Saving Data on the Server

No screens

Prof. Lydia Chilton
COMS 4170
19 February 2020
My goal is to use your time well.

• I want you to understand how design is a useful skill in your life.
• I want to interact with you on an individual basis

• I do not want anyone to be bored, lost, zoned-out, or stalking their ex on Facebook.

No screens.  If you are lost, interact.
This is when the class starts to get hard.
Please don’t underestimate this class

Simple, functional design is deceptively difficult
Reason #1:

Although websites look easy, there is so much going on beneath the surface that you don’t see.
Reason #2:

**Systems programming is hard.**

Individually, they are not hard. But together, they are complex.

<table>
<thead>
<tr>
<th>Non-PPC</th>
<th>PPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Phyllis</td>
<td>HTML</td>
</tr>
<tr>
<td>2: Angela</td>
<td>CSS</td>
</tr>
<tr>
<td>3: Dwight</td>
<td>JavaScript</td>
</tr>
<tr>
<td>4: Oscar</td>
<td>JQuery</td>
</tr>
<tr>
<td>5: Creed</td>
<td>Bootstrap</td>
</tr>
<tr>
<td>6: Pam</td>
<td>Events</td>
</tr>
<tr>
<td>7: Jim</td>
<td>Interaction</td>
</tr>
<tr>
<td>8: Stanley</td>
<td>Model / View</td>
</tr>
<tr>
<td>9: Michael</td>
<td>Data Storage</td>
</tr>
<tr>
<td>10: Kevin</td>
<td>Visual Information hierarchy</td>
</tr>
<tr>
<td>11: Kelly</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- HTML
- CSS
- JavaScript
- JQuery
- Bootstrap
- Events
- Interaction
- Model / View
- Data Storage
- Visual Information hierarchy
• I do **not want** anyone to be confused, bored, or zoned-out.
• Using screens distracts your neighbors, and makes it harder for them to pay attention.

No screens.  

If you are lost, interact.
No screens

Prof. Lydia Chilton
COMS 4170
19 February 2020
Homework 4
In HW4, you dynamically create widgets

**Buttons**

- Submit
- X
- X
- X
- X
- X

**Autocomplete**

Log your paper sales:
- To
- Toast
- Flat Top

**Drag and Drop**

<table>
<thead>
<tr>
<th>Non-PPC</th>
<th>PPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Phyllis</td>
<td></td>
</tr>
<tr>
<td>2: Angela</td>
<td></td>
</tr>
<tr>
<td>3: Dwight</td>
<td></td>
</tr>
<tr>
<td>4: Oscar</td>
<td></td>
</tr>
<tr>
<td>5: Creed</td>
<td></td>
</tr>
<tr>
<td>6: Pam</td>
<td></td>
</tr>
<tr>
<td>7: Jim</td>
<td></td>
</tr>
<tr>
<td>8: Stanley</td>
<td></td>
</tr>
<tr>
<td>9: Michael</td>
<td></td>
</tr>
<tr>
<td>10: Kevin</td>
<td></td>
</tr>
<tr>
<td>11: Kelly</td>
<td></td>
</tr>
</tbody>
</table>

Added minor customization (hovering and drop target feedback)
And users interacting data

Create / Delete data

Update data
But there’s a big problem:

<table>
<thead>
<tr>
<th>Log your paper sales</th>
<th># Reams</th>
</tr>
</thead>
<tbody>
<tr>
<td>James D. Halpert</td>
<td>100</td>
</tr>
<tr>
<td>Stanley Hudson</td>
<td>400</td>
</tr>
<tr>
<td>Michael G. Scott</td>
<td>1000</td>
</tr>
</tbody>
</table>
Problem:

The data doesn’t save
If you refresh the page, your new data is gone. Why?

The data is only stored on the browser.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Columbia Paper Infinity</title>
</head>
<body class="container">
  <div class="Jumbotron">
    <h1>Columbia Paper Infinity</h1>
    <div>
      <label>Log your paper sales:</label>
      <table>
        <thead>
          <tr>
            <th></th>
            <th>Client</th>
            <th># Reams</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>James D. Halpert</td>
            <td>Shake Shack</td>
            <td>100</td>
            <td><input type="radio" name="servername" value="1" checked="checked"></td>
          </tr>
          <tr>
            <td>Stanley Hudson</td>
            <td>Toast</td>
            <td>400</td>
            <td><input type="radio" name="servername" value="1"></td>
          </tr>
          <tr>
            <td>Michael G. Scott</td>
            <td>Computer Science Department</td>
            <td>1000</td>
            <td><input type="radio" name="servername" value="1"></td>
          </tr>
        </tbody>
      </table>
      <input type="submit" value="Submit">
    </tr>
  </div>
  <script>
    var salesperson = "Dwight K. Schrute"
    var sales = [
      {
        "salesperson": "James D. Halpert",
        "client": "Shake Shack",
        "reams": 100
      },
      {
        "salesperson": "Stanley Hudson",
        "client": "Toast",
        "reams": 400
      },
      {
        "salesperson": "Michael G. Scott",
        "client": "Computer Science Department",
        "reams": 1000
      }
    ];
  </script>
</body>
</html>
```
To keep data around, we need to store it somewhere else – another computer that will never get turned off.

**Server:** keeps the data

**Client:** gets data from server (and displays it to all users)
What data does the server keep?

emails = [
    {
        "from": "bollinger",
        "to": "chilton",
        "subject": "4170 is awesome!"
    },
    {
        "from": "obama",
        "to": "chilton",
        "subject": "belated medal of freedom"
    }
]
What data does the server keep?

products = [
    {
        "title": "Ivy League Web Design",
        "author": "chilton",
        "stars": "5"
    },
    {
        "title": "JavaScript and You",
        "author": "chilton",
        "stars": "6"
    }
]

Server: keeps the data

Client: gets data from server (and displays it to all users)
What data does the server keep?

cars = [
    {
        "location": "116 and broadway",
        "driver": "kenny",
        "car type": "uber XL"
    },
    {
        "location": "times square",
        "driver": "jen",
        "car type": "normal"
    }
]

Server: keeps the data
Client: gets data from server (and displays it to all users)
What data does the server keep?

profiles = [
    {
        "name": "maddy",
        "image": "./maddy.png",
        "likes": "1000",
        "dislikes": 0,
    },
    {
        "name": "julia",
        "image": "./julia.png",
        "likes": "1000",
        "dislikes": 0,
    },
]

Server:
keeps the data

Client:
gets data from server
(and displays it to all users)
We need to have another computer **store and serve** the data. That server is running a Python application called **Flask**.

**Server:**
keeps the data

**Client:**
gets data from server (and displays it to all users)
Example application:

Storing and Serving data in Flask
You must first install Flask
Example project for storing a list of names. (code is on the webpage)
You have written the world’s smallest Flask app. Now what?
How to render an HTML page with data

