

Saving Data on the Server

No screens



Prof. Lydia Chilton
COMS 4170
19 February 2020

Say your name



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.

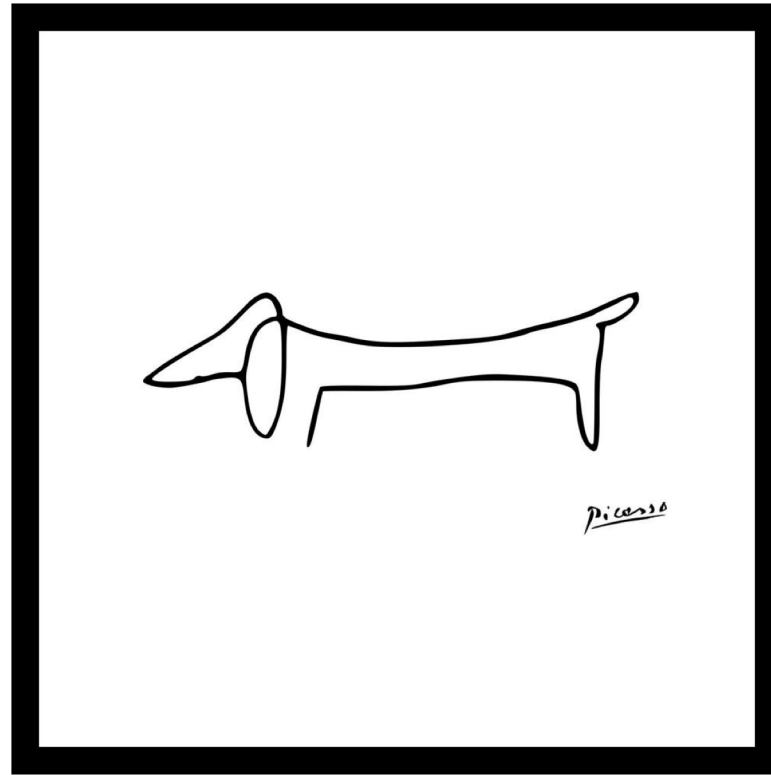


If you are lost, interact.



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.

Non-PPC	PPC
1: Phyllis	
2: Angela	
3: Dwight	
4: Oscar	
5: Creed	
6: Pam	
7: Jim	
8: Stanley	
9: Michael	
10: Kevin	
11: Kelly	

```
170
171 var names = [
172   "Phyllis",
173   "Angela",
174   "Dwight",
175   "Oscar",
176   "Creed",
177   "Pam"
178 ]
179 var list1 = []
180
181
182 function makeNames(names){
183   $("#names").empty()
184   $.each(names, function( index, value ) {
185     //make the draggable name object
186   });
187 }
188
189
190 $(document).ready(function(){
191   makeNames(names)
192
193   $("#team1_label").droppable({
194     drop: function( event, ui ) {
195       //get dropped name
196
197       //update names array
198
199       //update list1 array
200
201       //update the interface to display the new lists
202     }
203   });
204 })
205
206
```

Reason #2:

Systems programming is hard.

Non-PPC
1: Phyllis
2: Angela
3: Dwight
4: Oscar
5: Creed
6: Pam
7: Jim
8: Stanley
9: Michael
10: Kevin
11: Kelly

PPC

HTML

Events

CSS

Interaction

JavaScript

Model / View

JQuery

Data Storage

Bootstrap

Visual Information hierarchy

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

- 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.

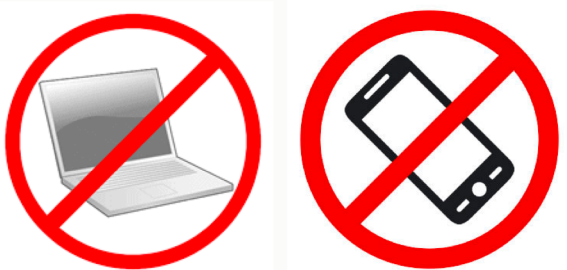


If you are lost, interact.



Saving Data on the Server

No screens



Prof. Lydia Chilton
COMS 4170
19 February 2020

Say your name



Homework 4

In HW4, you dynamically create widgets

Buttons

Submit



Autocomplete

Log your paper sales:

- Toast
- Flat Top

Drag and Drop

Non-PPC	PPC
1: Phyllis	
2: Angela	
3: Dwight	
4: Oscar	
5: Creed	
6: Pam	
7: Jim	
8: Stanley	
9: Michael	
10: Kevin	
11: Kelly	

Added minor customization
(hovering and drop target feedback)

And users interacting data

Columbia Paper Infinity

Log your paper sales:

<input type="text" value="Client"/>	<input type="text" value="# Reams"/>	<input type="button" value="Submit"/>
James D. Halpert	Shake Shack	100
Stanley Hudson	Toast	400
Michael G. Scott	Computer Science Department	1000

Each row in the table has a yellow button with an 'X' icon to its right.

Create / Delete data

Party Planning Committee

Non-PPC	PPC
1: Phyllis	
2: Angela	
3: Dwight	
4: Oscar	
5: Creed	
6: Pam	
7: Jim	
8: Stanley	

Update data

But there's a big problem:

File | /Users/lydiachilton/Dropbox/Columbia/Teaching/4170/Spring2020/hw/hw4-direct-manipulation/hw4-answers/log_sales/log_sales.html

Columbia Paper Infinity

Log your paper sales:

	<input type="text" value="Client"/>	<input type="text" value="# Reams"/>	<input type="button" value="Submit"/>
James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

Problem:

The data doesn't save

If you refresh the page, your new data is gone. Why?

Columbia Paper Infinity

Add data

Log your paper sales:

James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

Data appears

Log your paper sales:

Dwight K. Schrute	Computer Science Department	1	<input type="button" value="X"/>
James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

REFRESH PAGE

Data is gone!

