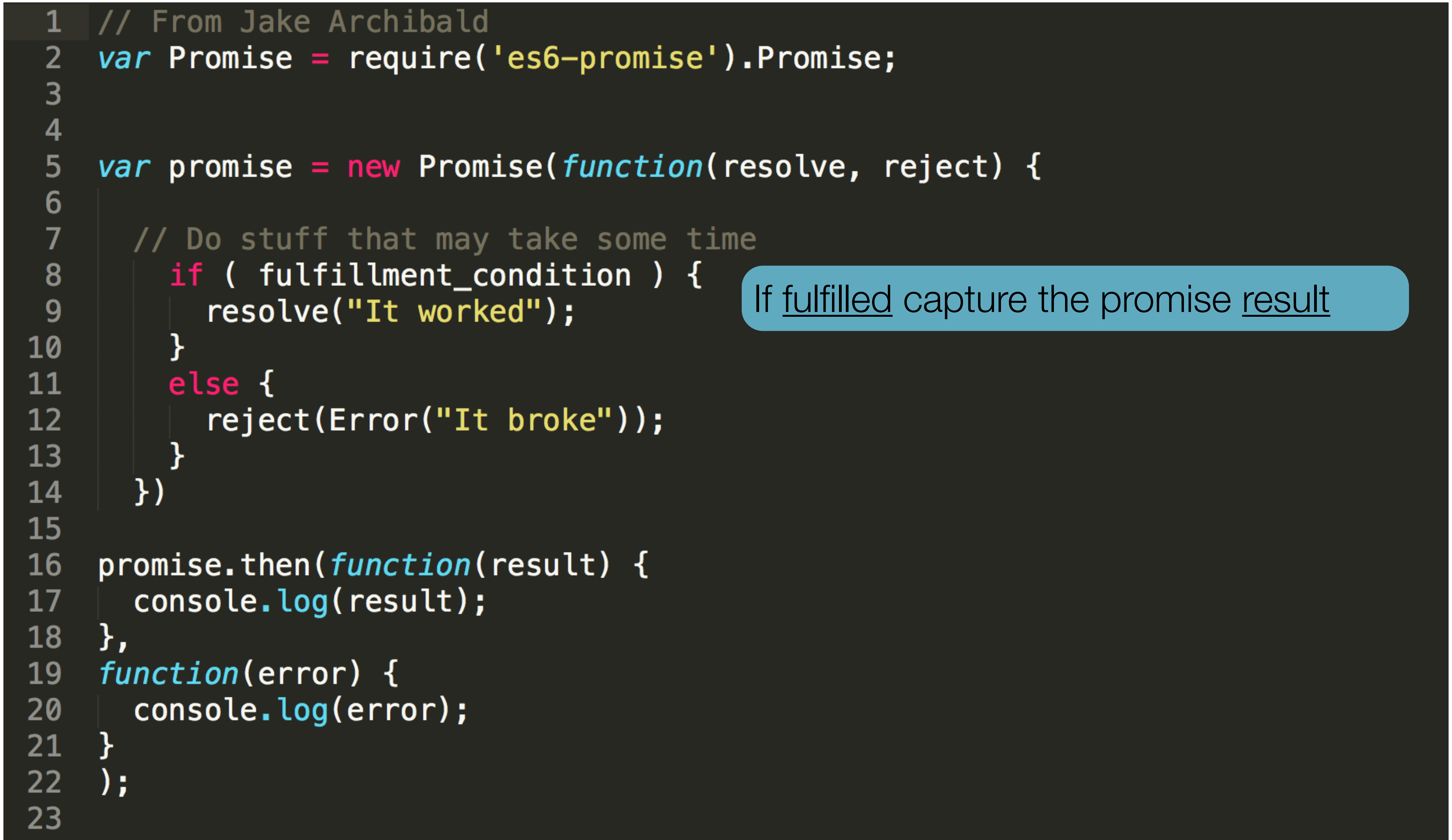
```
     npm install es6-promise
```

Pattern for using promises



```javascript
// From Jake Archibald
var Promise = require('es6-promise').Promise;


var promise = new Promise(function(resolve, reject) {

  // Do stuff that may take some time
    if ( fulfillment_condition ) {
      resolve("It worked");
    }
    else {
      reject(Error("It broke"));
    }
  })

promise.then(function(result) {
  console.log(result);
},
function(error) {
  console.log(error);
}
);
```

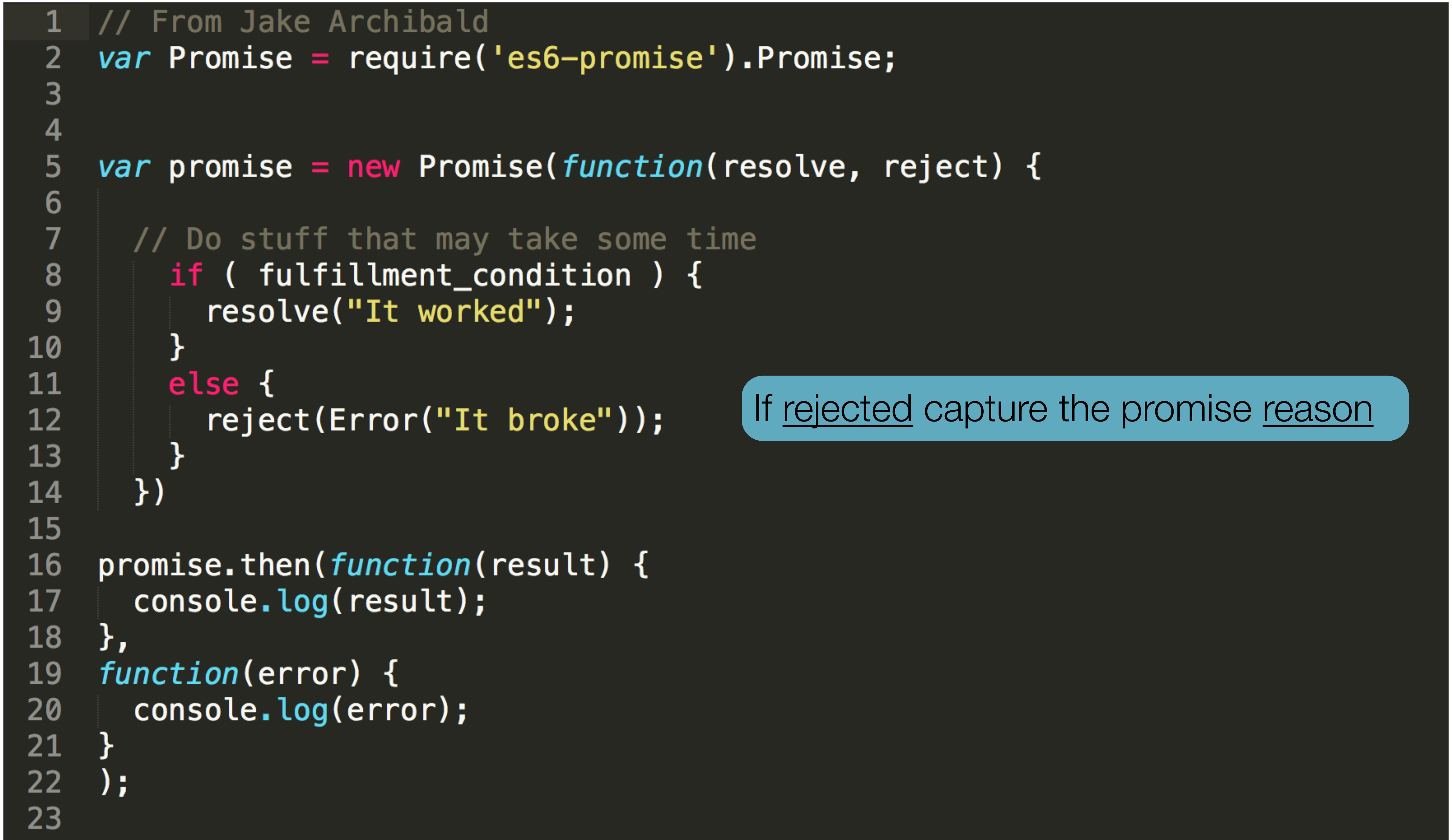
Pattern for using promises



```
// From Jake Archibald
var Promise = require('es6-promise').Promise;


var promise = new Promise(function(resolve, reject) {

  // Do stuff that may take some time
    if ( fulfillment_condition ) {
      resolve("It worked");
    }
    else {
      reject(Error("It broke"));
    }
  })

promise.then(function(result) {
  console.log(result);
},
function(error) {
  console.log(error);
}
);

```

If fulfilled capture the promise result

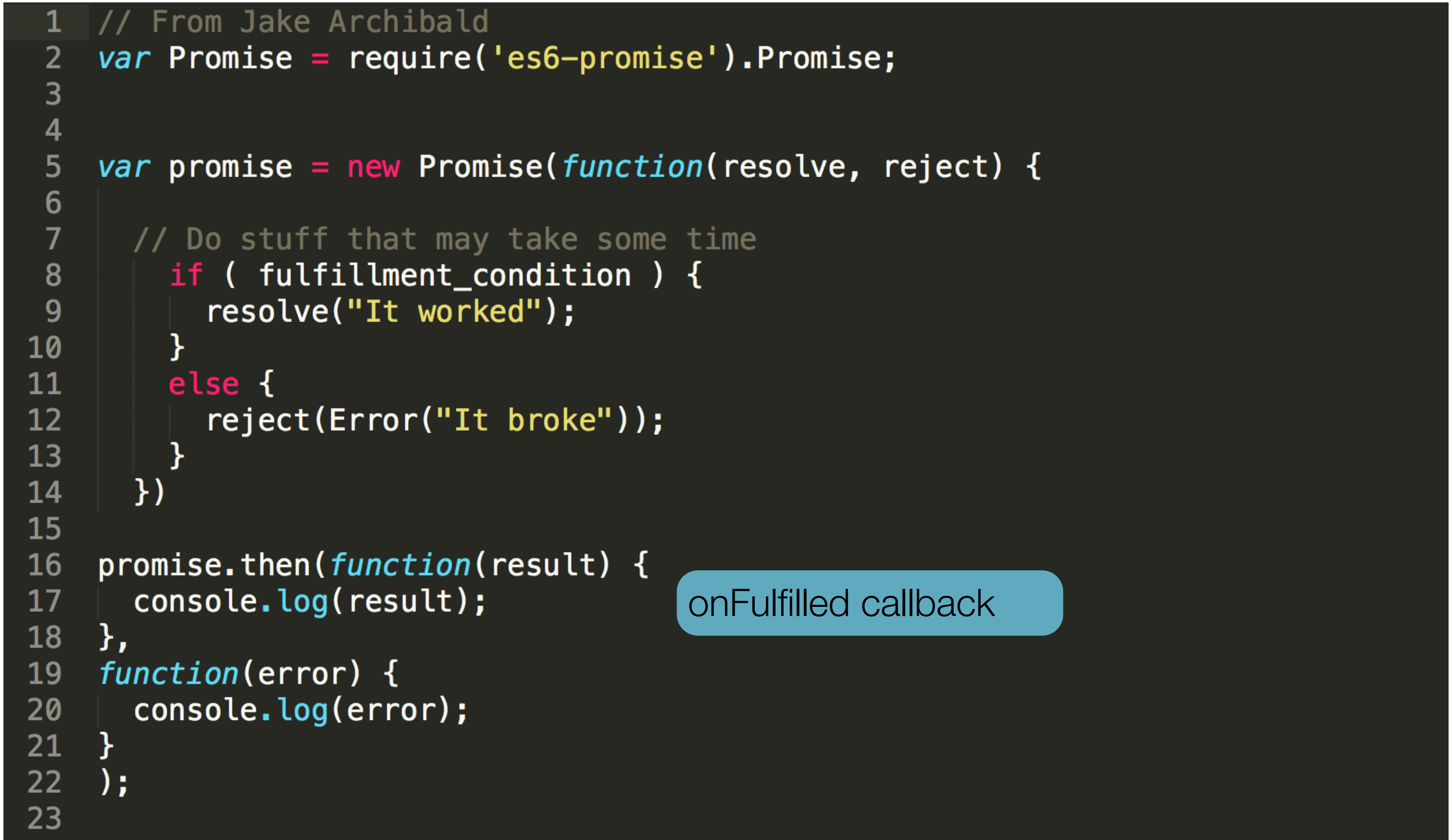
Pattern for using promises



```javascript
// From Jake Archibald
var Promise = require('es6-promise').Promise;


var promise = new Promise(function(resolve, reject) {

  // Do stuff that may take some time
    if ( fulfillment_condition ) {
      resolve("It worked");
    }
    else {
      reject(Error("It broke"));
    }
  })

promise.then(function(result) {
  console.log(result);
},
function(error) {
  console.log(error);
}
);

```

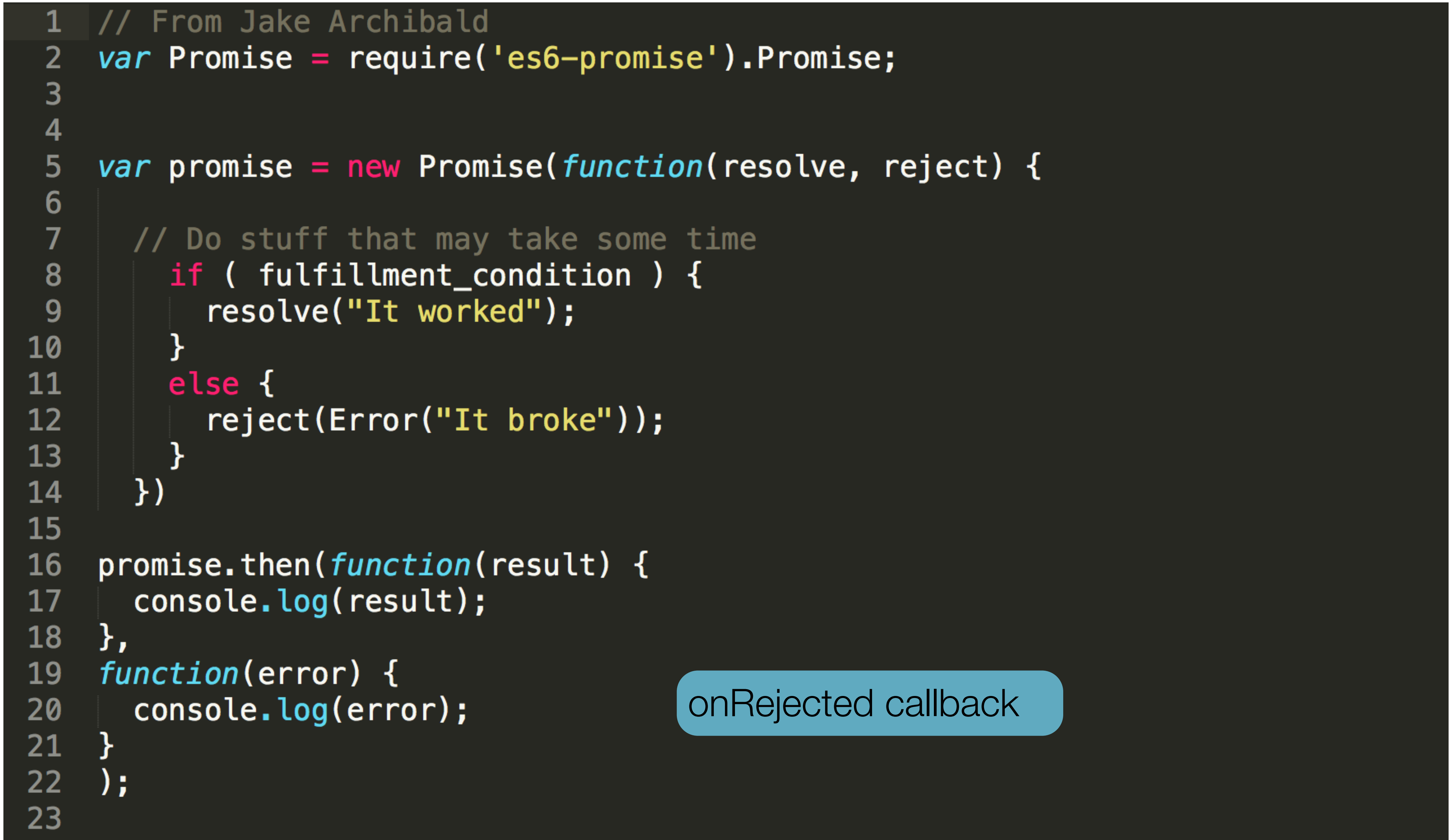
If rejected capture the promise reason

Pattern for using promises



```
// From Jake Archibald
var Promise = require('es6-promise').Promise;


var promise = new Promise(function(resolve, reject) {

  // Do stuff that may take some time
    if ( fulfillment_condition ) {
      resolve("It worked");
    }
    else {
      reject(Error("It broke"));
    }
  })

promise.then(function(result) {
  console.log(result);
},
function(error) {
  console.log(error);
}
);

