

# Ajax

- Asynchronous JavaScript & XML (Ajax).
- Exploration using Query Ajax.
- Using a very small subset of available functionality.

Introducing  
Ajax



Introduce Ajax via a JQuery example

# 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. . .
  - Http request caused whole-page refresh
  - Complete pages ‘Server Rendered’ and delivered to the browser, replacing the previous page
- Ajax Changed this dynamic. Requests can be issued by client for:
  - Specific information it needs
  - when it needs it,
  - for exactly where on page it is needed. .
- This avoids page flicker and facilitates greater efficiency.

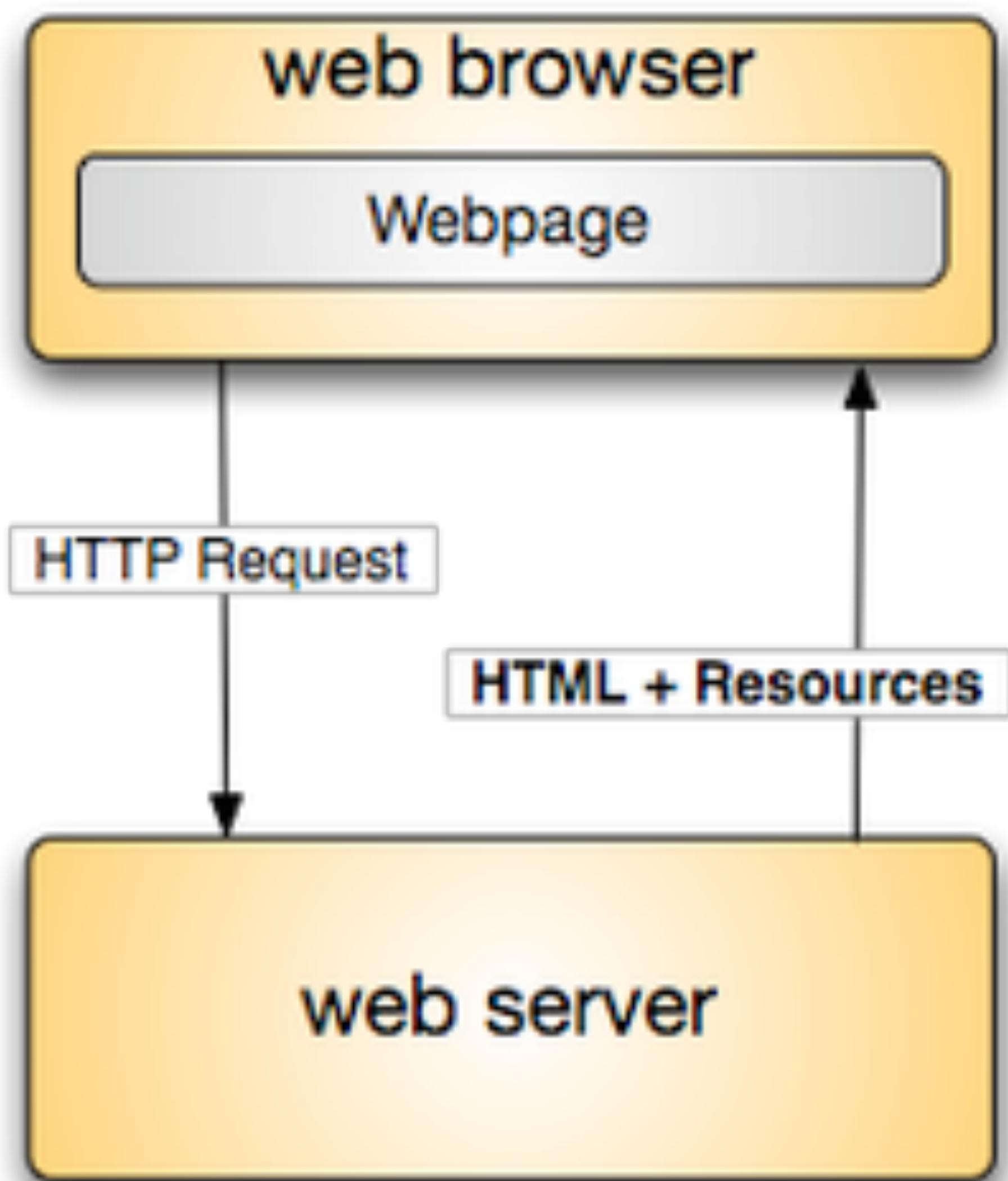
# Ajax

## Asynchronous communication

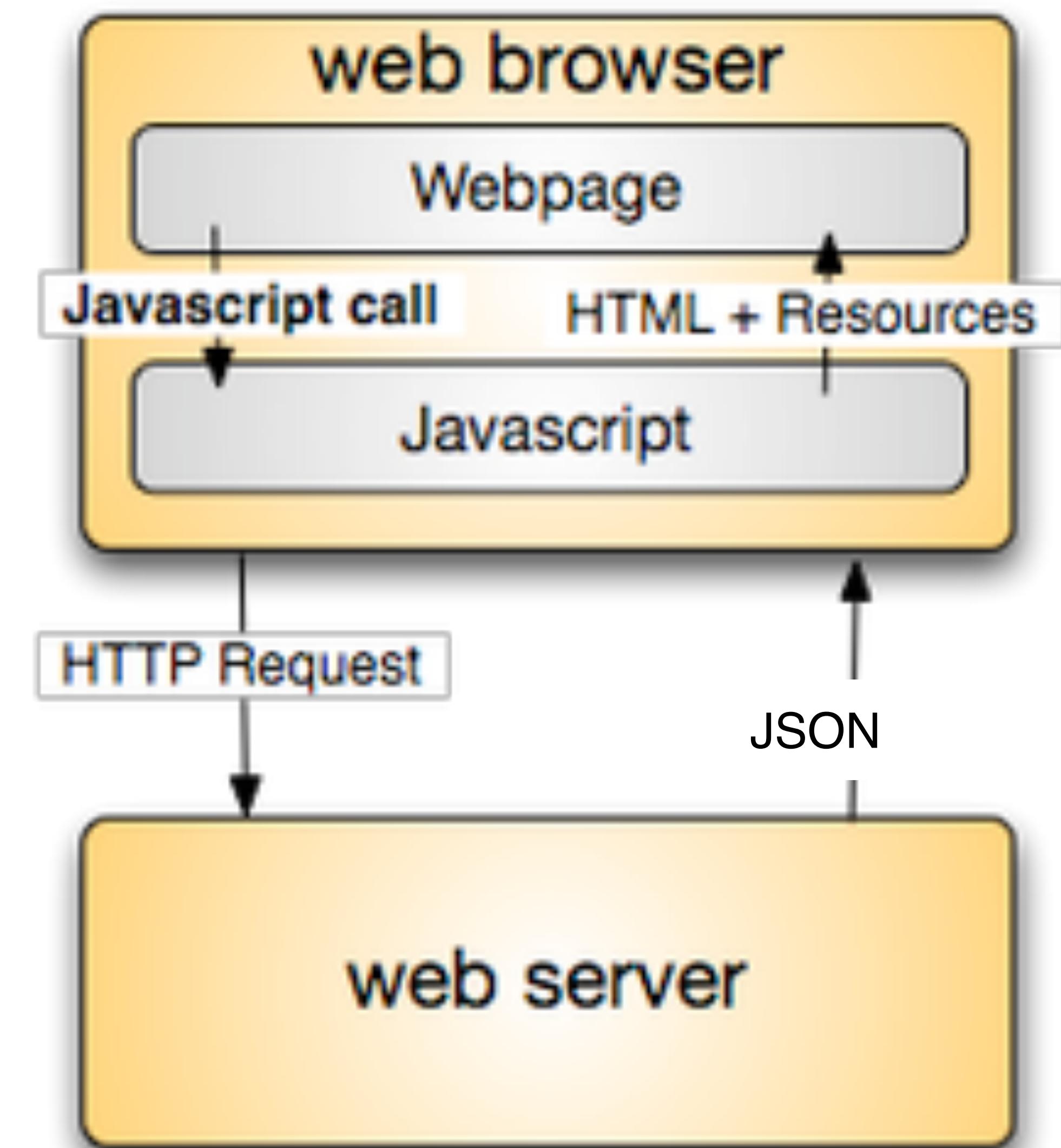
- Web page sends HTTP Ajax request.
- User free to continue other page activity.
- Request processed independently.
- Server transmits response to web page.
- Synchronous communication also possible.



## Traditional web model



## AJAX web model



## The jQuery function: \$(()) or jQuery()

A jQuery function

```
$.ajax({  
  type: 'POST',  
  url: '/donation/donate',  
  data: formData,  
  success: function (response) {  
    // Make use of response object  
  },  
});
```

HTTP method

See routes file for controller action name

The form data: example amountDonated

Asynchronous callback function

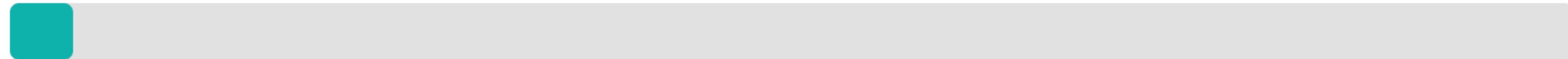
Process the response data in this block

Select Amount ▾

Paypal

Direct

**Donate**



Press Donate should update progress bar + table without page flicker/refresh

Select Amount ▾

- Paypal
- Direct

Donate



```
<form action="/donate" method="POST">
  <div class="ui dropdown" name="amount">
    <input type="hidden" name="amount">
    <div class="text">Select Amount</div>
    <i class="ui dropdown icon"></i>
    <div class="menu">
      <div class="item">50</div>
      <div class="item">100</div>
      <div class="item">1000</div>
    </div>
  </div>
  <div class="grouped inline fields">