Log your paper sales:

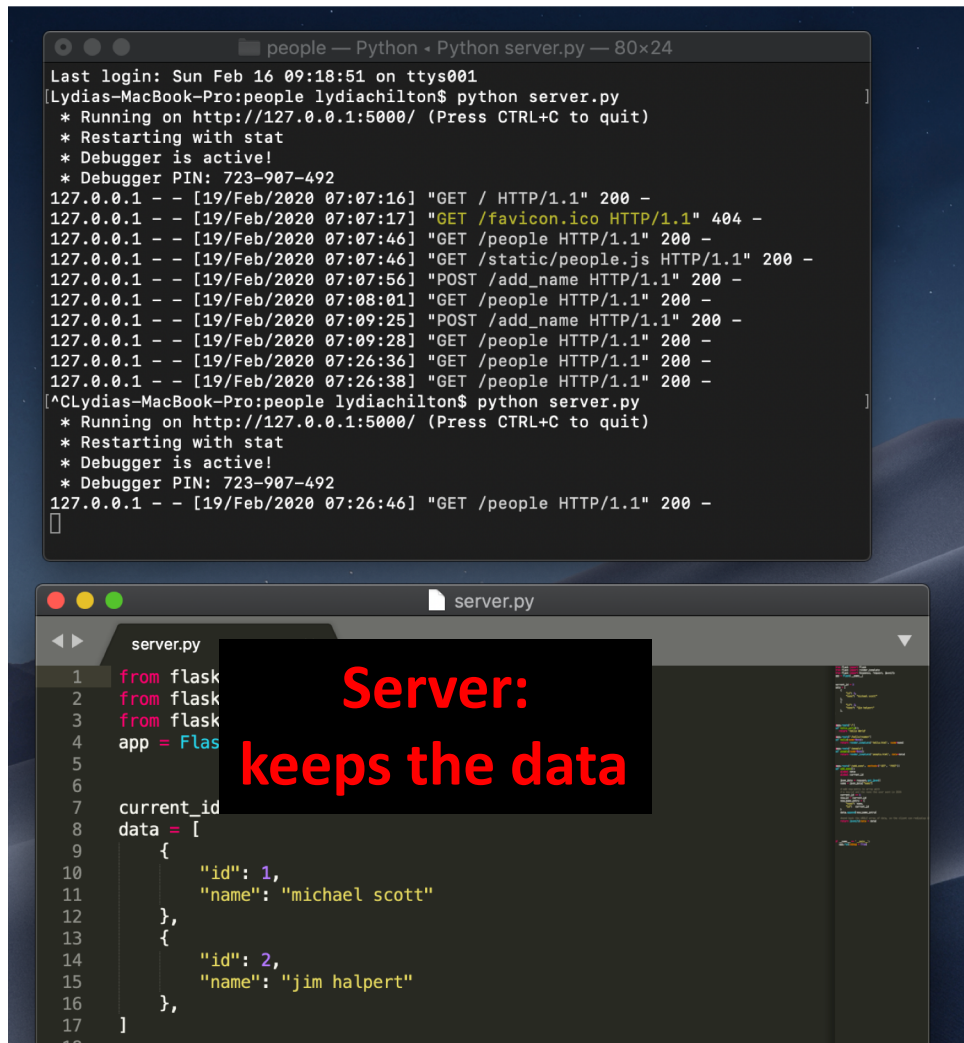
James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

```
1 <html>
2 <head>
3
4 <!-- My Scripts -->
5 <script>
6   var salesperson = "Dwight K. Schrute"
7
8   var sales = [
9     {
10      "salesperson": "James D. Halpert",
11      "client": "Shake Shack",
12      "reams": 100
13    },
14    {
15      "salesperson": "Stanley Hudson",
16      "client": "Toast",
17      "reams": 400
18    },
19    {
20      "salesperson": "Michael G. Scott",
21      "client": "Computer Science Department",
22      "reams": 1000
23    },
24  ]
25 </script>
26
27 </head>
28
29
30
31 <body>
32 <div class="container">
33   <div class="jumbotron">
34     <h1>Columbia Paper Infinity</h1>
35   </div>
36   <div id="logsales" >
37
38     <div class="row">
39       <div class="col-md-2">
40         Log your paper sales:
41       </div>
42       <div class="col-md-4">
43         <div class="ui-widget">
44           <input type="text" id="enter_client" placeholder="Client">
45           <div class="warning_div" id="client_warning_div"></div>
46         </div>

```

The data is only stored on the browser.

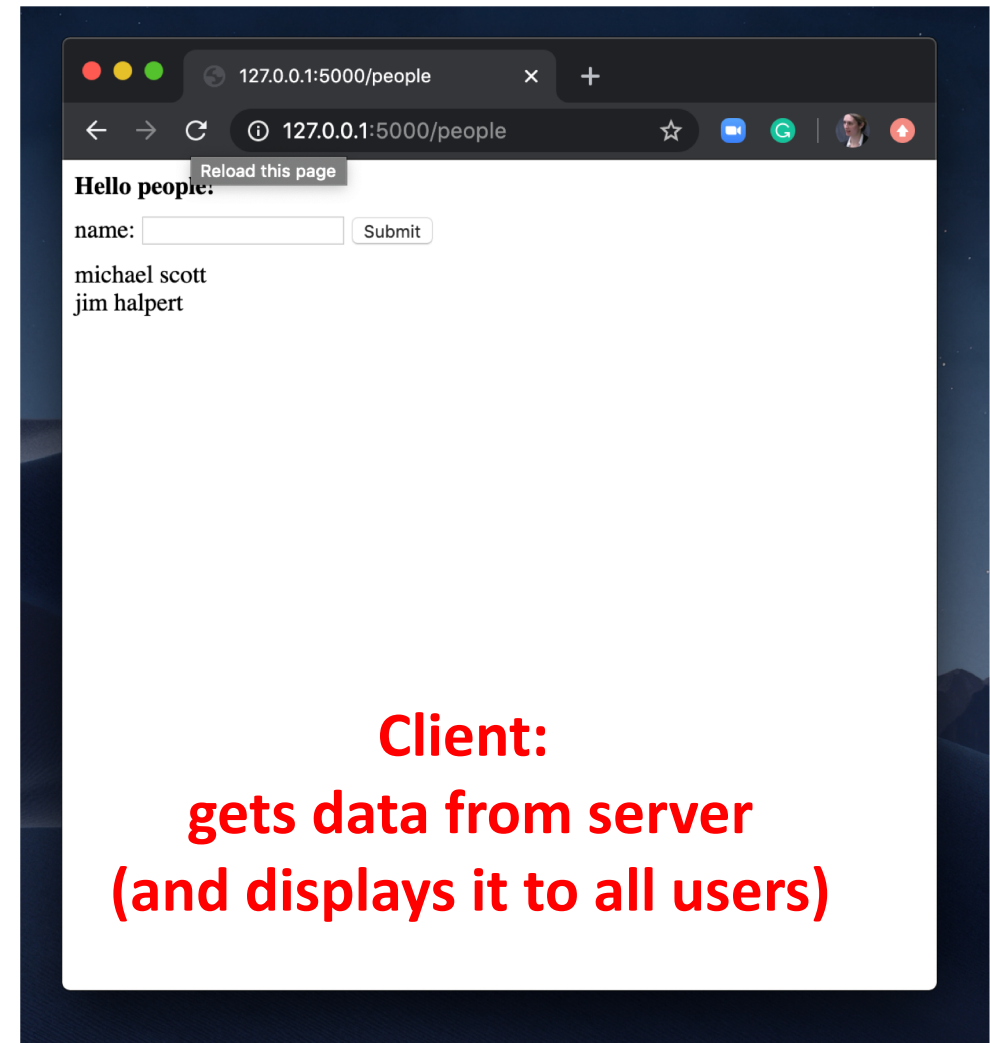
To keep data around, we need to store it somewhere else – another computer that will never get turned off.