```

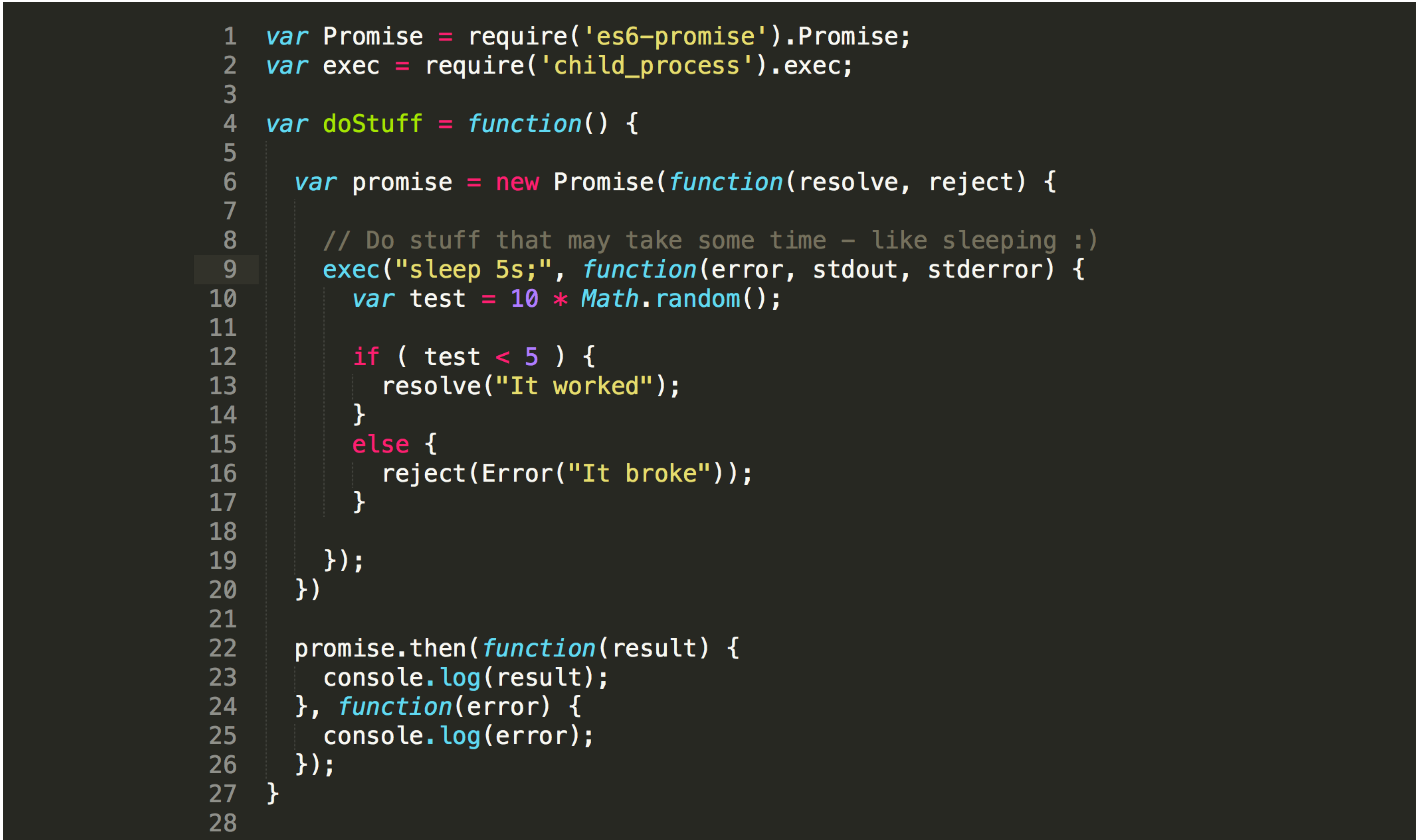
Pattern for using promises



```javascript
// From Jake Archibald
var Promise = require('es6-promise').Promise;


var promise = new Promise(function(resolve, reject) {

  // Do stuff that may take some time
    if ( fulfillment_condition ) {
      resolve("It worked");
    }
    else {
      reject(Error("It broke"));
    }
  })

promise.then(function(result) {
  console.log(result);
},
function(error) {
  console.log(error);
}
);

```

onRejected callback

Simple test case



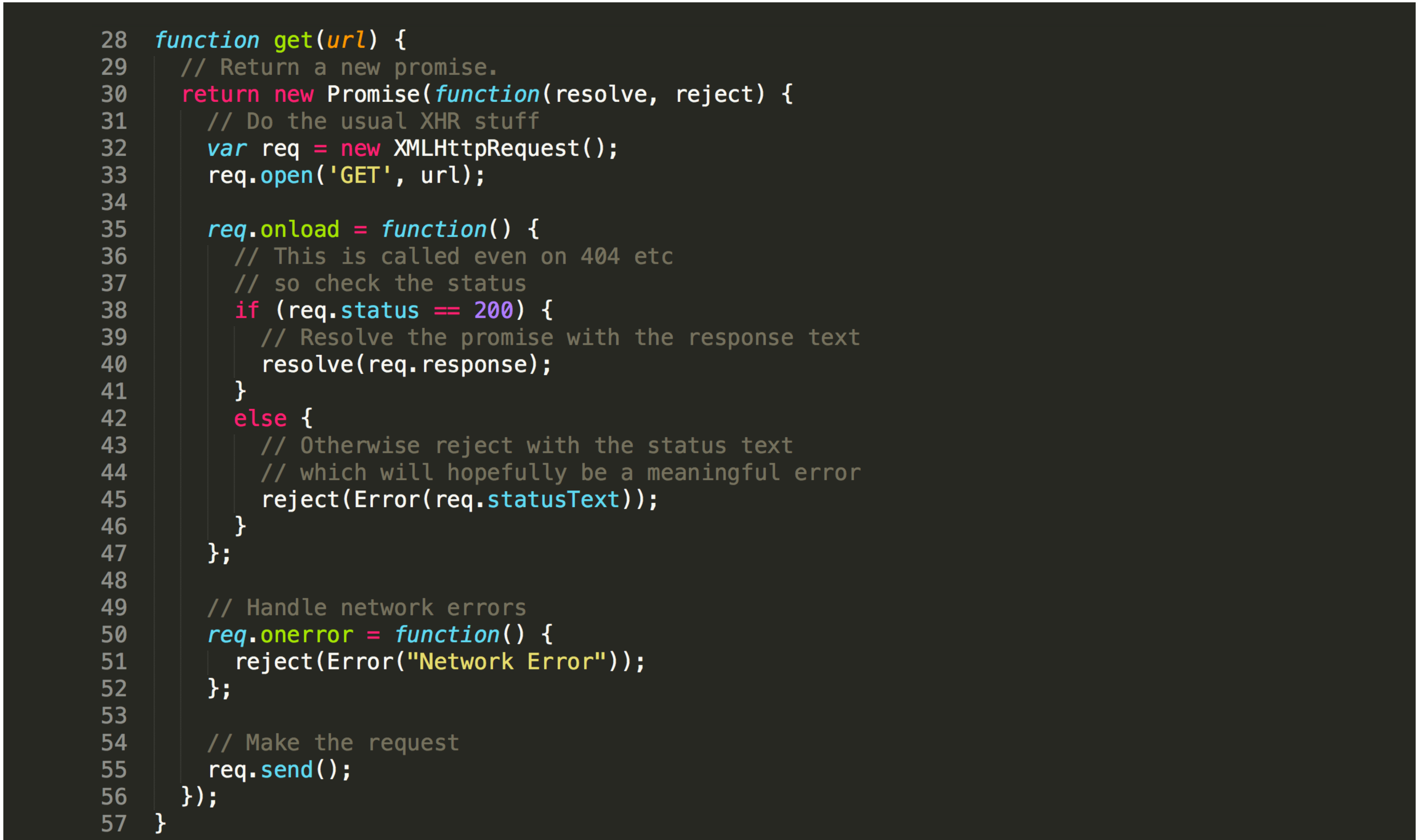
```javascript
var Promise = require('es6-promise').Promise;
var exec = require('child_process').exec;

var doStuff = function() {

  var promise = new Promise(function(resolve, reject) {

    // Do stuff that may take some time - like sleeping :)
    exec("sleep 5s;", function(error, stdout, stderror) {
      var test = 10 * Math.random();

      if ( test < 5 ) {
        resolve("It worked");
      }
      else {
        reject(Error("It broke"));
      }

    });
  })

  promise.then(function(result) {
    console.log(result);
  }, function(error) {
    console.log(error);
  });
}

```

Try me!

(Just load promises.js into a REPL and execute doStuff( ))

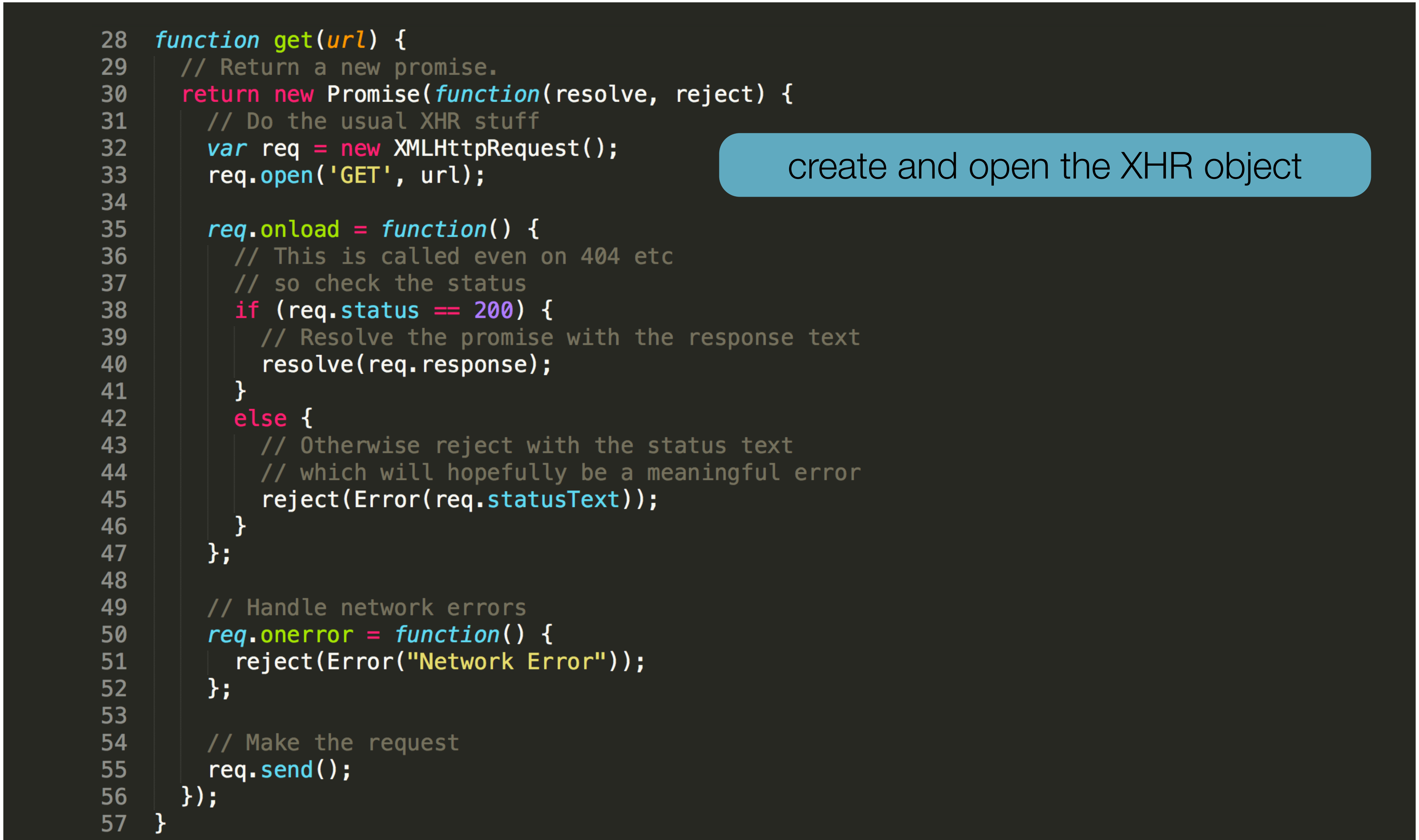
“Promise-ifying Ajax

After Jake Archibald



```javascript
function get(url) {
  // Return a new promise.
  return new Promise(function(resolve, reject) {
    // Do the usual XHR stuff
    var req = new XMLHttpRequest();
    req.open('GET', url);

    req.onload = function() {
      // This is called even on 404 etc
      // so check the status
      if (req.status == 200) {
        // Resolve the promise with the response text
        resolve(req.response);
      }
      else {
        // Otherwise reject with the status text
        // which will hopefully be a meaningful error
        reject(Error(req.statusText));
      }
    };

    // Handle network errors
    req.onerror = function() {
      reject(Error("Network Error"));
    };

    // Make the request
    req.send();
  });
}
```