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="paypal">
        <label>Paypal</label>
      </div>
    </div>
    <div class="field">
      <div class="ui radio checkbox">
        <input type="radio" name="method" value="direct">
        <label>Direct</label>
      </div>
    </div>
  </div>
  <button class="ui blue submit button">Donate</button>
</form>

<div class="ui divider"></div>

<div class="ui teal progress" data-percent="${progress}" id="mainprogress">
  <div class="bar"></div>
</div>
```

# jQuery

Donation ajax call

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

```
<form action="/donate">  
    ...  
</form>
```

Ajax requires Form change

# Ajax Function in the Client

```
$('.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);  
        }  
    });  
});
```

JavaScript

Provides routing to controller action donate

The form data: example amountDonated

Asynchronous call back delivers response

# Play/Java Web Application - in the Server

```
# 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);
}
```

renderJSON (instead of html)



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

The code defines a `submitForm` function. It first serializes form data from a specific class. Then it performs an asynchronous POST request to the URL `/donation/donate`, passing the serialized form data as data. In the `success` callback, it logs the response progress to the console, updates a progress bar with the new percentage, and updates a label with the current progress and the candidate's name.

progress bar asynchronously updated

asynchronous call to this url

write progress bar label

update progress bar in client

## Not using Ajax

Press Donate button

Entire page refreshes

Noticeable flicker

## Using Ajax

Press Donate button

Only the progress bar affected

No noticeable flicker

Home Sign Up Log In Make Donation Report Log Out

Welcome Homer

Please give generously

Amount ▾

PayPal  
 Direct

**DONATE +**

Amount target achieved