The image shows two screenshots from a development environment. The top screenshot is a terminal window titled 'people — Python · Python server.py — 80x24'. It displays the output of running a Python server. The logs show the server starting on http://127.0.0.1:5000/ and handling several requests, including GET requests for /, /favicon.ico, and /people, and POST requests for /add_name. The bottom screenshot is a code editor window titled 'server.py' showing the following code:

```
1 from flask
2 from flask
3 from flask
4 app = Flas
5
6
7 current_id
8 data = [
9     {
10      "id": 1,
11      "name": "michael scott"
12     },
13     {
14      "id": 2,
15      "name": "jim halpert"
16     },
17 ]
```

**Server:
keeps the data**



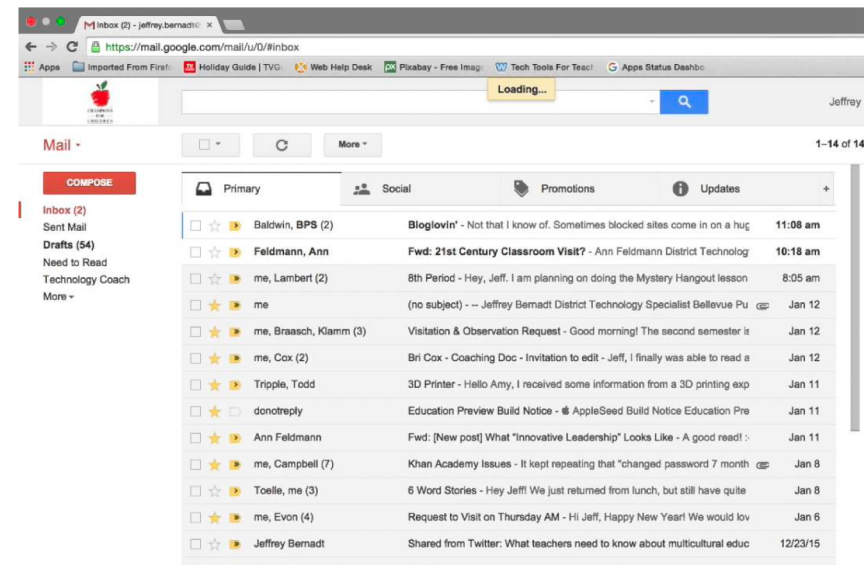
The image shows a screenshot of a web browser window. The address bar shows the URL '127.0.0.1:5000/people'. The page content includes a heading 'Hello people:', a form with a text input field labeled 'name:' and a 'Submit' button, and a list of names: 'michael scott' and 'jim halpert'. A red overlay with text is positioned at the bottom of the browser window.