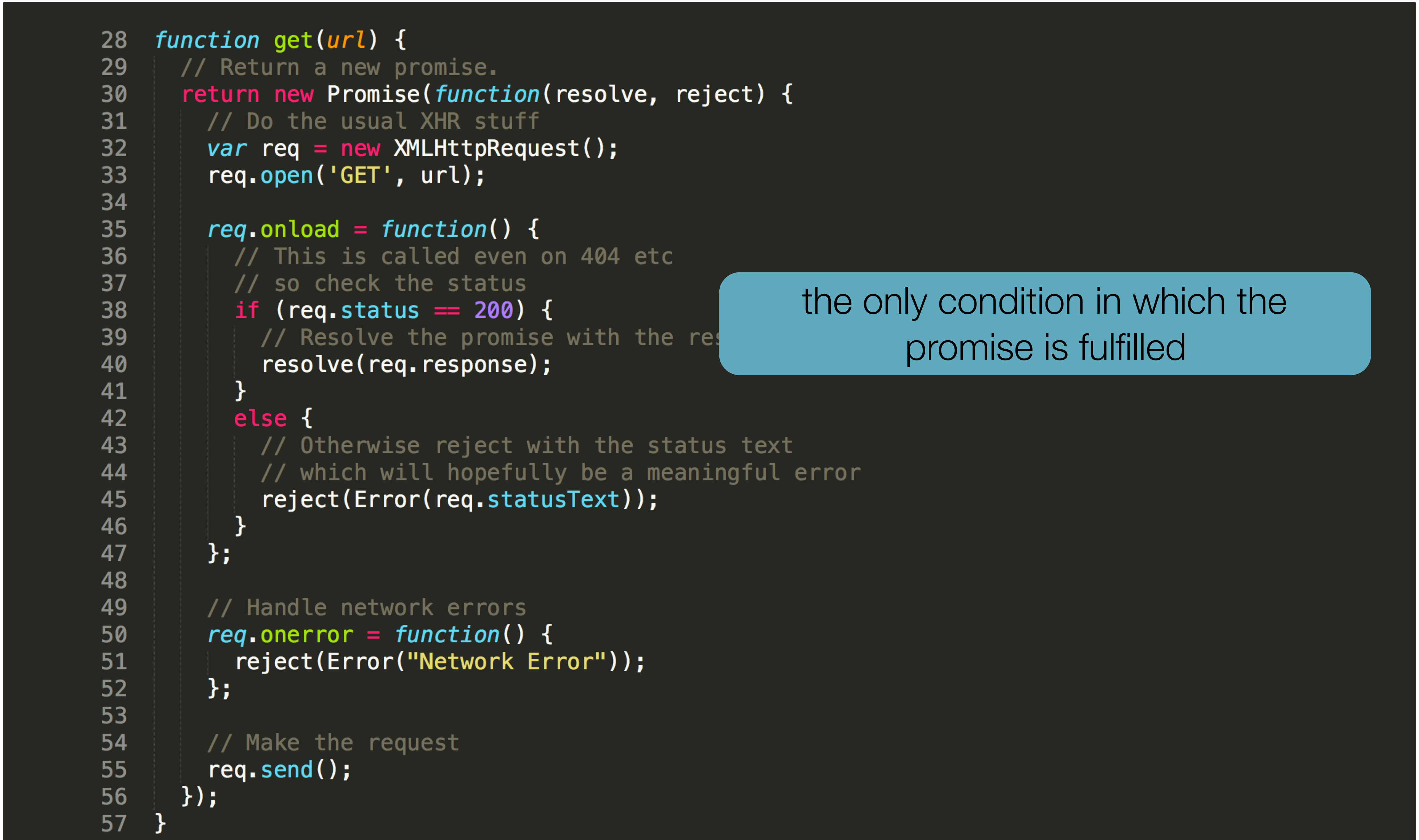
After Jake Archibald



```javascript
function get(url) {
  // Return a new promise.
  return new Promise(function(resolve, reject) {
    // Do the usual XHR stuff
    var req = new XMLHttpRequest();
    req.open('GET', url);

    req.onload = function() {
      // This is called even on 404 etc
      // so check the status
      if (req.status == 200) {
        // Resolve the promise with the response text
        resolve(req.response);
      }
      else {
        // Otherwise reject with the status text
        // which will hopefully be a meaningful error
        reject(Error(req.statusText));
      }
    };

    // Handle network errors
    req.onerror = function() {
      reject(Error("Network Error"));
    };

    // Make the request
    req.send();
  });
}
```

create and open the XHR object

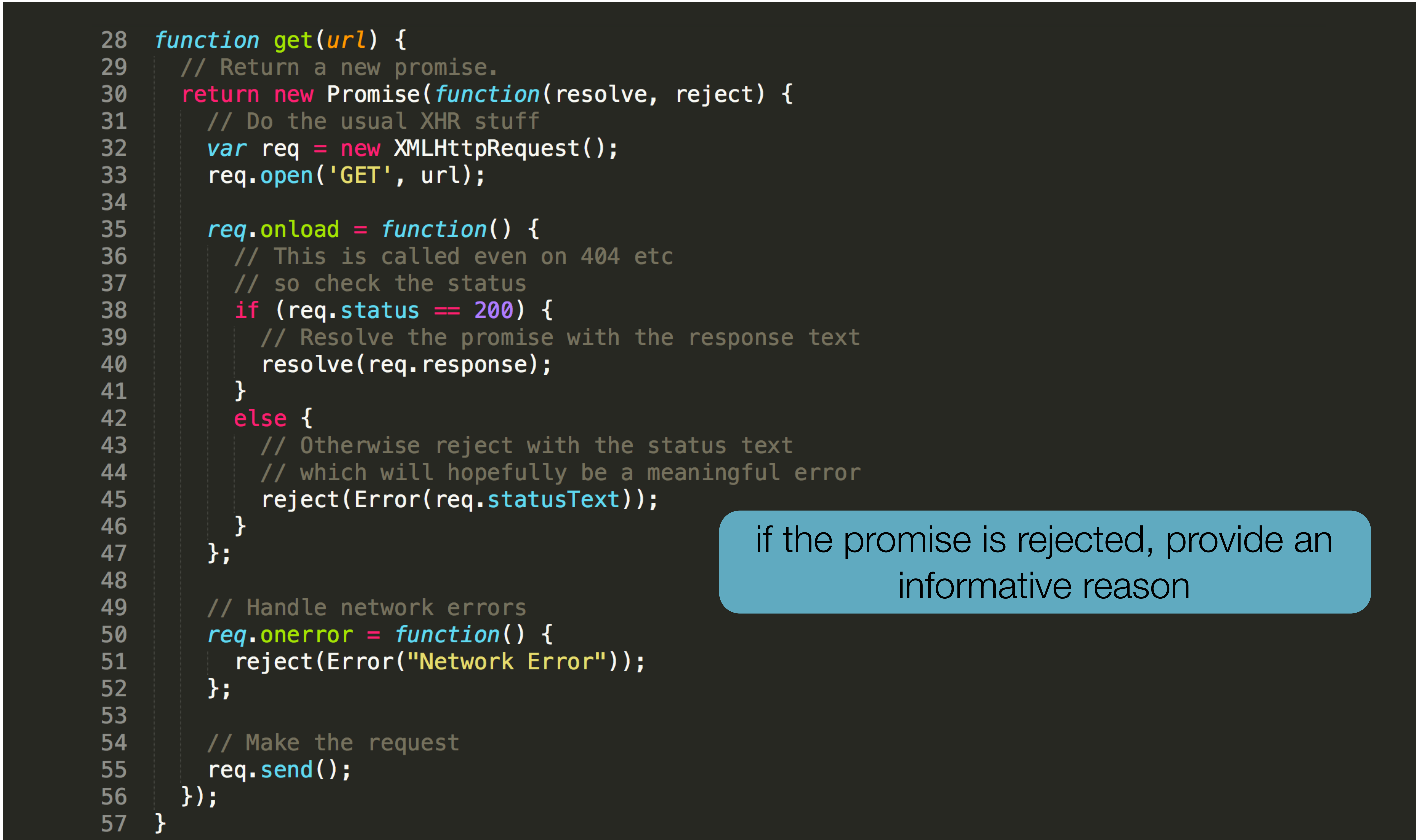
After Jake Archibald



```javascript
function get(url) {
  // Return a new promise.
  return new Promise(function(resolve, reject) {
    // Do the usual XHR stuff
    var req = new XMLHttpRequest();
    req.open('GET', url);

    req.onload = function() {
      // This is called even on 404 etc
      // so check the status
      if (req.status == 200) {
        // Resolve the promise with the res
        resolve(req.response);
      }
      else {
        // Otherwise reject with the status text
        // which will hopefully be a meaningful error
        reject(Error(req.statusText));
      }
    };

    // Handle network errors
    req.onerror = function() {
      reject(Error("Network Error"));
    };

    // Make the request
    req.send();
  });
}
```

the only condition in which the promise is fulfilled

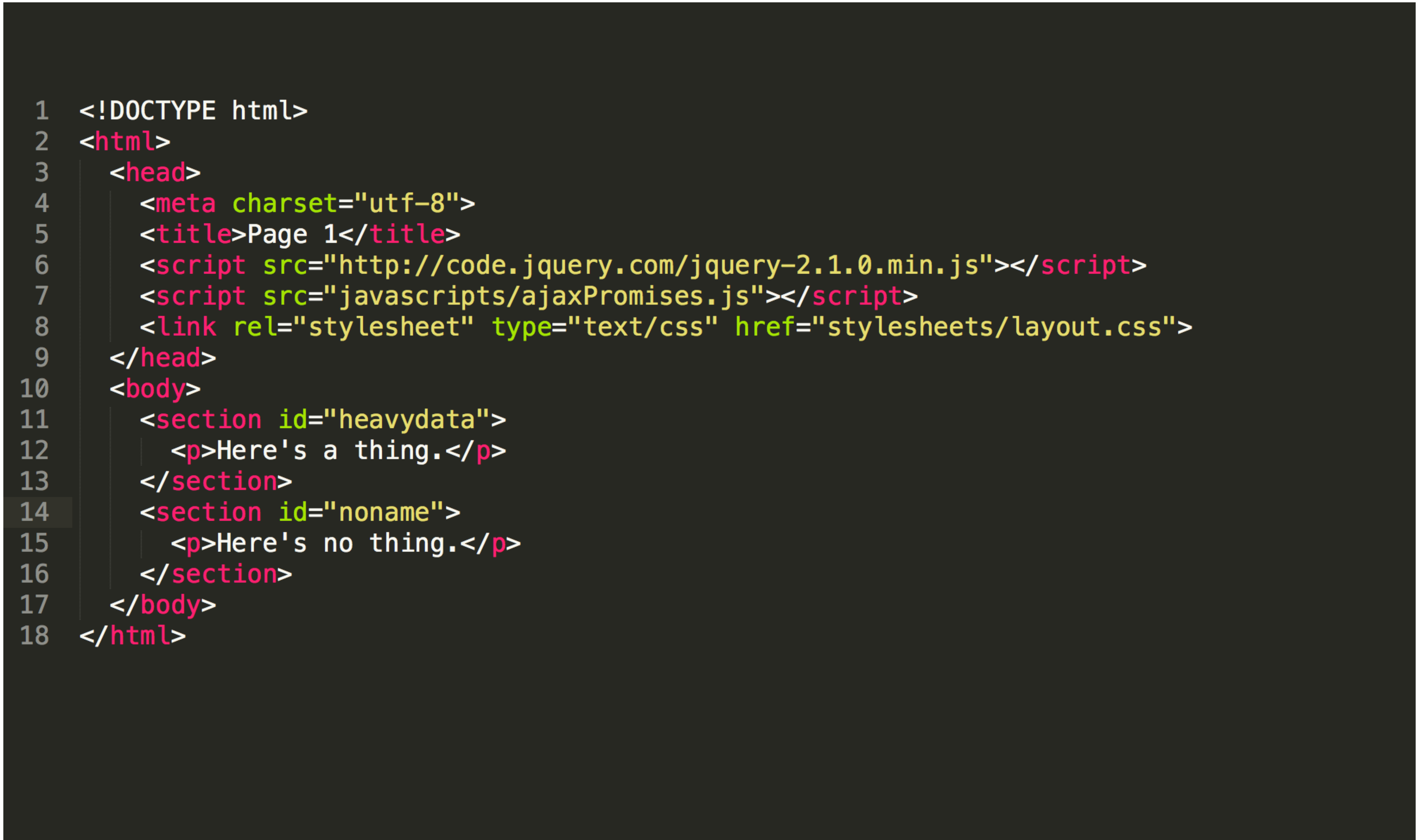
After Jake Archibald



```
function get(url) {
  // Return a new promise.
  return new Promise(function(resolve, reject) {
    // Do the usual XHR stuff
    var req = new XMLHttpRequest();
    req.open('GET', url);

    req.onload = function() {
      // This is called even on 404 etc
      // so check the status
      if (req.status == 200) {
        // Resolve the promise with the response text
        resolve(req.response);
      }
      else {
        // Otherwise reject with the status text
        // which will hopefully be a meaningful error
        reject(Error(req.statusText));
      }
    };

    // Handle network errors
    req.onerror = function() {
      reject(Error("Network Error"));
    };

    // Make the request
    req.send();
  });
}
```

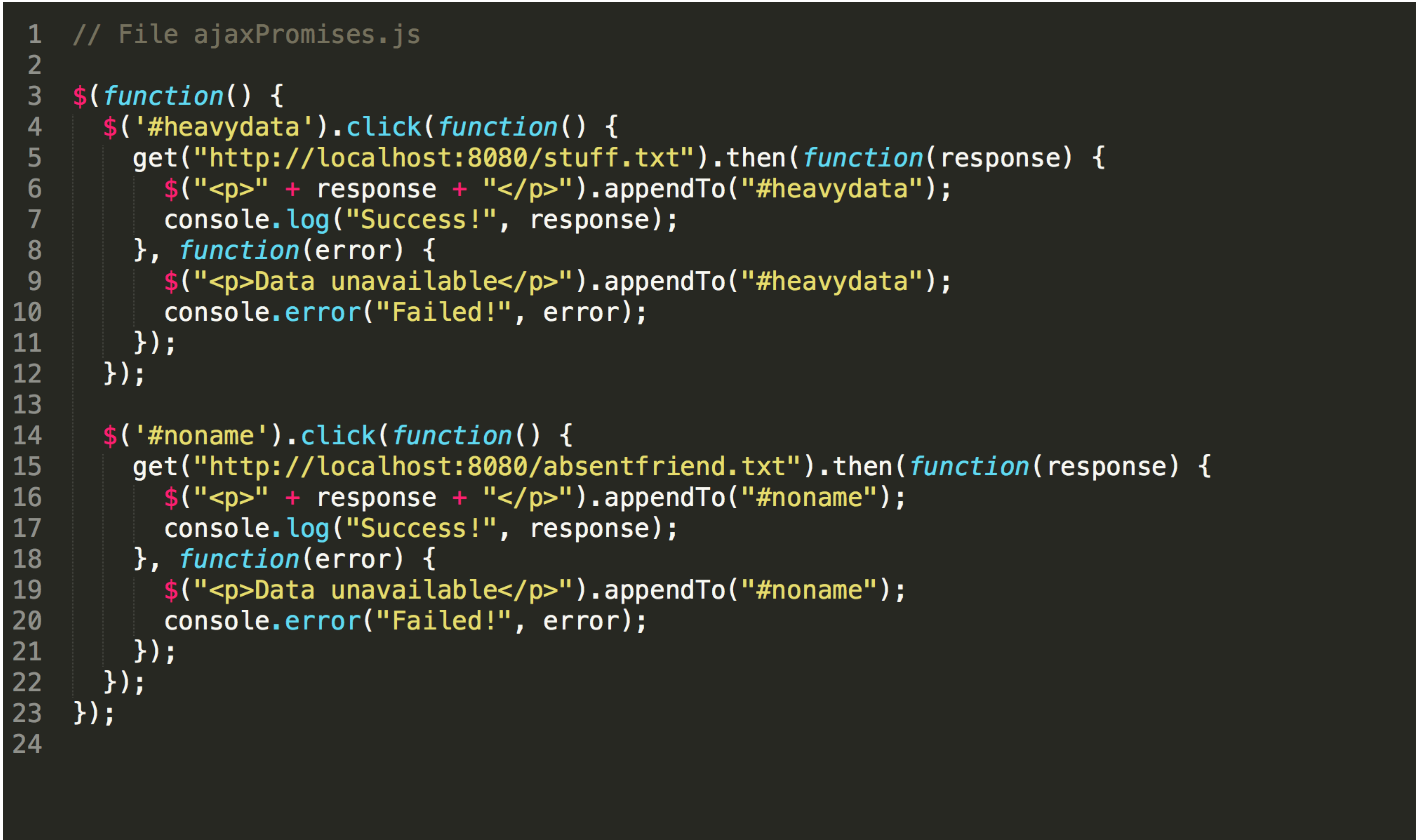
if the promise is rejected, provide an informative reason

