JavaScript Introduction

Topics discussed this presentation

- Asynchronous JavaScript & XML (Ajax).
- This presentation based on jQuery Ajax.
- Using a very small subset of available functionality.

Ajax

What is it?

- A technology to manage transmission of data between client and server.
- Generally text-based data.
- Binary data transmission also possible.
- Originally data format Extensible Markup Language (XML).
- JSON now the format of choice.

Ajax

Why use it?

- Once Upon a Time in the Web . . .
  - data request caused whole-page refresh.
- Ajax requests server what it needs,
  - when it needs it and,
  - for exactly where on page it is needed.
  - finding the target with perfect aim.
- This avoids nuisance page flicker and, facilitates greater efficiency.

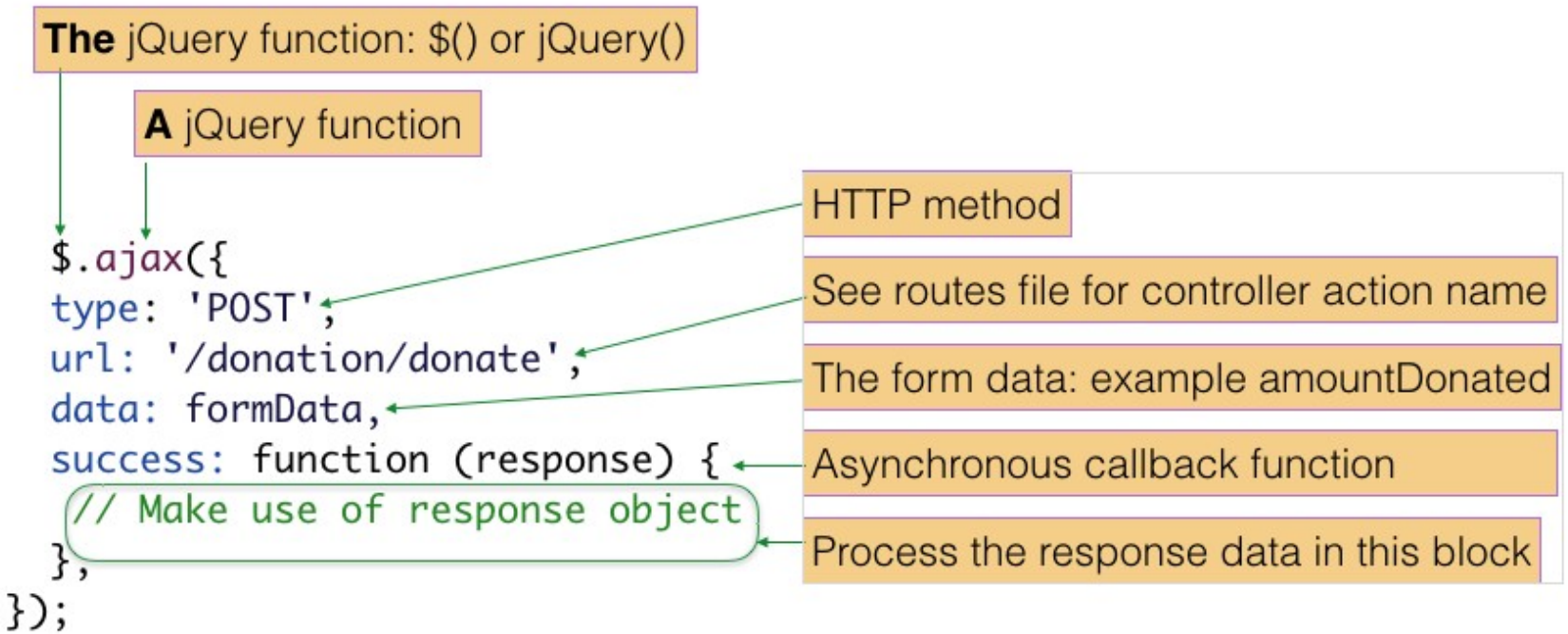
Ajax

Asynchronous communication

- Web page sends HTTP Ajax request.
- User free to continue other page activity.
- Request processed independently.
- Server transmits respose to web page.
- Synchronous communication also possible.

jQuery

Donation ajax call



jQuery

Donation ajax call

```
<form class="ui form" action="/donation/donate" method="POST">
    ...
    ...
</form>
```

```
<form class="ui form">
    ...
    ...
</form>
```

Ajax requires `form` change

jQuery

Donation ajax call

HTML

```
<form class ="ui form segment">
    <input name="candidateEmail" type="hidden">
    <input name="amountDonated" type="hidden">
<div class="ui blue submit button">Donate <i class="add icon"></i></div>
```

JavaScript

```
$('.ui.submit.button').click(function() {
    var formData = $('.ui.form.segment input').serialize();
    $.ajax({
      type: 'POST',
      url: '/donation/donate',
      data: formData,
      success: function(response) {
         console.log("make donation page submitForm response: " + response.progress);
      }
    });
  });
```