**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"  
  },  
]
```

Server:
keeps the data



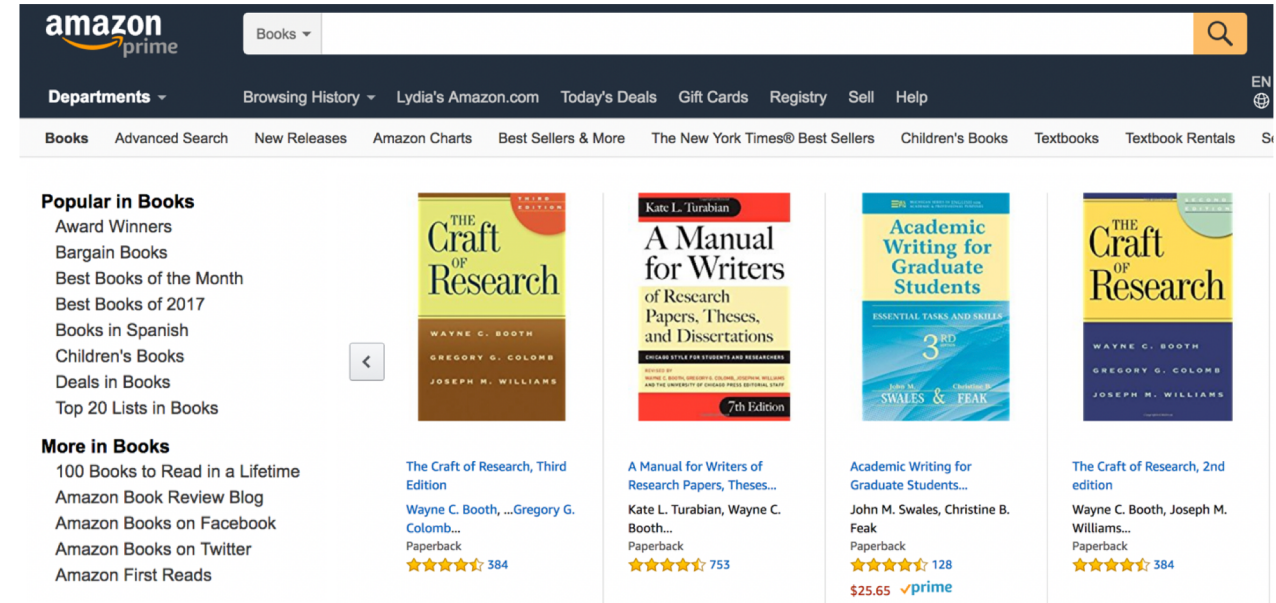
Client:
gets data from server
(and displays it to all users)

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

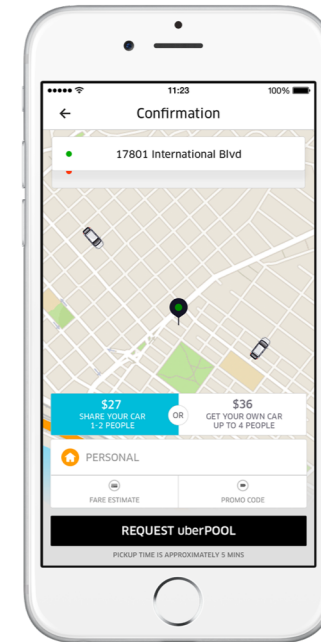
Client:
gets data from server
(and displays it to all users)



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



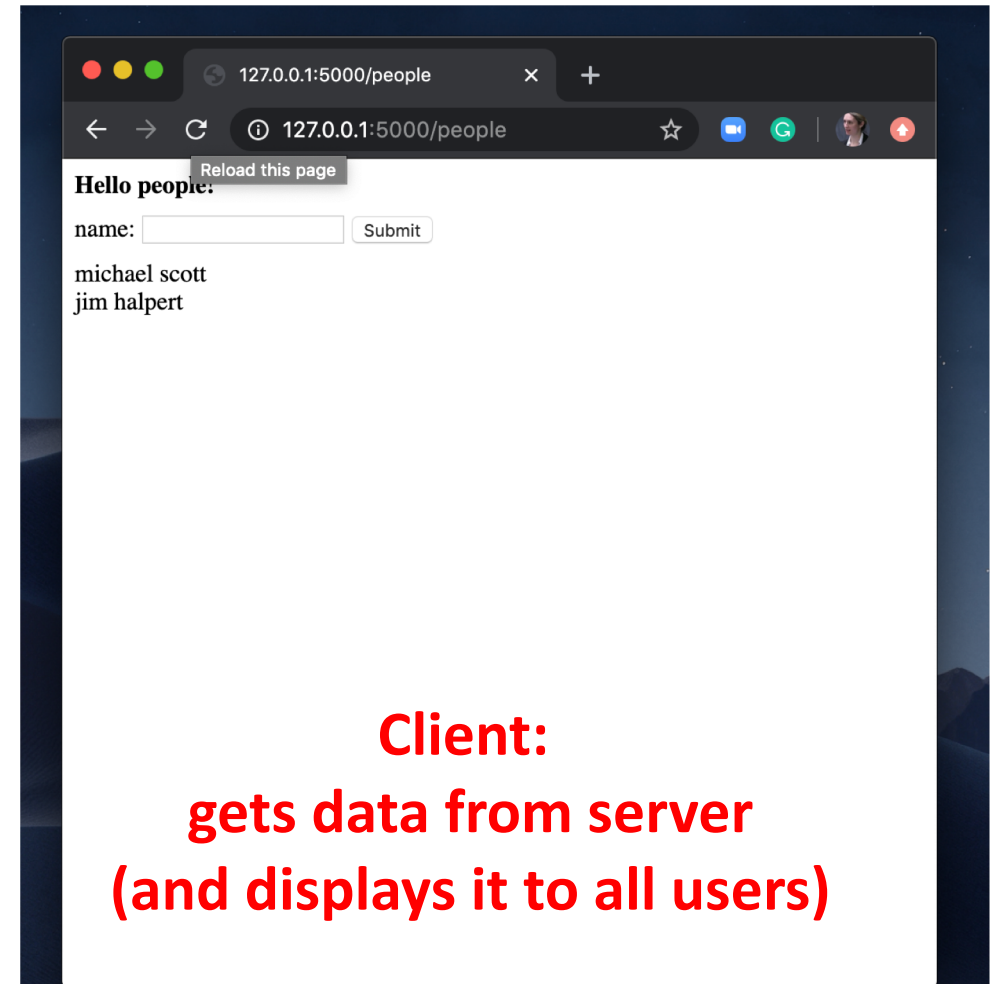
Client:
gets data from server
(and displays it to all users)

We need to have another computer store and serve the data.
That server is running a Python application called Flask.

```
people — Python · Python server.py — 80x24
Last login: Sun Feb 16 09:18:51 on ttys001
Lydias-MacBook-Pro:people lydiachilton$ python server.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 723-907-492
127.0.0.1 - - [19/Feb/2020 07:07:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:17] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [19/Feb/2020 07:07:46] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:46] "GET /static/people.js HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:56] "POST /add_name HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:08:01] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:09:25] "POST /add_name HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:09:28] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:26:36] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:26:38] "GET /people HTTP/1.1" 200 -
^CLydias-MacBook-Pro:people lydiachilton$ python server.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 723-907-492
127.0.0.1 - - [19/Feb/2020 07:26:46] "GET /people HTTP/1.1" 200 -
```

