software studio

asynchronous calls: examples

Daniel Jackson

timers

```
var alert_timers = function () {
    setTimeout(function () {alert("page about to expire!");}, 2000);
    setInterval(function () {alert("take a typing break!");}, 4000);
};
```

- asynchronous event due to timeouts
- note that alert is modal (and synchronous)

loading HTML

```
var loading = function () {
    var url = 'http://localhost:3000/status';
    $('#status').html('<img src="animated.gif"/>').load(url);
};
```

client side

```
  # GET /status
  def status
    sleep 2
    render :text => "status : UP"
  end
```

server side

- **method chaining** idiom: element.html().load()
- animated GIF is replaced when callback executes
- *sleep* 2 simulates network delay

continual refresh

```
var continual = function () {
    var url = "http://localhost:3000/status";
    setInterval(
        function () {
            $('#status').html('<img src="animated.gif"/>').load(url);
        }, 5000);
};
```

client side

```
  # GET /status
  def status
    sleep 2
    render :text => "status : UP"
  end
```

server side

› every 5s, gets status and displays it

getting a string, passing to callback

```javascript
var simple_get = function () {
    var url = 'http://localhost:3000/debug';
    $.get(url, function (d) {alert("Server says: " + d);});
};
```

client side

```ruby
  # GET /debug
  def debug
    render :text => "Sorry! This app has failed catastrophically!"
  end
```

server side

› server returns string
› callback simply calls alert

sending an object

```
var get_with_send = function () {
    var url = 'http://localhost:3000/welcome';
    var data = {user: "Daniel"};
    $.get(url, data, function (d) {alert(d);});
};
```

client side

```
  # GET /welcome
  def welcome
    user = params[:user]
    render :text => "Welcome, " + user
  end
```

server side

- client passes Javascript object
- because call is $.get, appended as query string on url
- server returns string

getting a JSON object

```
var get_json_status = function () {
    var url = 'http://localhost:3000/status.json';
    $.ajax({ url : url,
             success : function (result) {
               $('#status').html('status: ' + result.status +
                                 ' at: ' + result.time);
             },
             type : 'GET', dataType : 'json'
           });
};
```

client side

```
  # GET /status
  def status
    sleep 2
    respond_to do |format|
      format.html { render :text => "status: UP" }
      format.json { render :json => {:status => "UP", :time => "2pm"} }
    end
  end
```

server side

- **$.ajax:** most basic AJAX method in jquery
- server returns JSON encoded object
- JQuery's .ajax decodes, passes JS object to callback

AJAX in Rails form

index.html.erb

```
<p>Please type your name.</p>

<%= form_tag("/register", :method => "get", :remote => true) do %>
  <%= label_tag(:name, "Your name:") %>
  <%= text_field_tag(:name) %>
  <%= submit_tag("Submit") %>
<% end %>
```

controller

```
def register
  @name = params[:name]
  respond_to do |format|
    format.js
    format.html { render :text => 'Not an AJAX call, ' + @name }
  end
end
```

register.js.erb

```
alert('Now I know your name, <%= @name %>.');
```

- *:remote => true* for Ajax requests
- also useful: *link_to_remote* method