Provides routing to controller action donate

The form data: example amountDonated

Asynchronous call back delivers response

Controller

```
JSONObject obj = new JSONObject();
obj.put("progress", getProgress());
renderJSON(obj);
```

Composes and transmits JSON object response

jQuery

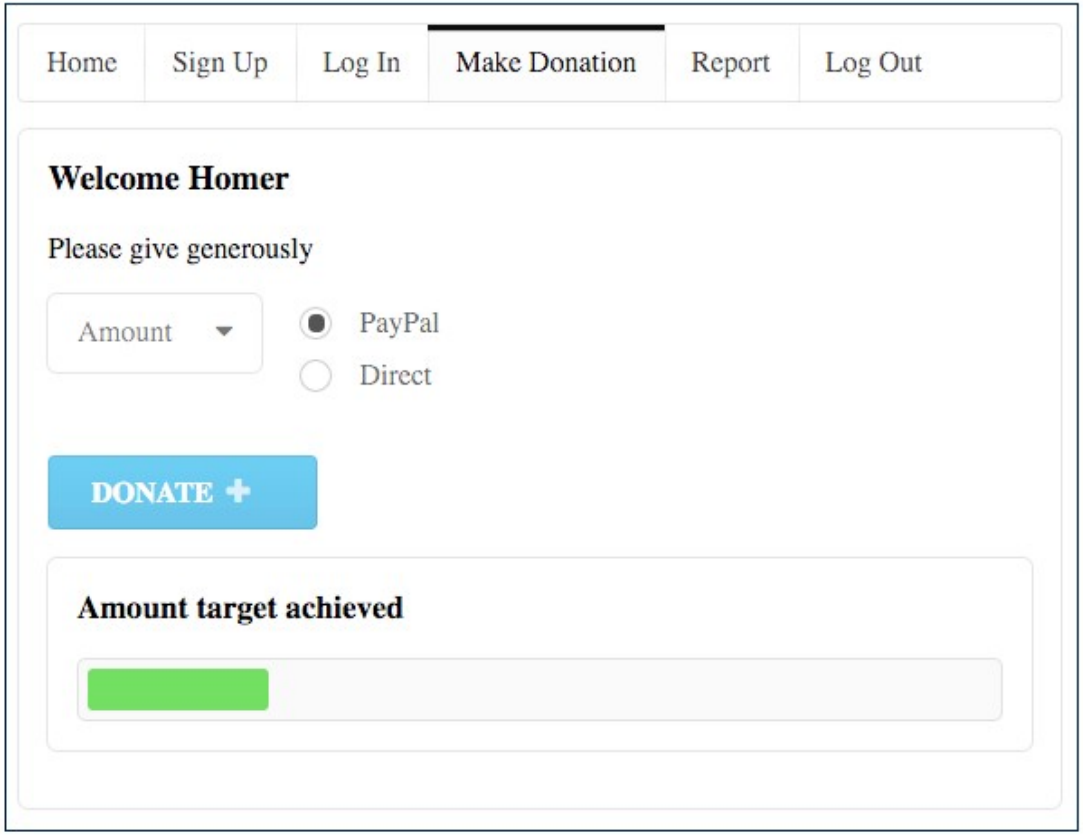
Asynchronous Java and XML (Ajax)

Not using Ajax

- Press Donate button
- Entire page refreshes
- Noticeable flicker

Using Ajax

- Press Donate button
- Only the progress bar affected
- No noticeable flicker



jQuery

Asynchronous Java and XML (Ajax)

Client Side (js)


```
function submitForm() {
  var formData = $('.ui.form.segment input').serialize();
  $.ajax({
    type: 'POST',
    url: '/donation/donate',
    data: formData,
      success: function(response) {
          console.log("make donation page submitForm response: " + response.progress);
          $('.ui.indicating.progress').progress({
              percent: response.progress
            });
          $('#progresslabel').text(response.progress + " % of target achieved to date for candidate " + response.candidate);
      }
  });
}
```

asynchronous call to this url

write progress bar label

```
# routes
POST    /donation/donate            DonationController.donate
```

```
# DonationController

public static void donate(long amountDonated, String methodDonated, String candidateEmail)
{
        ...
        ...
        JSONObject obj = new JSONObject();
        obj.put("progress", progressPercent);
        obj.put("candidate", candidate.firstName + " " + candidate.lastName);
        renderJSON(obj);

    }
}
```

Server Side (java/play)

JavaScript

Summary of Best Practices

- Write code complying with ECMAScript6 (ES6).
- Use quality IDE such as WebStorm.
- Apply styleguide, example Airbnb.
- Use strict mode.
- Avoid use of global variables:
  - Use global abatement or other techniques.
- Do not rely on semicolon insertion.
- Do not use:
  - == (use ===)
  - != (use !==)
- Avoid use of *continue* statement.
- Do not use block-less statements (e.g. following *for, while, if*).