My index.html



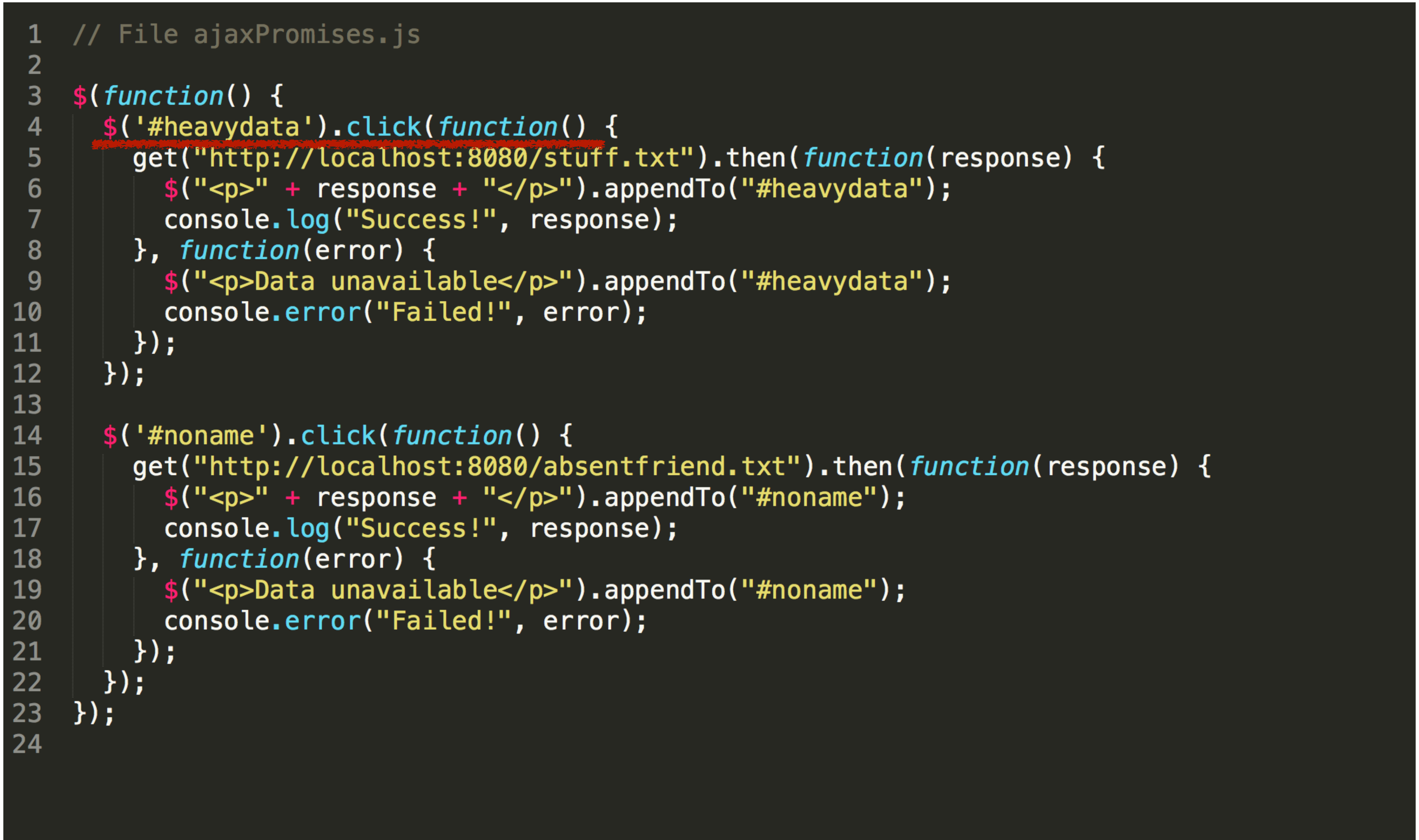
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Page 1</title>
    <script src="http://code.jquery.com/jquery-2.1.0.min.js"></script>
    <script src="javascripts/ajaxPromises.js"></script>
    <link rel="stylesheet" type="text/css" href="stylesheets/layout.css">
  </head>
  <body>
    <section id="heavydata">
      <p>Here's a thing.</p>
    </section>
    <section id="noname">
      <p>Here's no thing.</p>
    </section>
  </body>
</html>
```

file: ajaxPromises.js



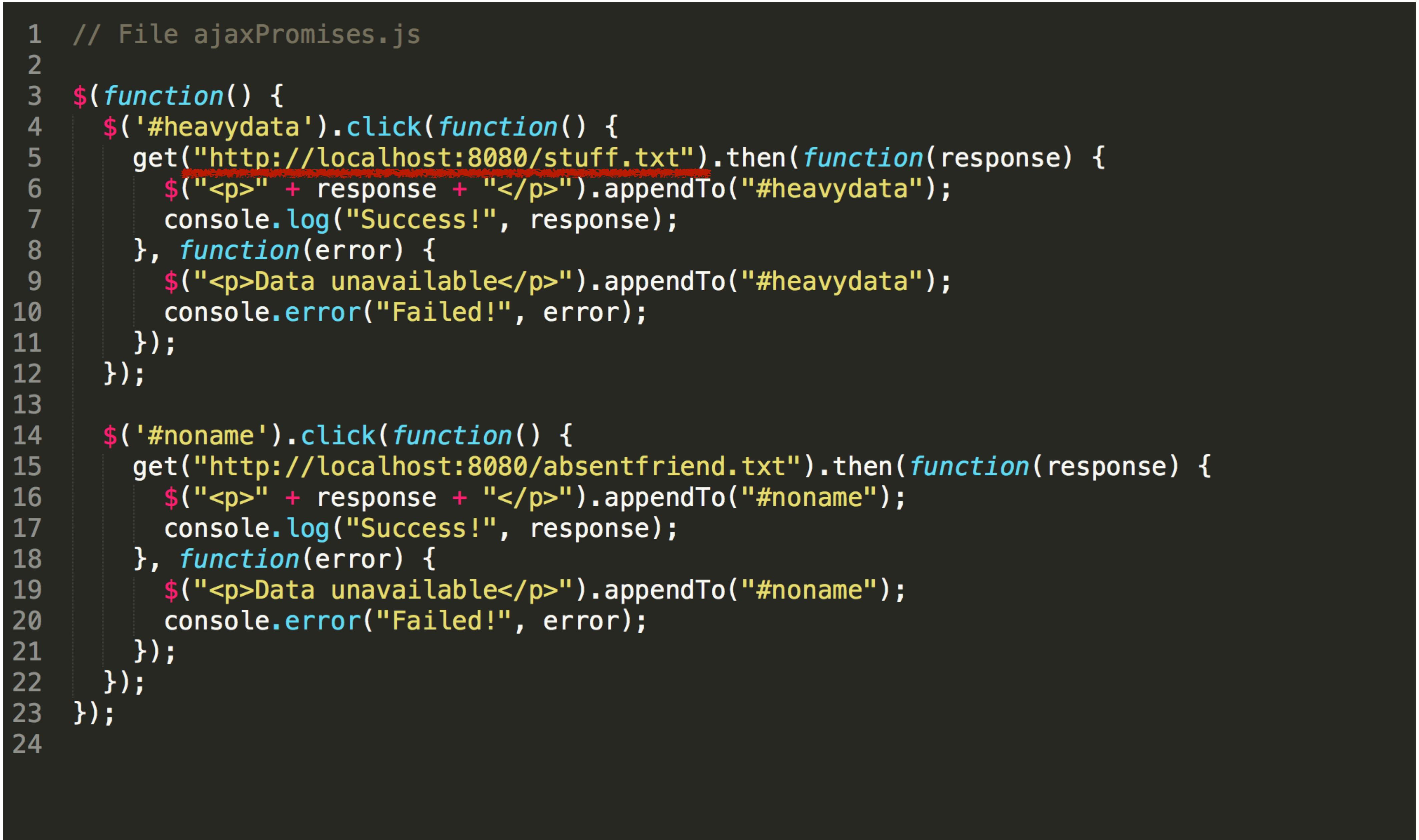
```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});

```

file: ajaxPromises.js



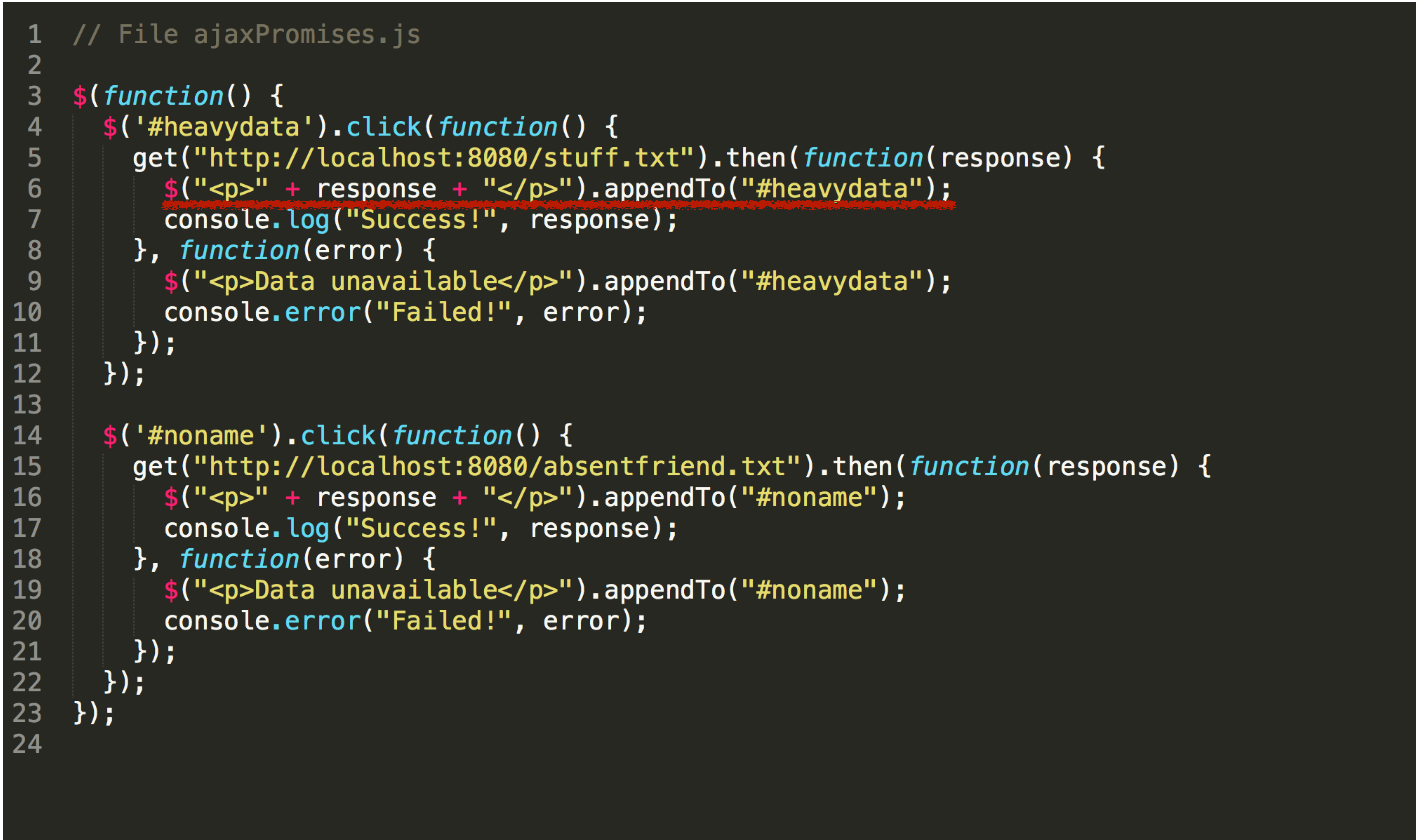
```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});

```

file: ajaxPromises.js



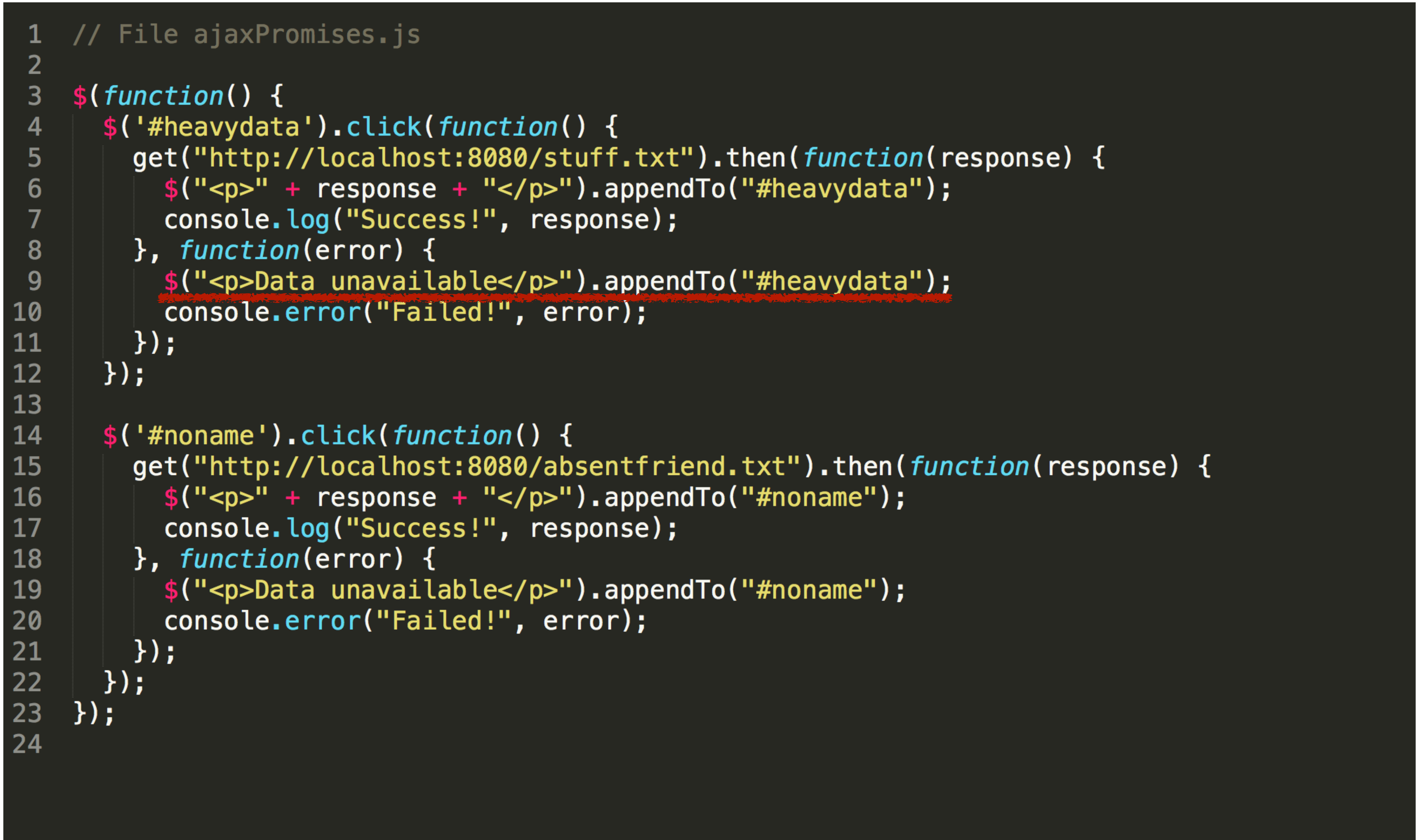
```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