```python
from flask import Flask

app = Flask(__name__)

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

@app.route('/hello/<name>/')
def hello(name=None):
    if name == '__main__':
        return render_template('hello.html', name=name)
    return render_template('hello.html', name=name)

if __name__ == '__main__':
    app.run()
```

```html
<html>
<head></head>
<body>
  <p>Hello {{name}}!</p>
</body>
</html>
```
How to send an array of data to JavaScript?

```python
from flask import Flask

app = Flask(__name__)

data = [
    {
        "id": 1,
        "name": "michael scott"
    },
    {
        "id": 2,
        "name": "jim halpert"
    },
]

@app.route('/people')
def people(name=None):
    return render_template('people.html', data=data)

if __name__ == '__main__':
    app.run()
```

```html
<html>
<head>
    <script>
        var data = '{{ data }}';
    </script>
</head>
<body>
    Hello people!
</body>
</html>
```
How to send an array of data to JavaScript?

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

data = [
    {
        "id": 1,
        "name": "michael scott"
    },
    {
        "id": 2,
        "name": "jim halpert"
    },
]

@app.route('/', methods=['GET'])
def hello_world():
    return 'Hello World'

@app.route('/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)

@app.route('/people')
def people(name=None):
    return render_template('people.html', data=data)

if __name__ == '__main__':
    app.run()
```

```html
<html>
<head>
    <script>
        var data = {{data|tojson}};
    </script>
</head>
<body>
    <h1>Hello people!</h1>
</body>
</html>
```
Iterate over the data
Display all the names

```html
<html>
<head>
<script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>

<script>
var data = {{ data|tojson }};

// Shorthand for $( document ).ready()
$(document).ready(function(){
  $.each(data, function(i, datum){
    var new_name = $('"<div>"'+datum['name']+'"</div>"
    $('"#people_container"').append(new_name)
  });
});
</script>
</head>
<body>
Hello people!
<div id="people_container">
</div>
</body>
</html>
```
How do we allow people to add names?

```html
<br>
<br>
name: <input id="new_name"></input>  <button id="submit_name">Submit</button>
<br>
<br>
<div id="people_container">
</div>
```
What does the click handler do?
Now what does the click handler do?
This is MVC updating. It will add names to the list. Will it save? **NO. The changes are only on the client**
Save the data to the server
Save the data to the server

```javascript
var saveName = function(name) {
    var data_to_save = {"name": name}
    $.ajax(
        {
            type: "POST",
            url: "add_name",
            contentType: "application/json; charset=utf-8",
            data: JSON.stringify(data_to_save),
            success: function(result) {
                var all_data = result["data"]
                data = all_data
                displayNames(data)
            },
            error: function(request, status, error) {
                console.log("Error");
                console.log(request)
                console.log(status)
                console.log(error)
            }
        }
    );
}
```
What happens on the server?
How do we test if the data saves to the server?

Refresh the page to see if the new data stays
Next Homework
Problem 1. Get the Flask sample code to run
Problem 2. Put a backend behind Log Sales

<table>
<thead>
<tr>
<th>Log your paper sales:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>James D. Halpert</td>
<td>100</td>
</tr>
<tr>
<td>Stanley Hudson</td>
<td>400</td>
</tr>
<tr>
<td>Michael G. Scott</td>
<td>1000</td>
</tr>
</tbody>
</table>
Problem 3. Put a backend behind PPC

<table>
<thead>
<tr>
<th>Non-PPC</th>
<th>PPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Phyllis</td>
<td></td>
</tr>
<tr>
<td>2: Angela</td>
<td></td>
</tr>
<tr>
<td>3: Dwight</td>
<td></td>
</tr>
<tr>
<td>4: Oscar</td>
<td></td>
</tr>
<tr>
<td>5: Creed</td>
<td></td>
</tr>
<tr>
<td>6: Pam</td>
<td></td>
</tr>
<tr>
<td>7: Jim</td>
<td></td>
</tr>
<tr>
<td>8: Stanley</td>
<td></td>
</tr>
</tbody>
</table>
HW5: Due Wednesday Feb 26th:

Problem 1. Get the Flask sample code to run

Problem 2. Put a backend behind Log Sales

Problem 3. Put a backend behind PPC

Please get this done today.
Fill out participation now!
HW 5 is out.
Participation