```
server.py
1 from flask
2 from flask
3 from flask
4 app = Flas
5
6
7 current_id
8 data = [
9
10     {
11         "id": 1,
12         "name": "michael scott"
13     },
14     {
15         "id": 2,
16         "name": "jim halpert"
17     },
18 ]
```

**Server:
keeps the data**

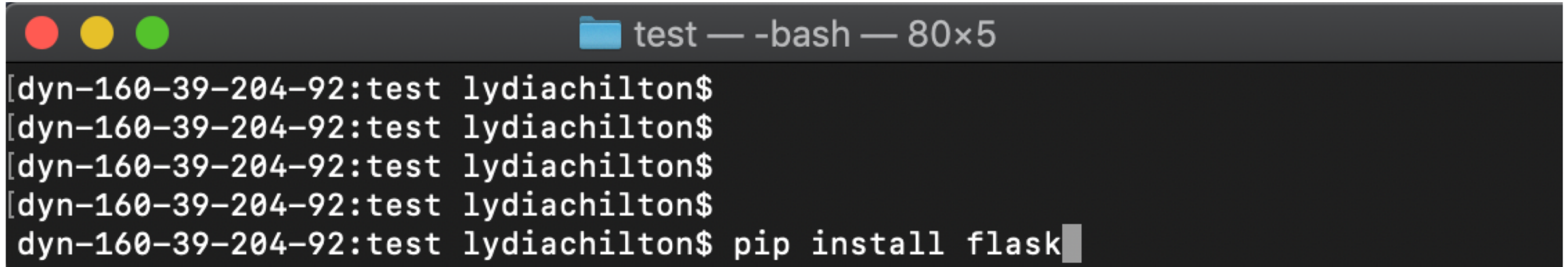


**Client:
gets data from server
(and displays it to all users)**

Example application:

Storing and Serving data in Flask

You must first install Flask

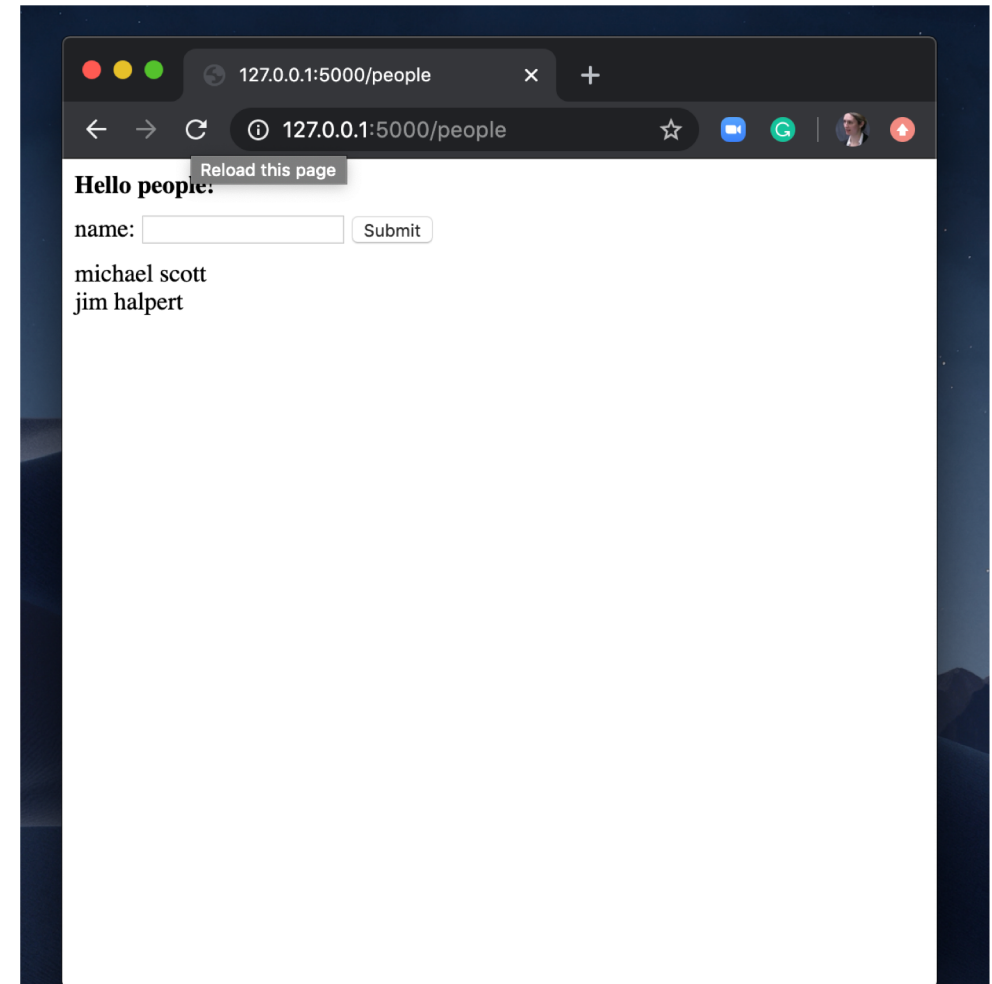


```
test — -bash — 80x5  
[dyn-160-39-204-92:test lydiachilton$  
[dyn-160-39-204-92:test lydiachilton$  
[dyn-160-39-204-92:test lydiachilton$  
[dyn-160-39-204-92:test lydiachilton$  
dyn-160-39-204-92:test lydiachilton$ pip install flask
```

Example project for storing a list of names. (code is on the webpage)

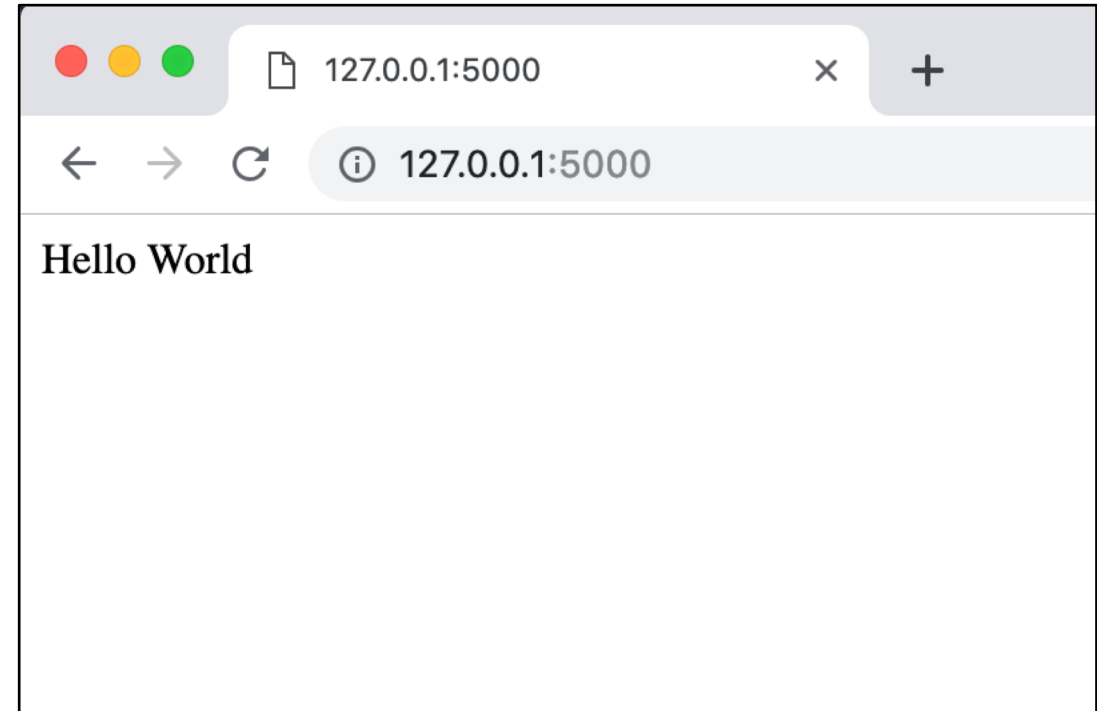
```
people — Python · Python server.py — 80x24
Last login: Sun Feb 16 09:18:51 on ttys001
Lydias-MacBook-Pro:people lydiachilton$ python server.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 723-907-492
127.0.0.1 - - [19/Feb/2020 07:07:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:17] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [19/Feb/2020 07:07:46] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:46] "GET /static/people.js HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:07:56] "POST /add_name HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:08:01] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:09:25] "POST /add_name HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:09:28] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:26:36] "GET /people HTTP/1.1" 200 -
127.0.0.1 - - [19/Feb/2020 07:26:38] "GET /people HTTP/1.1" 200 -
^CLydias-MacBook-Pro:people lydiachilton$ python server.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 723-907-492
127.0.0.1 - - [19/Feb/2020 07:26:46] "GET /people HTTP/1.1" 200 -
█
```

```
server.py
server.py x
1 from flask import Flask
2 from flask import render_template
3 from flask import Response, request, jsonify
4 app = Flask(__name__)
5
6
7 current_id = 2
8 data = [
9     {
10         "id": 1,
11         "name": "michael scott"
12     },
13     {
14         "id": 2,
15         "name": "jim halpert"
16     },
17 ]
```



You have written the world's smallest Flask app. Now what?

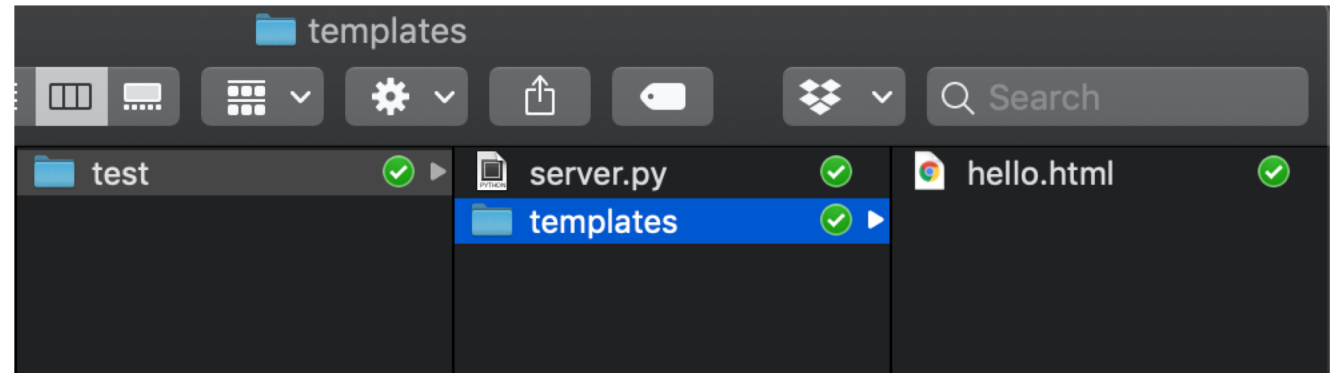
```
server.py
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return 'Hello World'
7
8 if __name__ == '__main__':
9     app.run()
```



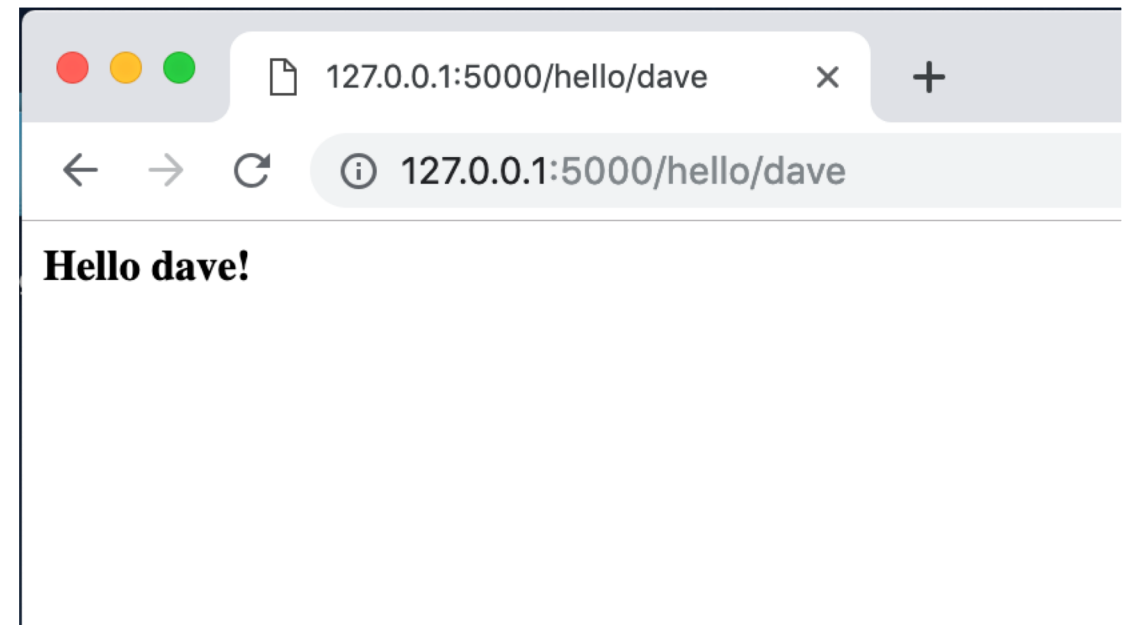
```
people — Python — Python server.py — 77x8
Lydias-MacBook-Pro:people lydiachilton$
Lydias-MacBook-Pro:people lydiachilton$
Lydias-MacBook-Pro:people lydiachilton$ python server.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 162-019-624
```


How to render an HTML page with data

```
server.py x hello.html x
1 from flask import Flask
2 from flask import render_template
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello_world():
7     return 'Hello World'
8
9 @app.route('/hello/<name>')
10 def hello(name=None):
11     return render_template('hello.html', name=name)
12
13 if __name__ == '__main__':
14     app.run()
15
16
```



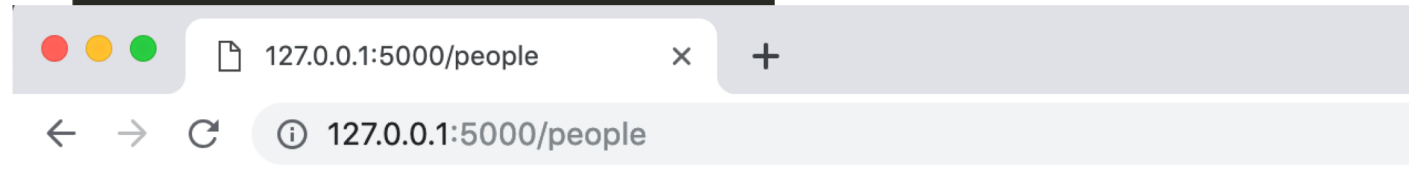
```
server.py x hello.html x
1 <html>
2 <head></head>
3 <body>
4
5 <b>Hello {{name}}!</b>
6 </body>
7 </html>
8
9
```



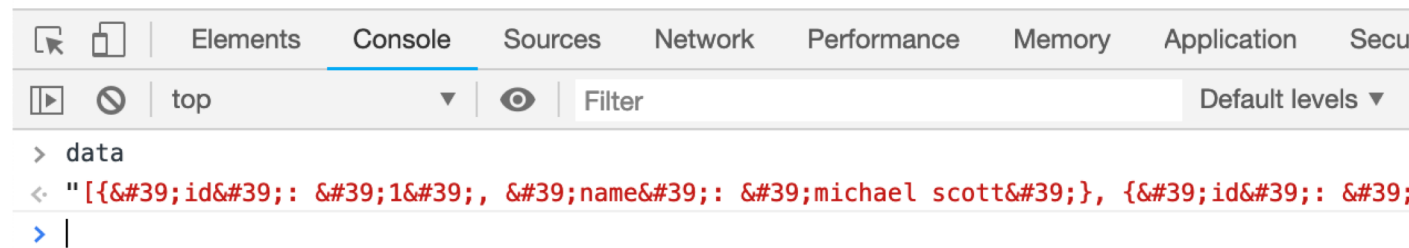
How to send an array of data to JavaScript?

```
server.py x hello.html x people.  
1 from flask import Flask  
2 from flask import render_template  
3 app = Flask(__name__)  
4  
5  
6 data = [  
7 {  
8 "id": 1,  
9 "name": "michael scott"  
10 },  
11 {  
12 "id": 2,  
13 "name": "jim halpert"  
14 },  
15 ]  
16  
17  
18  
19 @app.route('/')  
20 def hello_world():  
21     return 'Hello World'  
22  
23 @app.route('/hello/<name>')  
24 def hello(name=None):  
25     return render_template('hello.html', name=name)  
26  
27 @app.route('/people')  
28 def people(name=None):  
29     return render_template('people.html', data=data)  
30  
31 if __name__ == '__main__':  
32     app.run()  
33  
34  
35
```

```
people.html x server  
1 <html>  
2 <head>  
3  
4 <script>  
5     var data = '{{ data }}';  
6 </script>  
7
```



Hello people!

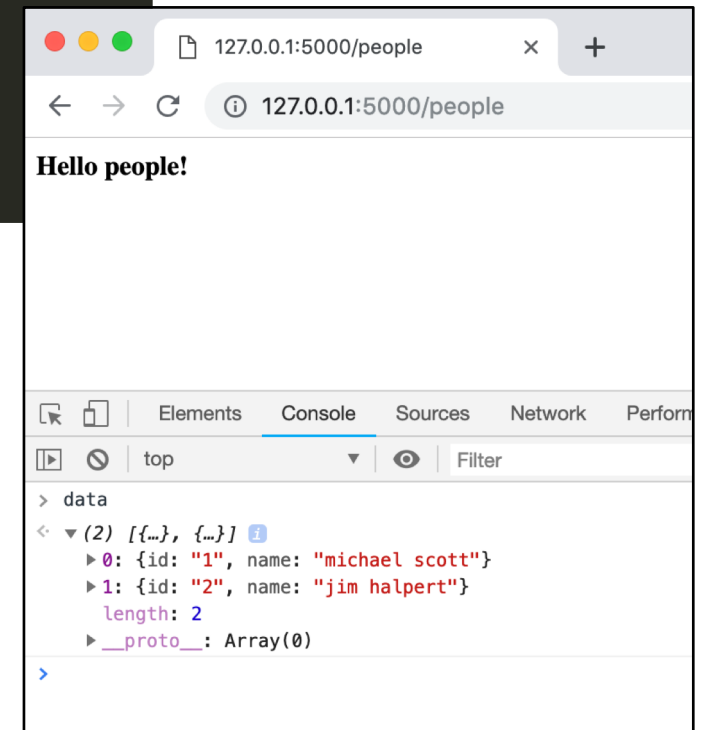


How to send an array of data to JavaScript?