file: ajaxPromises.js



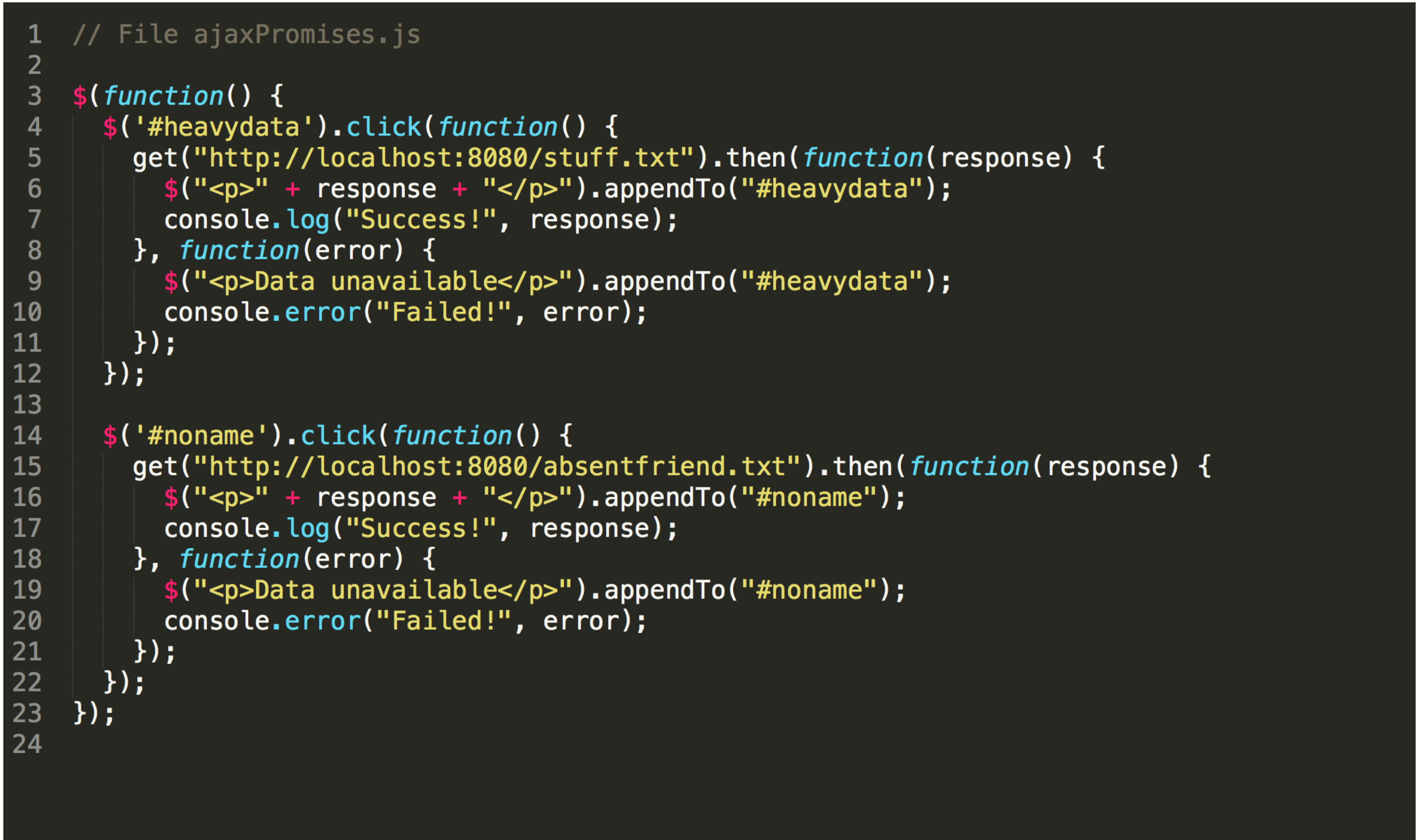
```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});

```

file: ajaxPromises.js



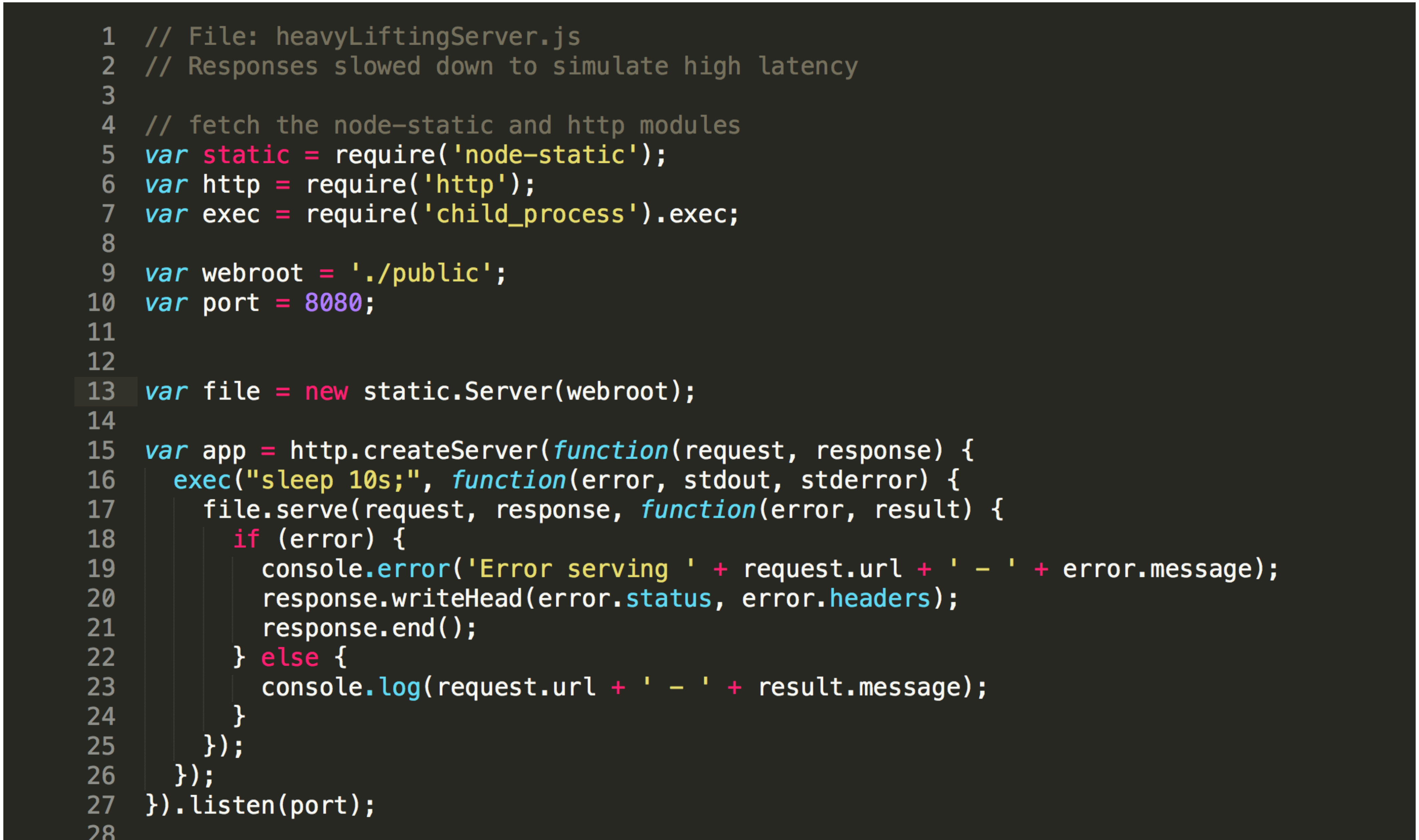
```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

file: ajaxPromises.js



```javascript
// File ajaxPromises.js

$(function() {
  $('#heavydata').click(function() {
    get("http://localhost:8080/stuff.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#heavydata");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    get("http://localhost:8080/absentfriend.txt").then(function(response) {
      $("<p>" + response + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

My heavy lifting server!



```javascript
// File: heavyLiftingServer.js
// Responses slowed down to simulate high latency

// fetch the node-static and http modules
var static = require('node-static');
var http = require('http');
var exec = require('child_process').exec;

var webroot = './public';
var port = 8080;


var file = new static.Server(webroot);

var app = http.createServer(function(request, response) {
  exec("sleep 10s;", function(error, stdout, stderror) {
    file.serve(request, response, function(error, result) {
      if (error) {
        console.error('Error serving ' + request.url + ' - ' + error.message);
        response.writeHead(error.status, error.headers);
        response.end();
      } else {
        console.log(request.url + ' - ' + result.message);
      }
    });
  });
}).listen(port);
```

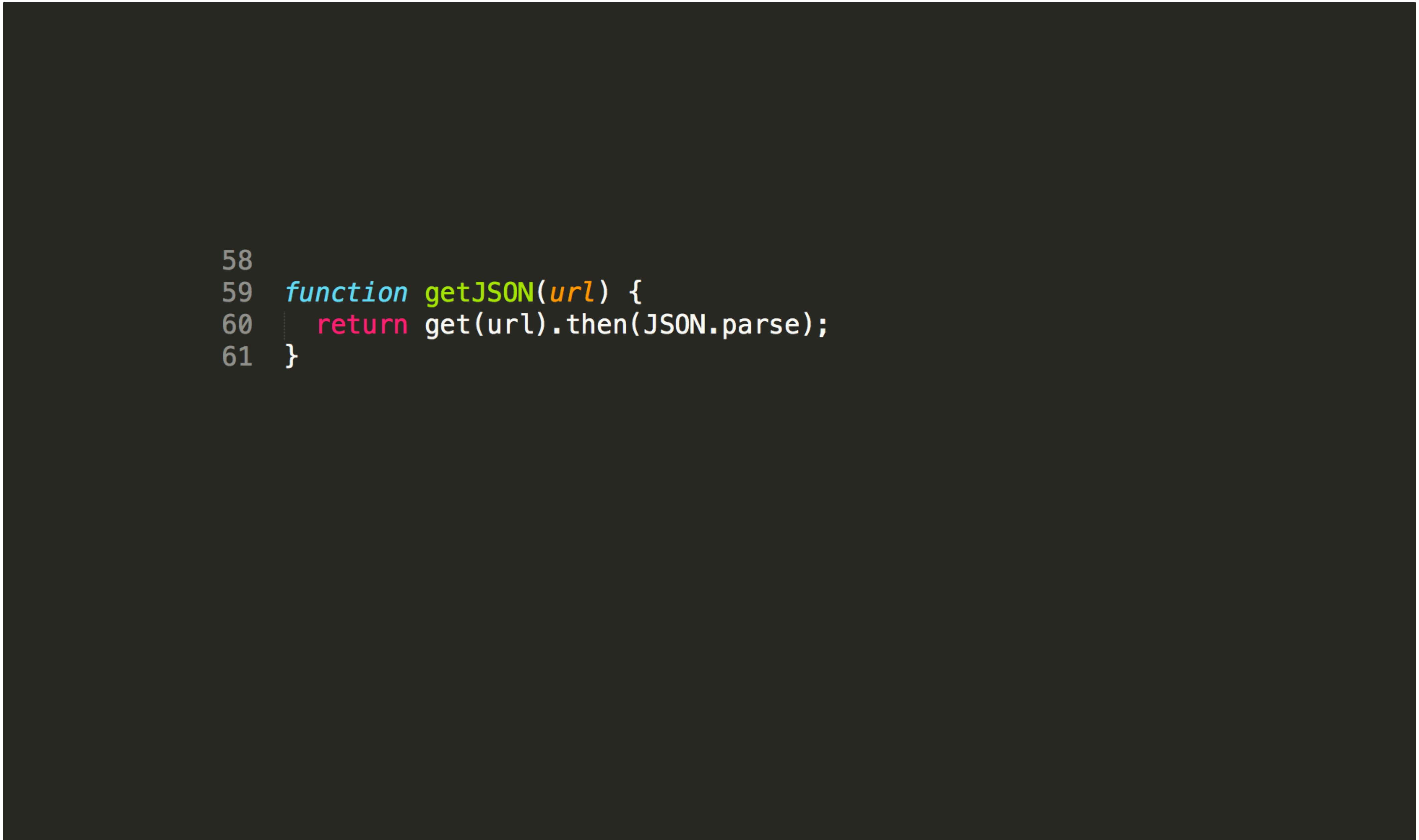
Try me!

