Partials

Objectives:

- Discuss rendering partial HTML
- Understand how partials can be useful

It's time to combine these concepts of DOM manipulation and making AJAX requests.

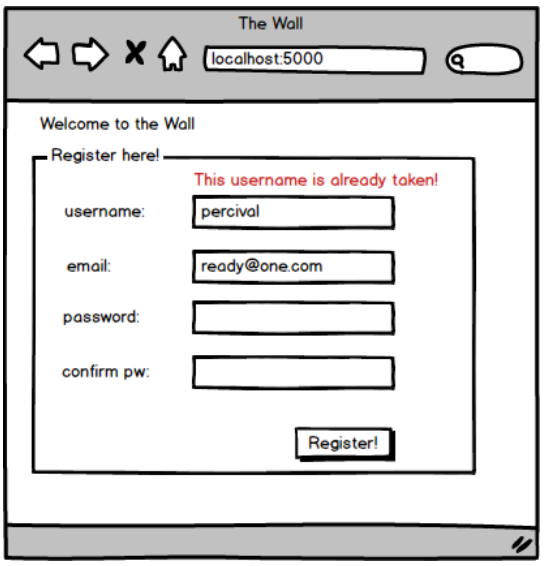
With AJAX, the *browser's JavaScript interpreter* sends an HTTP request. The response can then be used to manipulate the DOM. That means the server does not have to render the entire page of HTML again and the browser does not reload! So if the server is not rendering the entire page of HTML, then what is it sending as a response?

We will be covering two different responses that the server could send back: **partials** and **JSON.** We'll begin with partials.

Partials are snippets of HTML

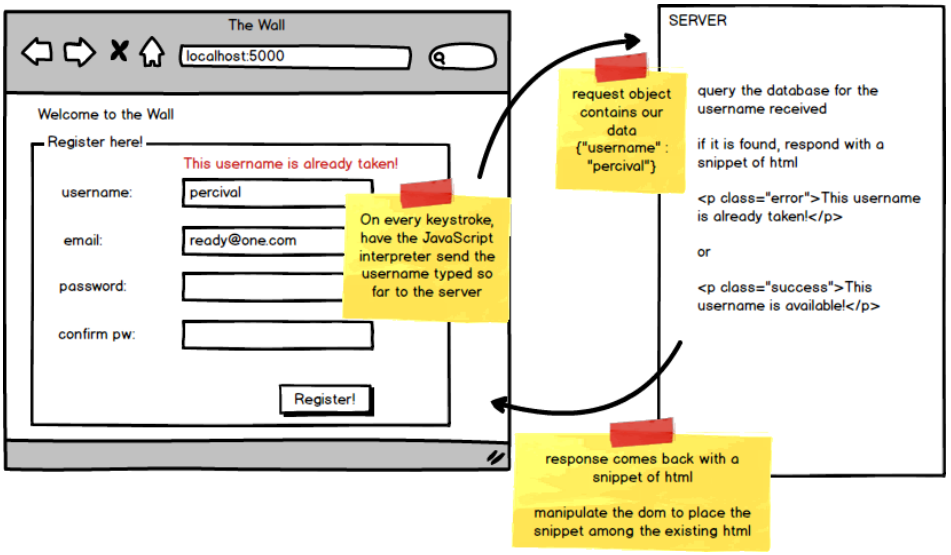
A partial is a small snippet of HTML. When we make a partial, we won't even include the usual meta data we typically place at the top of an HTML document. We do not need any of this meta data because we are going to incorporate the HTML snippet into the document already on the client, which already has the meta data needed.

To understand why we would want a partial, consider the scenario where a user is filling out a registration form. The application expects a unique username. As the user is filling out the form, before it's even submitted, wouldn't it be nice if we could notify the user if the username typed in is taken already? It would save the user some time to know that right away, rather than having to wait until after the form is filled out and submitted.



Notice another benefit of having this error message pop up without having to reload the entire page: We could leave everything else on the page alone, including all the inputs that the user already filled out. If you were ever frustrated about losing user input when reloading a page with flash messages, this should be of great interest to you. We may manipulate the DOM just enough to include this one desired piece of HTML with our error message.

To make this happen, we could have our browser listen for every keystroke that takes place in the username field. With each keystroke, we'll have the JavaScript interpreter send a request to the server to query the database for the username that the user has typed so far. If we find the username in our database already, we'll have the server respond with just a `<p>` tag holding our error message. Otherwise, we may respond with a `<p>` tag containing a message that the username is available. Once the `<p>` tag arrives on the client side, we'll manipulate the DOM to include this `<p>` tag.



Here's the code to make this happen! Some basics, such as stylesheets and importing jQuery, are removed for brevity.

templates/wall.html

```
<html>
    <head>
        <script src="{{ url_for('static', filename='wall.js') }}"></script>
    </head>
    <form action="/register" method="post" id="regForm">
        <div id="usernameMsg"></div>   <!-- notice the empty div reserved for our message --
>
        <input id="username" type="text">
        <button type="submit">Submit</button>
    </form>
</html>
```

static/wall.js

```
$(document).ready(function(){
    $('#username').keyup(function(){
        var data = $("#regForm").serialize()   // capture all the data in the form in the va
riable data
        $.ajax({
            method: "POST",   // we are using a post request here, but this could also be do
ne with a get
            url: "/username",
            data: data
        })
        .done(function(res){
            $('#usernameMsg').html(res)  // manipulate the dom when the response comes back
        })
    })
})
```

server.py

```
@app.route("/username", methods=['POST'])
def username():
    found = False
    mysql = connectToMySQL('ajaxWall')        # connect to the database
    query = "SELECT username from users WHERE users.username = %(user)s;"
    data = { 'user': request.form['username'] }
    result = mysql.query_db(query, data)
    if result:
        found = True
    return render_template('partials/username.html', found=found)  # render a partial and re
turn it
    # Notice that we are rendering on a post! Why is it okay to render on a post in this sce
nario?
    # Consider what would happen if the user clicks refresh. Would the form be resubmitted?
```

Our partial, shown below, is kept in a partials folder nested inside our templates folder. For this code snippet, *we did not remove anything!* What you see is the entire file. Notice the lack of any meta data - it is not needed because it will become part of wall.html.

templates/partials/username.html

```
{% if found == True %}
<p class="error">Username has been taken.</p>
{% endif %}
{% if found == False %}
<p class="success"> This username is available</p>
{% endif %}
```

Here's a review of the different pieces we've talked about so far: