ECEN 4213
Embedded Computer System Design

Flask, jQuery and Ajax

Instructor: Dr. Weihua Sheng, weihua.sheng@okstate.edu

TA: Zhanjie Chen, zhanjie.chen@okstate.edu

Claudia Pauyac, cpauyac@okstate.edu

Fall 2025

SCHOOL OF **ELECTRICAL** AND
COMPUTER ENGINEERING

I. Flask

II. jQuery and Ajax

I. Flask - Overview

- **What is Flask?**

  Flask is a web framework that provides libraries to build lightweight web applications in Python. It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

- **What is WSGI?**

  It is an acronym for web server gateway interface which is a standard for Python web application development. It is considered as the specification for the universal interface between the web server and web application.

- **What is Jinja2?**

  Jinja2 is a web template engine which combines a template with a certain data source to render the dynamic web pages.

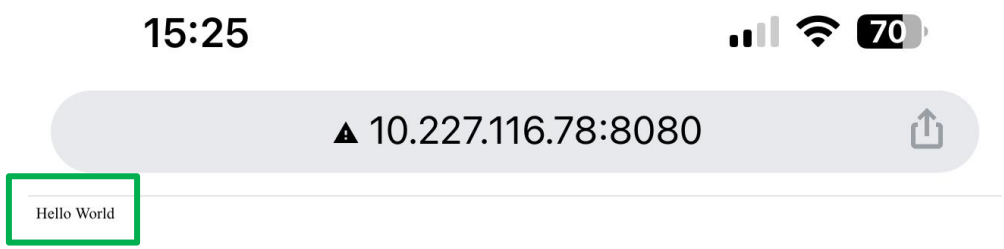
Note: To install Flask, open a terminal and run "pip install flask"
https://www.javatpoint.com/flask-tutorial

Example

Save the following code in *start_flask.py* and run it using *python3 start_flask.py*:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def  hello_world():
        return "Hello World"

if  __name__ == '__main__':
        IP = "10.227.XX.XX" # fill in your IP.
        PORT = "8080"
        DEBUG = True
        app.run(host = IP, port = PORT, debug = DEBUG)
```

Use your cellphone to open the link: *http://10.227.XX.XX:8080*. Make sure your cell phone is connected to the same Wi-Fi. You will be able to see the webpage shown in the right figure.



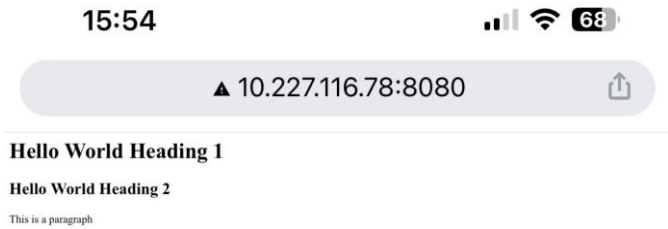
```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/') # '/' URL is bound with hello_world() function
def hello_world():
        return "Hello World"

@app.route('/html_render')
def html_render():
   return render_template('hello.html')


if __name__ == '__main__':
        IP = "10.227.XX.XX" # fill in your IP.
        PORT = "8080"
        DEBUG = True
        app.run(host = IP, port = PORT, debug = DEBUG)
```

hello.html, place it in the ***templates*** folder

```
<!DOCTYPE html>
<html>

  <head>
      <title>Hello World</title>
  </head>

  <body>
    <h1>Hello World Heading 1</h1>
    <h2>Hello World Heading 2</h2>
    <p>This is a paragraph</p>
  </body>

</html>
```

The Jinja2 template engine is used by Flask, where the the *render_template()* function can render the HTML file.

Open the link: *http://10.227.xx.xx:8080/html_render*.
You will be able to see the webpage shown in the right figure.



I. Flask

II. jQuery and Ajax

I. jQuery - Overview

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish and wraps them into methods that we can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

Takeaway: jQuery is like a toolbox that gives you shortcuts for common webpage tasks.

https://www.w3schools.com/jquery/default.asp

Get started

To use jQuery, we need to add it to the HTML code:

- Download the jQuery library from *jQuery.com*
  `<script src="jquery-3.6.0.min.js"></script>`
- Or include jQuery from a CDN (Content Delivery Network), like Google
  `<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>`

jQuery syntax

- The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s)
- Basic syntax is: *$(selector).action()*
  - A *$* sign to define/access jQuery
  - A *(selector)* to "query (or find)" HTML elements
  - A jQuery *action()* to be performed on the element(s)

Examples:
- *$(this).hide()* - hides the current element
- *$("p").hide()* - hides <p> elements
- *$(".test").hide()* - hides all elements with class="test"
- *$("#test").hide()* - hides the element with id="test"

Selector

- jQuery selectors allow us to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
- All selectors in jQuery start with the dollar sign and parentheses: *$()*.

```
<script>
$(document).ready(function(){
        $("button").click(function(){
                $("p").hide();
                });
});
</script>
```

Here, all jQuery methods are inside a document ready event. This is to prevent any jQuery code from running before the document is finished loading (is ready).

```
<script>
$(function(){
        $("button").click(function(){
                $("p").hide();
                });
});
</script>
```

This is a shorter method for the document ready event. They have the same effect.

Event and Effect

```
<script>
$(document).ready(function(){
        $("button").click(function(){
                $("p").hide();
                });
});
</script>
```

- $("button") selects the button element
- **click** is a mouse event
- **hide()** is a jQuery method/effect
- The logic of the code is: when the button element is clicked, all <p> elements will be hidden

Some events:

Mouse Events	Keyboard Events	Form Events	Document /Window Events
click	keypress	submit	load
doubleclick	keydown	change	resize
Mouseenter	keyup	Focus	scroll
mouseleave		blur	unload

Some effects:

- Hide
- Show
- Toggle
- Slide
- Fade
- Animate

AJAX

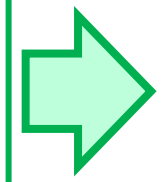
- AJAX (Asynchronous JavaScript and XML) is a way of exchanging data with a server and updating parts of a webpage - without reloading the whole page.
- jQuery provides several methods for AJAX functionality. With the jQuery AJAX methods, we can request text, HTML, XML, or JSON from a remote server.

https://www.w3schools.com/jquery/jquery_ajax_intro.asp
https://www.w3schools.com/xml/ajax_intro.asp

AJAX getJSON()

$.getJSON(url, data, func) sends a GET request to URL and will send the contents of the data object as query parameters. Once the data arrived, it will call the given function with the return value as argument.

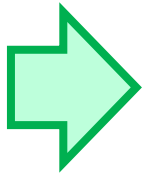
```
<html>
    <head>
        <title>Hello World</title>
        <script
        src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
        <script>
            $(function(){
                $("#a_button").click(function(){
                    $.getJSON('/an_ajax_example', {}, function(data) {
                    $("p").text(data.a_text_msg);
                    });
                    return false;
                });
            });
        </script>
    </head>

    <body>
        <p>Nothing</p>
        <input type="button" value="Click me" id="a_button">
    </body>
</html>
```



hello.html

In this example, the url is "/an_ajax_example", the data is set to empty {} and the function is to change content of the <p> element using the returned value.

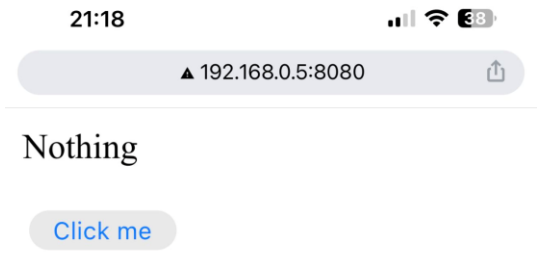
```
from flask import Flask,render_template,jsonify
app = Flask(__name__)

@app.route('/')
def hello_world():
        return render_template('hello.html')

@app.route('/an_ajax_example')
def an_ajax_example_function():
        print("The Flask server got a message")
        return jsonify(a_text_msg = "I got it.")

if __name__ == '__main__':
        IP = "192.168.0.5" # fill in your IP.
        PORT = "8080"
        DEBUG = True
        app.run(host = IP, port = PORT, debug =
        DEBUG)
```



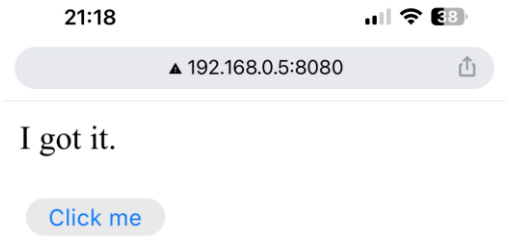
In the Flask server, we have a corresponding function to make response. Here, we can see the URL is '/an_ajax_example' which is the same as the one in $.getJSON() function call. The following function an_ajax_example_function() will make a response and return a JSON format message. The key is a_text_msg and the value is "I got it.".

```
(base) zhidongsu@ZhidongdeMacBook-Pro code % python hello.py
 * Serving Flask app 'hello' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://192.168.0.5:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 120-091-390
192.168.0.3 - - [27/Oct/2022 21:18:28] "GET / HTTP/1.1" 200 -
The Flask server got a message
192.168.0.3 - - [27/Oct/2022 21:18:34] "GET /an_ajax_example HTTP/1.1" 200 -
```

This figure shows the printed information when running the Flask code. We can see the server print the information “The Flask server got a message”



Initial webpage



After clicking the button