(Screen shots of the JSON version start on slide 19)

Other cool things you can do

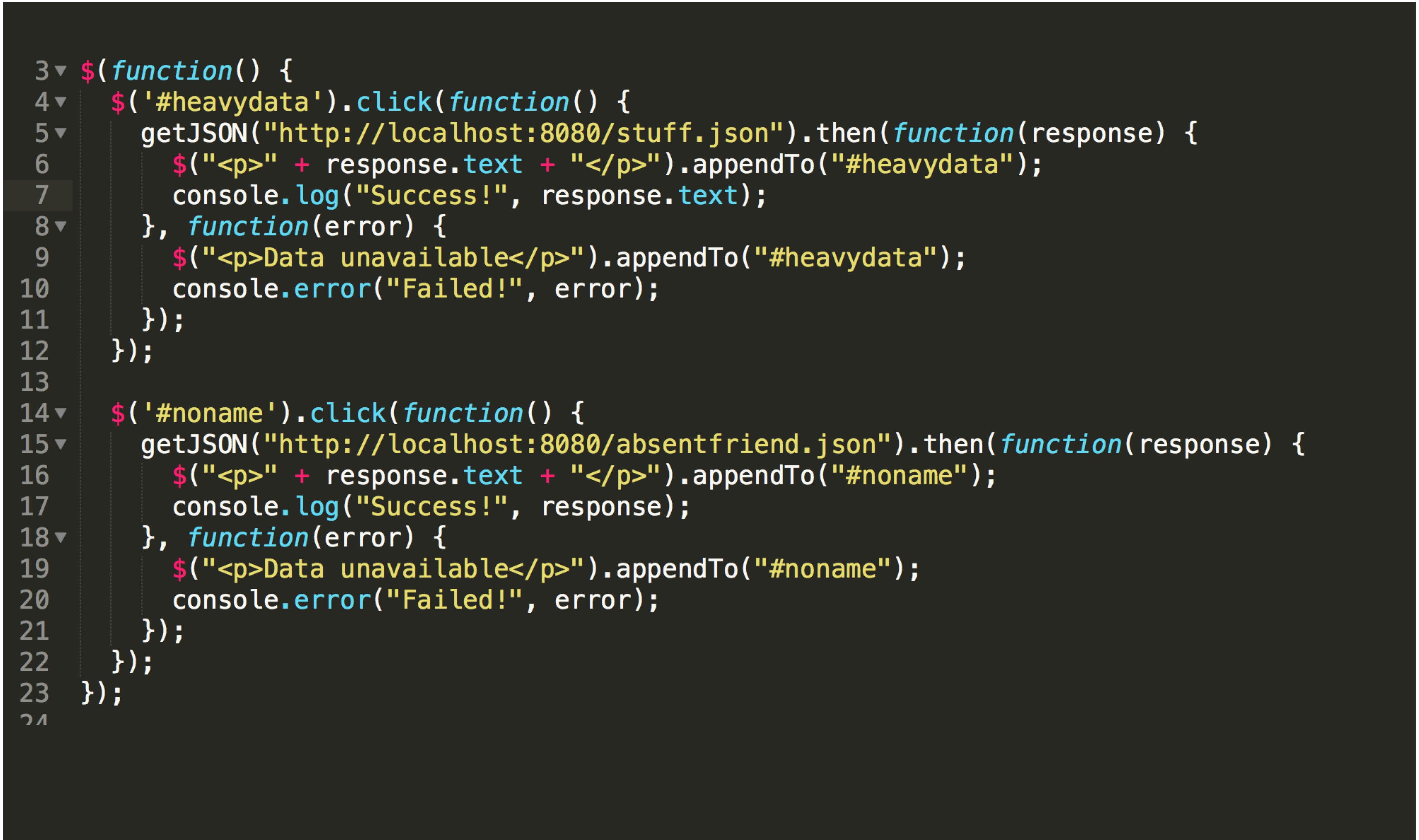
- Chaining “thens”
  - to transform values or run async actions one after another
- Customise the error handling for chained “thens”
- Explicitly sequence the values of parallel promises as they are fulfilled
- …

E.g. Transform a value



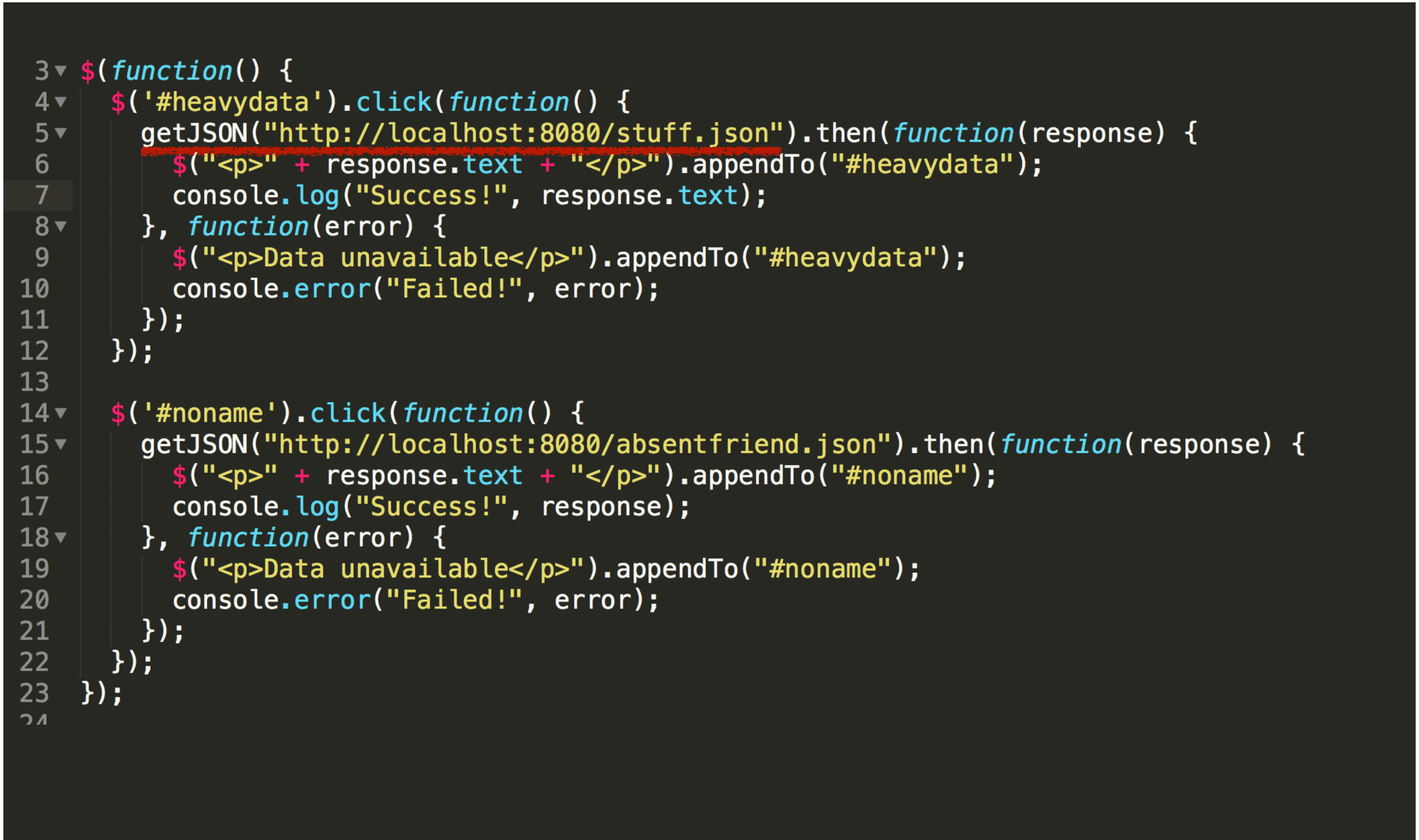
```javascript

function getJSON(url) {
  return get(url).then(JSON.parse);
}
```

And update our document ready function



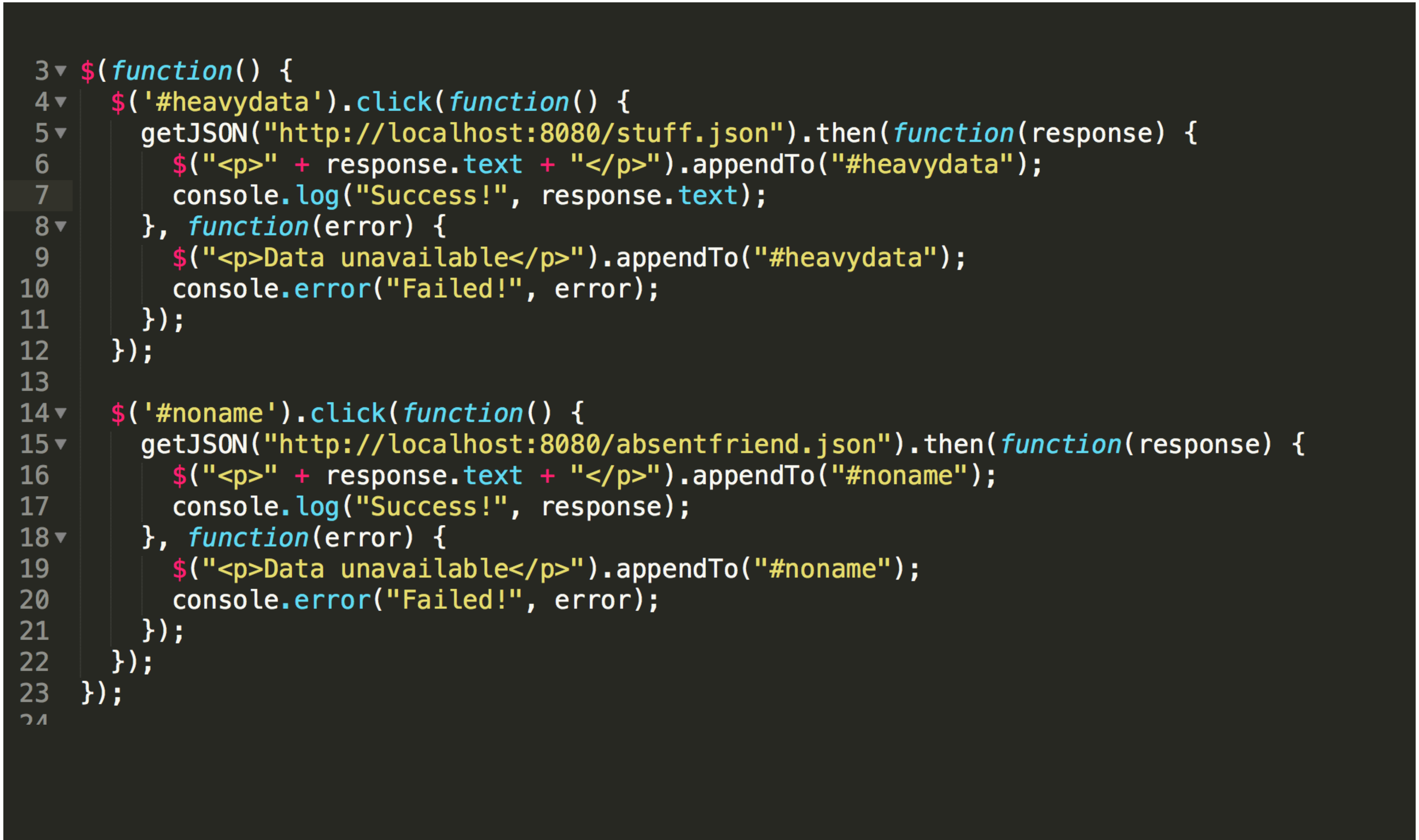
```
$(function() {
  $('#heavydata').click(function() {
    getJSON("http://localhost:8080/stuff.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#heavydata");
      console.log("Success!", response.text);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    getJSON("http://localhost:8080/absentfriend.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

And update our document ready function



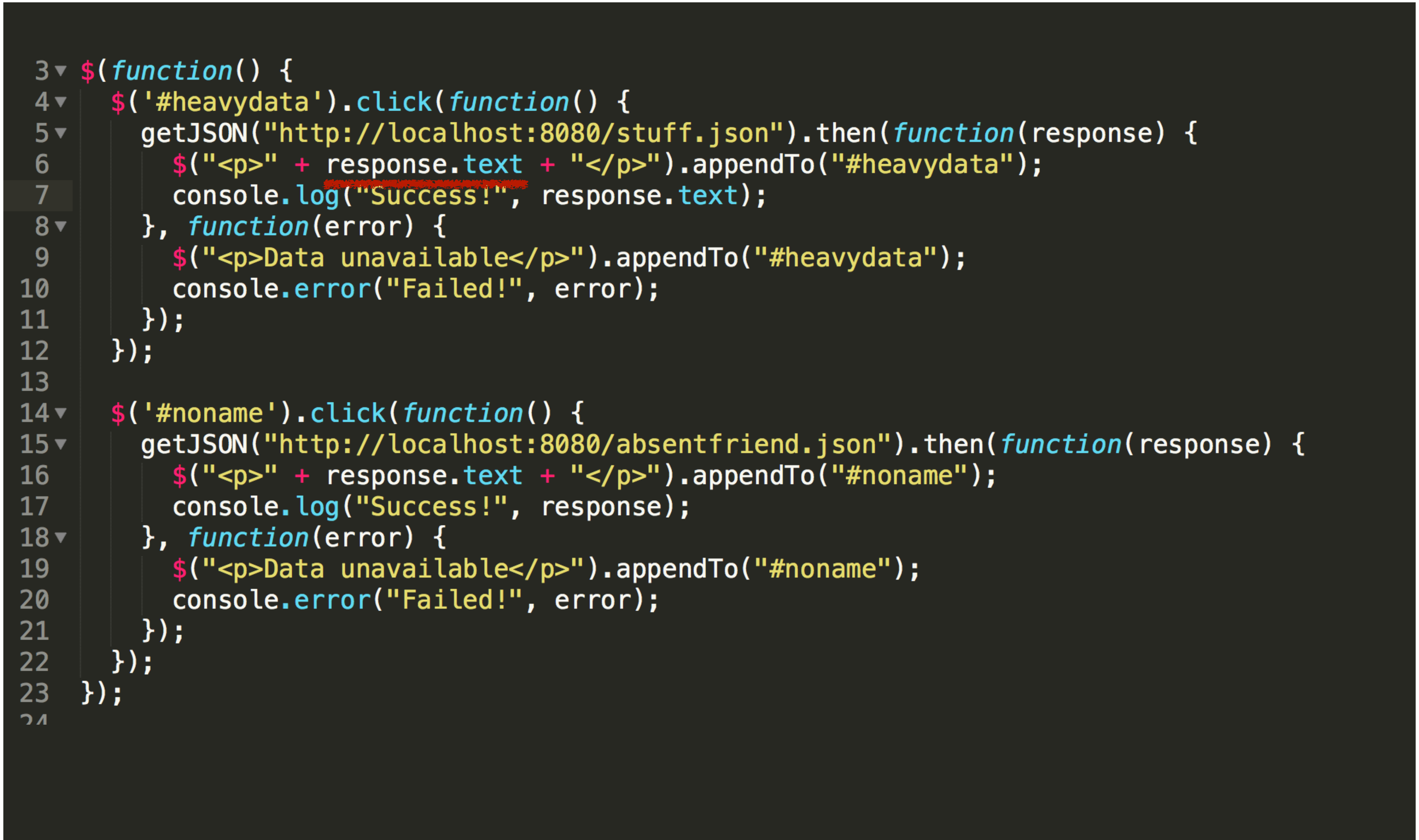
```
$(function() {
  $('#heavydata').click(function() {
    getJSON("http://localhost:8080/stuff.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#heavydata");
      console.log("Success!", response.text);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    getJSON("http://localhost:8080/absentfriend.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

And update our document ready function



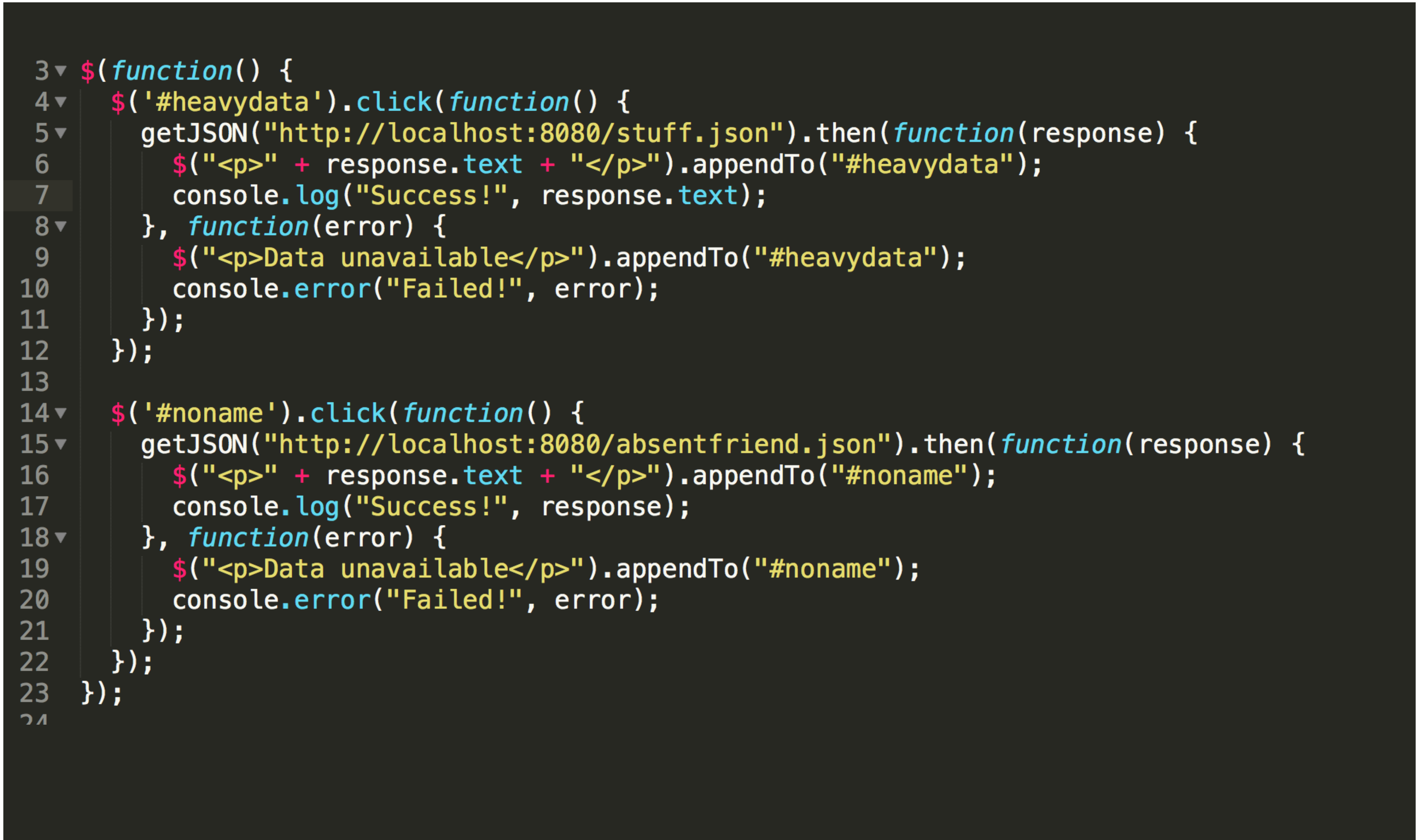
```
$(function() {
  $('#heavydata').click(function() {
    getJSON("http://localhost:8080/stuff.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#heavydata");
      console.log("Success!", response.text);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    getJSON("http://localhost:8080/absentfriend.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

And update our document ready function

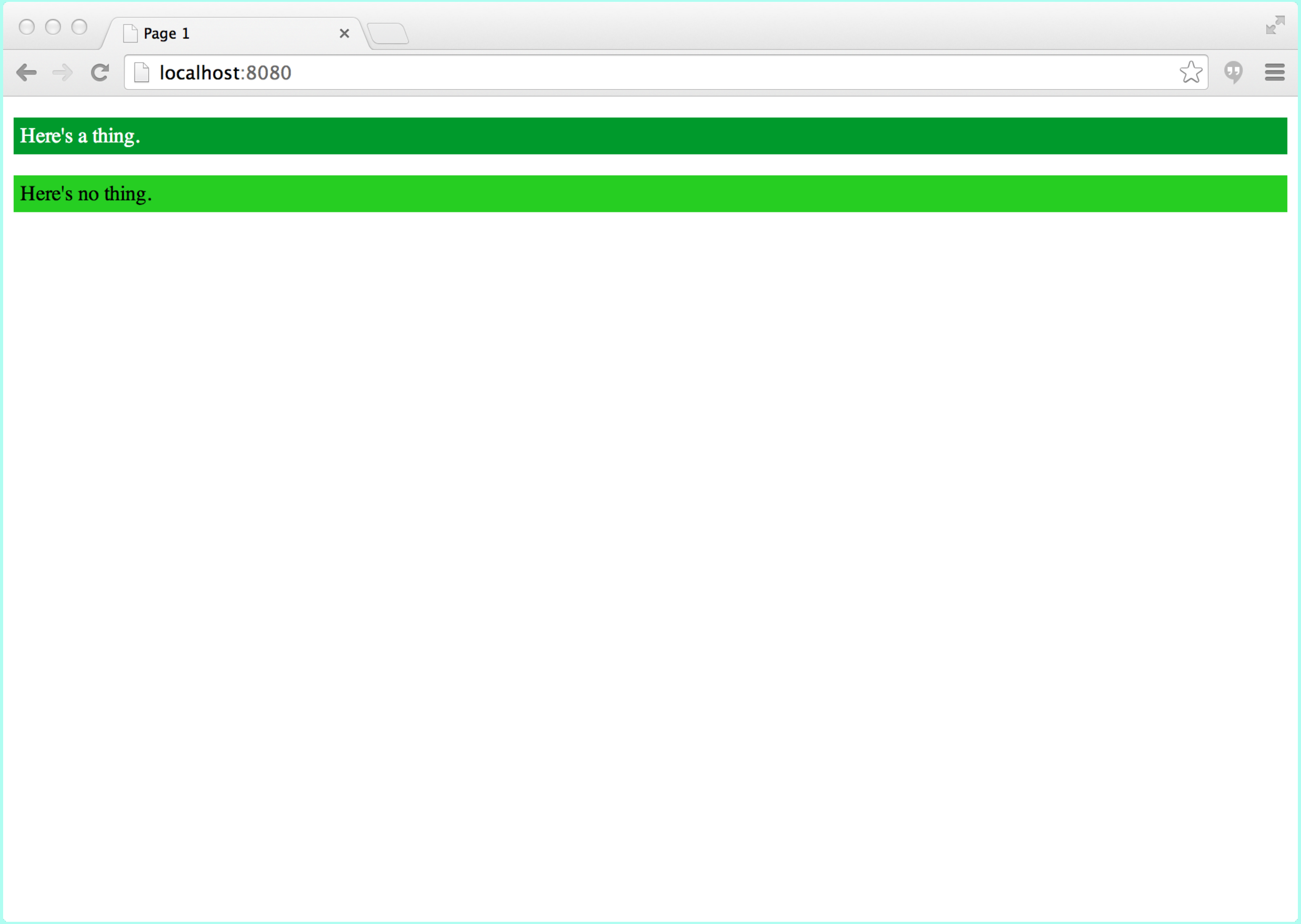


```
$(function() {
  $('#heavydata').click(function() {
    getJSON("http://localhost:8080/stuff.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#heavydata");
      console.log("Success!", response.text);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    getJSON("http://localhost:8080/absentfriend.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```

And update our document ready function

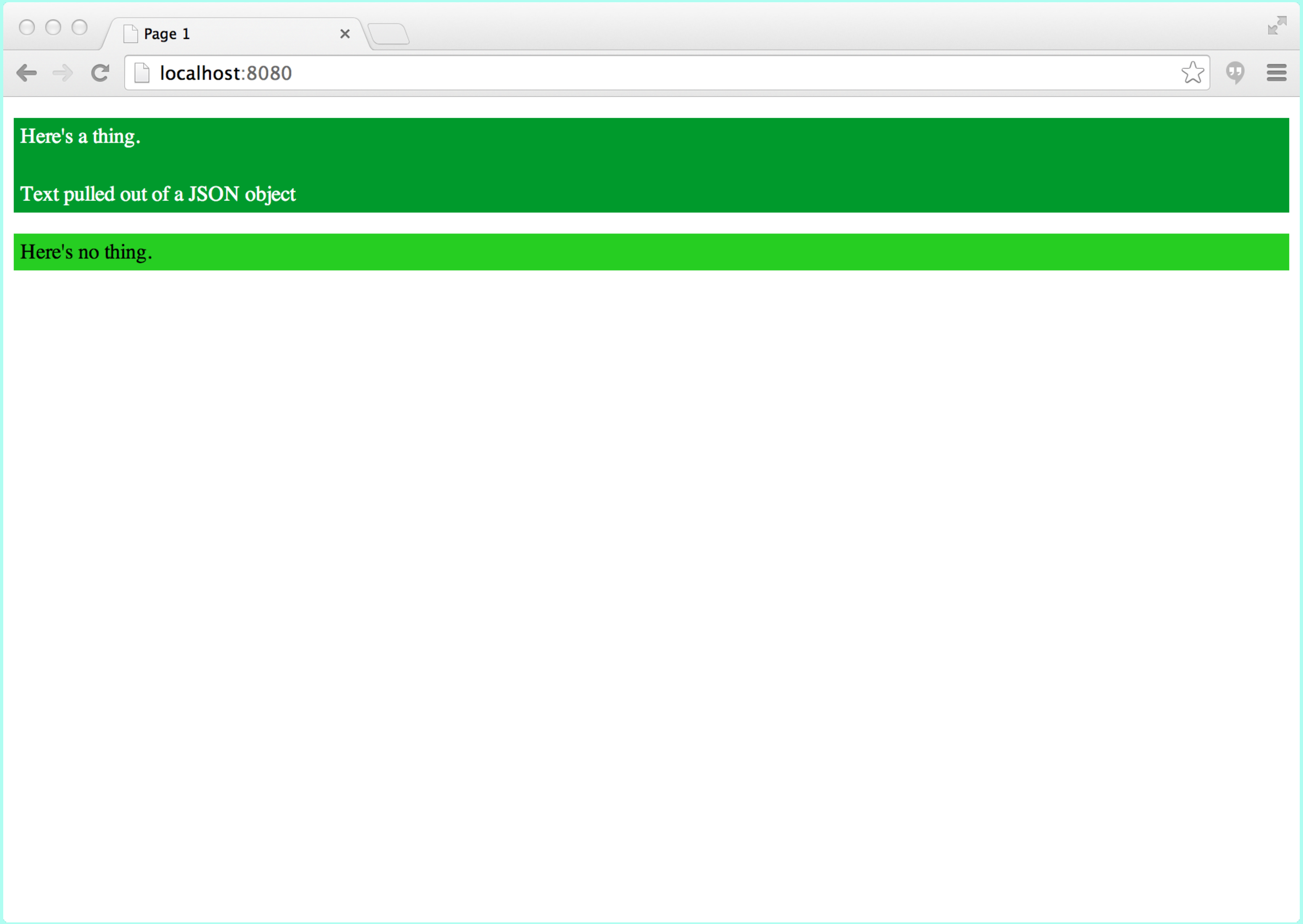


```
$(function() {
  $('#heavydata').click(function() {
    getJSON("http://localhost:8080/stuff.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#heavydata");
      console.log("Success!", response.text);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#heavydata");
      console.error("Failed!", error);
    });
  });

  $('#noname').click(function() {
    getJSON("http://localhost:8080/absentfriend.json").then(function(response) {
      $("<p>" + response.text + "</p>").appendTo("#noname");
      console.log("Success!", response);
    }, function(error) {
      $("<p>Data unavailable</p>").appendTo("#noname");
      console.error("Failed!", error);
    });
  });
});
```



Here's a thing.

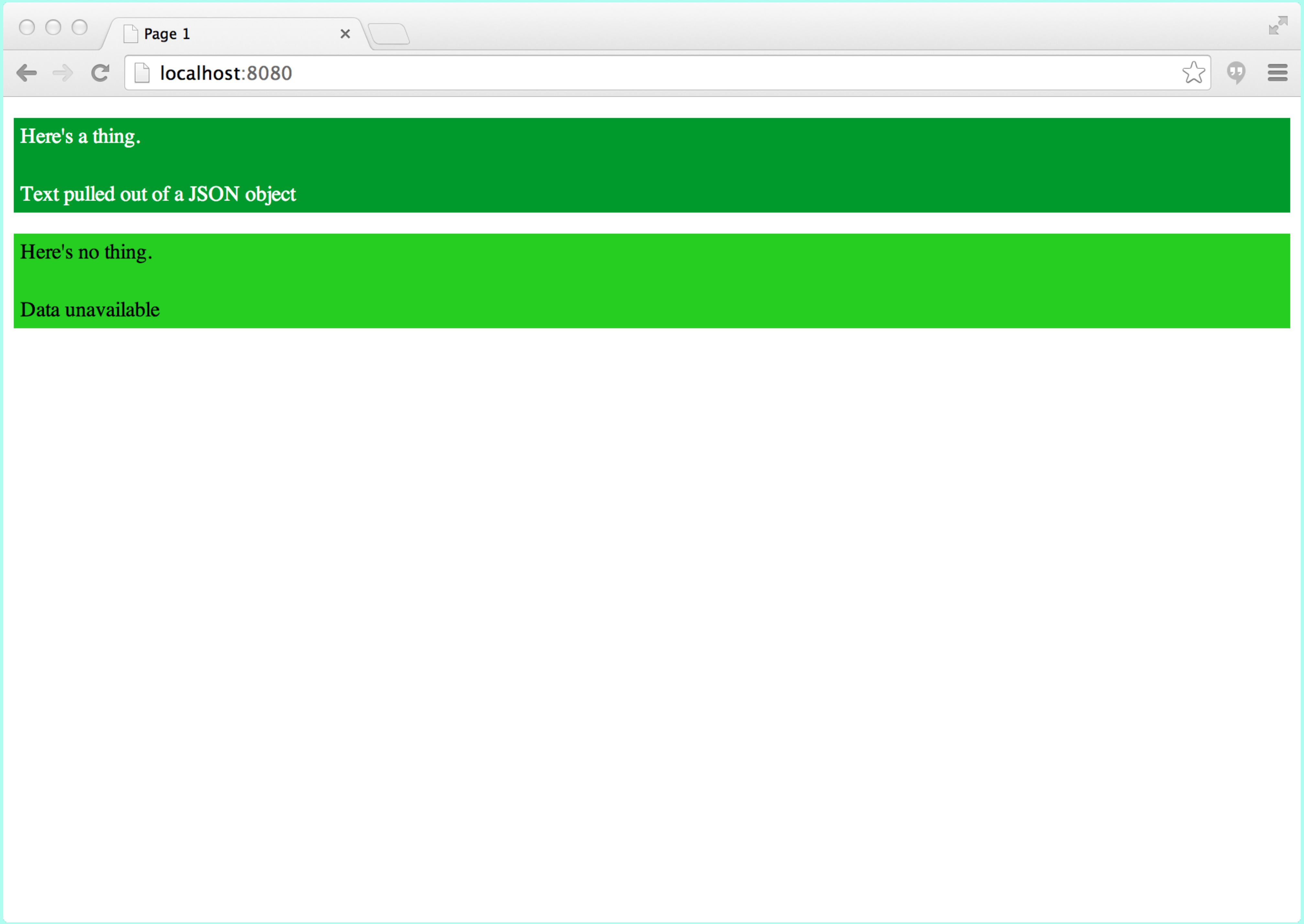
Here's no thing.



Here's a thing.

Text pulled out of a JSON object

Here's no thing.



Here's a thing.

Text pulled out of a JSON object

Here's no thing.

Data unavailable

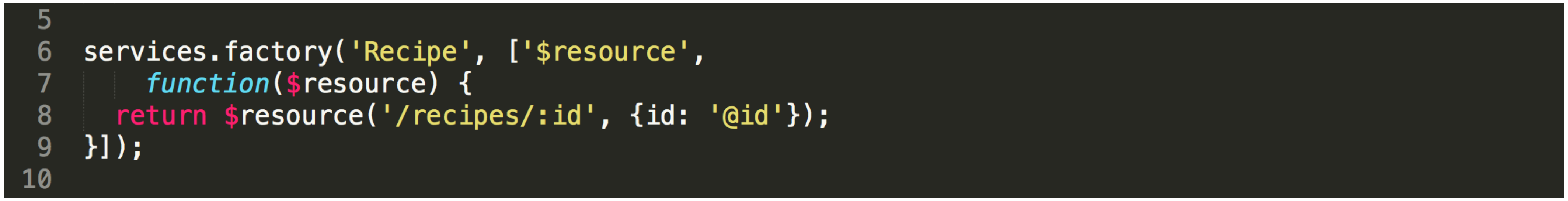
Promises and Angular.js

- Angular has a Promises/A+ compliant implementation baked in: $q
- Any change to scope state due to a promise resolution will be automatically reflected in the View

Source of example:

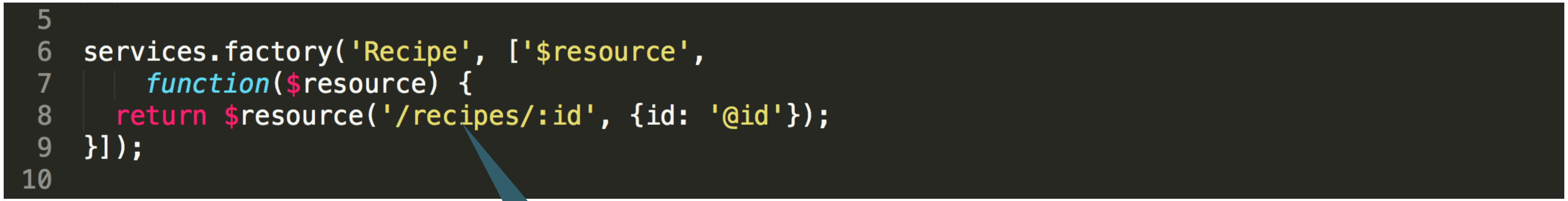
https://github.com/shyamseshadri/angularjs-book.git

Angular Resources



```
services.factory('Recipe', ['$resource',
    function($resource) {
  return $resource('/recipes/:id', {id: '@id'});
}]);
```

Angular Resources



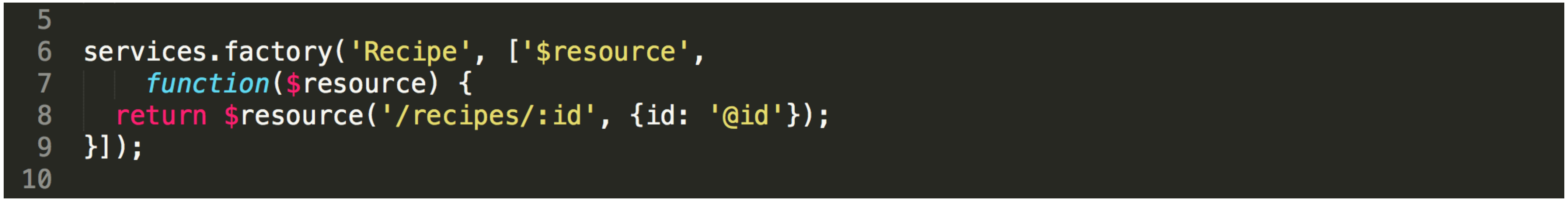
```
services.factory('Recipe', ['$resource',
    function($resource) {
  return $resource('/recipes/:id', {id: '@id'});
}]);
```

Maps (for example)
Recipe.get({id: 15})
onto
/recipes/15

REST over http verbs

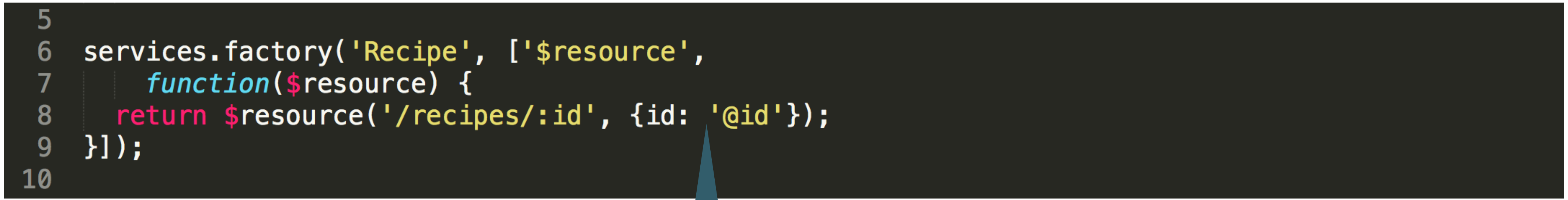
http method	Angular method	url	Comments
Get	Resource.query( )	/resource	
Get	Resource.get({id: ID})	/resource/ID	
Put	Resource.update( )	/resource/ID	Need to create a custom method
Post	Resource.save( )	/resource/ID	
Delete	Resource.delete( )	/resource/ID	

Angular Resources



```
services.factory('Recipe', ['$resource',
        function($resource) {
    return $resource('/recipes/:id', {id: '@id'});
}]);
```

Angular Resources



```
services.factory('Recipe', ['$resource',
    function($resource) {
  return $resource('/recipes/:id', {id: '@id'});
}]);
```

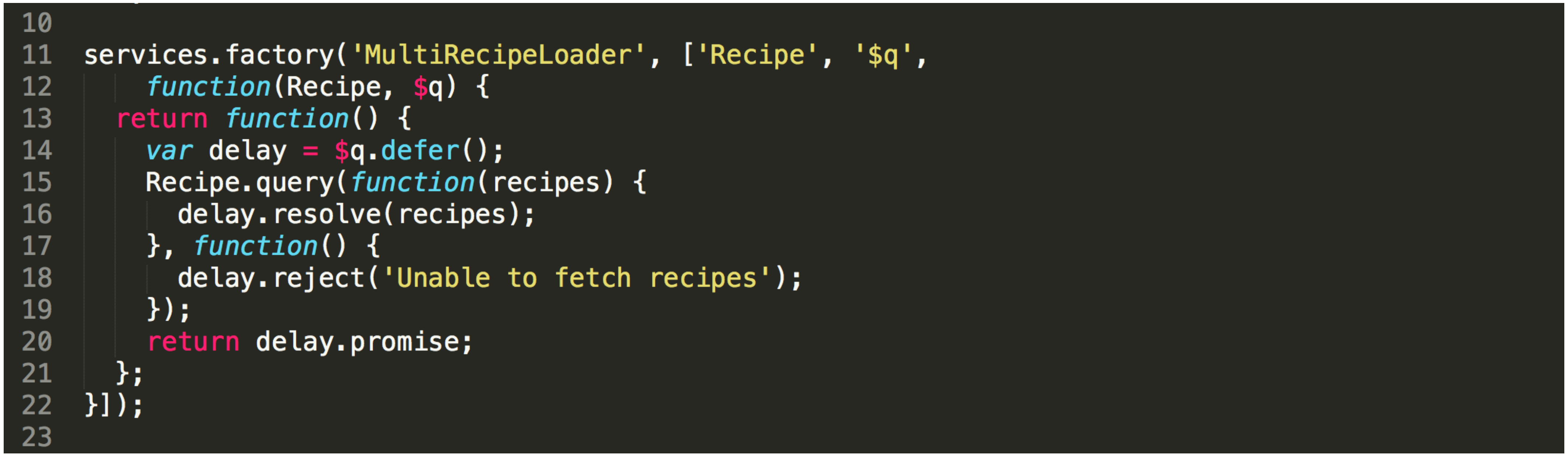
Suppose myRecipe is fully instantiated (with id: 20, say)

```
var recipe = new Recipe(myRecipe);
recipe.save( );
```

will generate a Post request to:

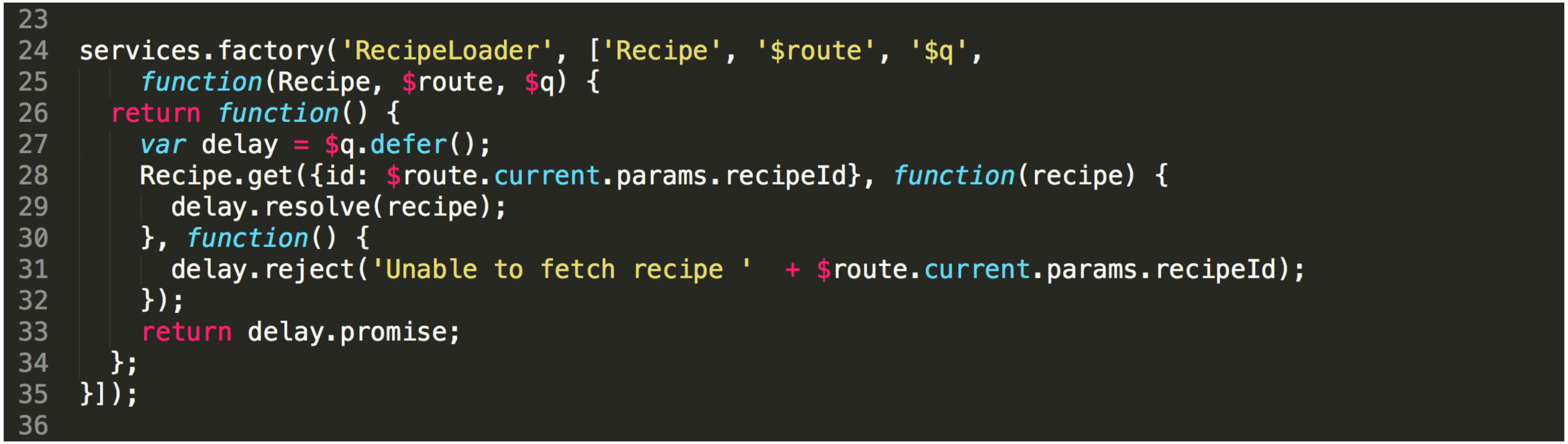
```
/recipes/20
```

promise of recipes

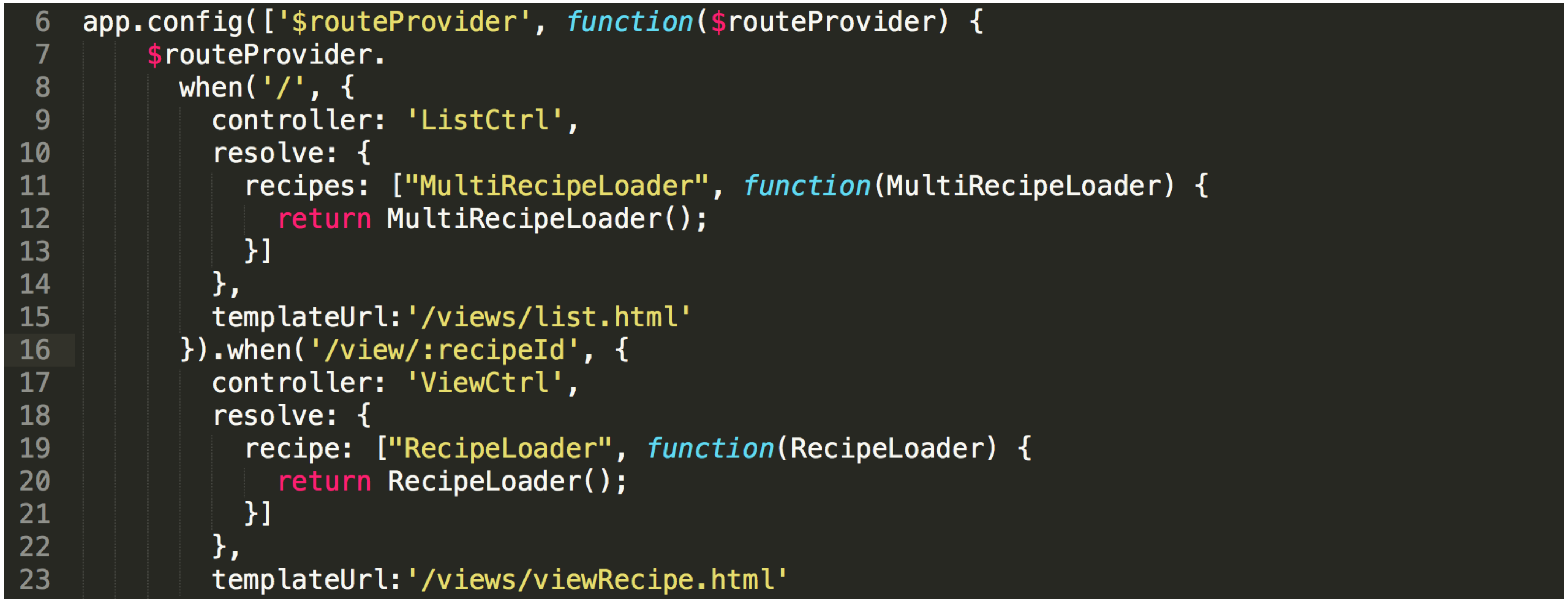


```
services.factory('MultiRecipeLoader', ['Recipe', '$q',
    function(Recipe, $q) {
  return function() {
    var delay = $q.defer();
    Recipe.query(function(recipes) {
      delay.resolve(recipes);
    }, function() {
      delay.reject('Unable to fetch recipes');
    });
    return delay.promise;
  };
}]);
```

promise of a recipe


```

services.factory('RecipeLoader', ['Recipe', '$route', '$q',
    function(Recipe, $route, $q) {
  return function() {
    var delay = $q.defer();
    Recipe.get({id: $route.current.params.recipeId}, function(recipe) {
      delay.resolve(recipe);
    }, function() {
      delay.reject('Unable to fetch recipe '  + $route.current.params.recipeId);
    });
    return delay.promise;
  };
}]);

```

Resolving promises



```
app.config(['$routeProvider', function($routeProvider) {
    $routeProvider.
      when('/', {
        controller: 'ListCtrl',
        resolve: {
          recipes: ["MultiRecipeLoader", function(MultiRecipeLoader) {
            return MultiRecipeLoader();
          }]
        },
        templateUrl:'/views/list.html'
      }).when('/view/:recipeId', {
        controller: 'ViewCtrl',
        resolve: {
          recipe: ["RecipeLoader", function(RecipeLoader) {
            return RecipeLoader();
          }]
        },
        templateUrl:'/views/viewRecipe.html'
```

What we have done

- Promise basics
- For more extensive discussion see:
    - http://www.html5rocks.com/en/tutorials/es6/promises/