```
server.py x hello.html x people.  
1 from flask import Flask  
2 from flask import render_template  
3 app = Flask(__name__)  
4  
5  
6 data = [  
7 {  
8     "id": 1,  
9     "name": "michael scott"  
10 },  
11 {  
12     "id": 2,  
13     "name": "jim halpert"  
14 },  
15 ]  
16  
17  
18  
19 @app.route('/')  
20 def hello_world():  
21     return 'Hello World'  
22  
23 @app.route('/hello/<name>')  
24 def hello(name=None):  
25     return render_template('hello.html', name=name)  
26  
27 @app.route('/people')  
28 def people(name=None):  
29     return render_template('people.html', data=data)  
30  
31 if __name__ == '__main__':  
32     app.run()  
33  
34  
35
```

```
people.html x server  
1 <html>  
2 <head>  
3  
4 <script>  
5     var data = '{{ data }}';  
6 </script>  
7  
8 </head>  
9 <body>  
10  
11 <b>Hello people!</b>  
12 </body>  
13 </html>  
14
```

```
<script>  
    var data = {{data|tojson}}  
</script>
```



Iterate over the data

```
people.html
server.py
hello.html

1 <html>
2 <head>
3   <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4   <script>
5     var data = [{ data|tojson }];
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9
10      $.each(data, function(i, datum){
11        console.log(datum)
12      })
13
14    })
15  </script>
16
17 </head>
18 <body>
19
20
21 <b>Hello people!</b>
22 <div id="people_container">
23 </div>
24
25 </body>
26 </html>
```

127.0.0.1:5000/people

127.0.0.1:5000/people

Hello people!

Elements Console Sources Network

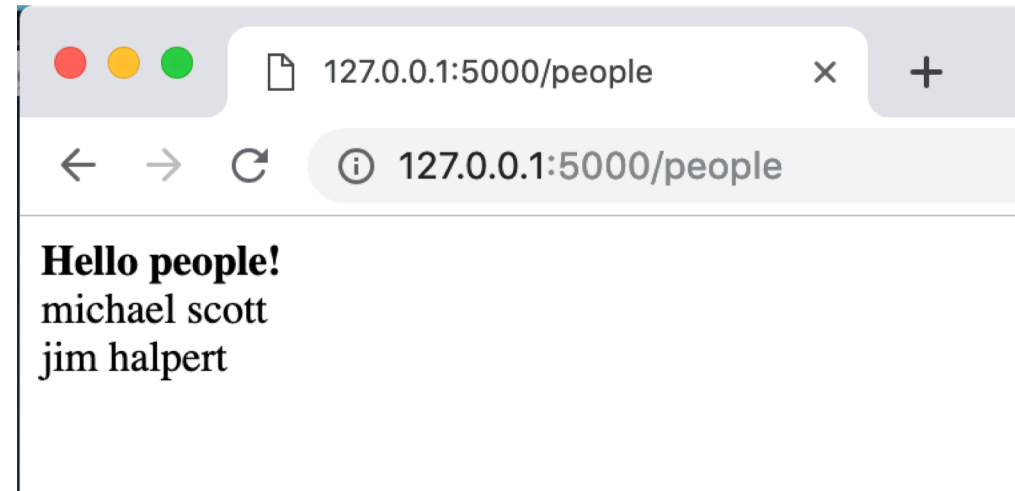
top Filter

- ▶ {id: "1", name: "michael scott"}
- ▶ {id: "2", name: "jim halpert"}

Display all the names

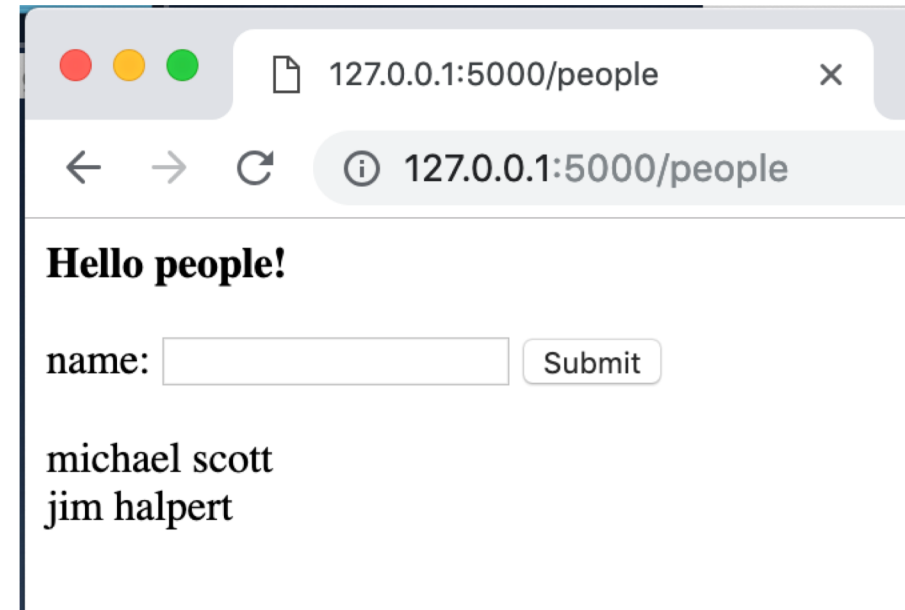
```
people.html
server.py
hello.html

1 <html>
2 <head>
3   <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4   <script>
5     var data = [{ data|tojson }];
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9
10      $.each(data, function(i, datum){
11        var new_name= "<div>" + datum["name"] + "</div>"
12        $("#people_container").append(new_name)
13      })
14    })
15  </script>
16
17 </head>
18
19 <body>
20
21 <b>Hello people!</b>
22 <div id="people_container">
23 </div>
24
25
26 </body>
27 </html>
28
29
```



How do we allow people to add names?

```
<b>Hello people!</b>
<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?

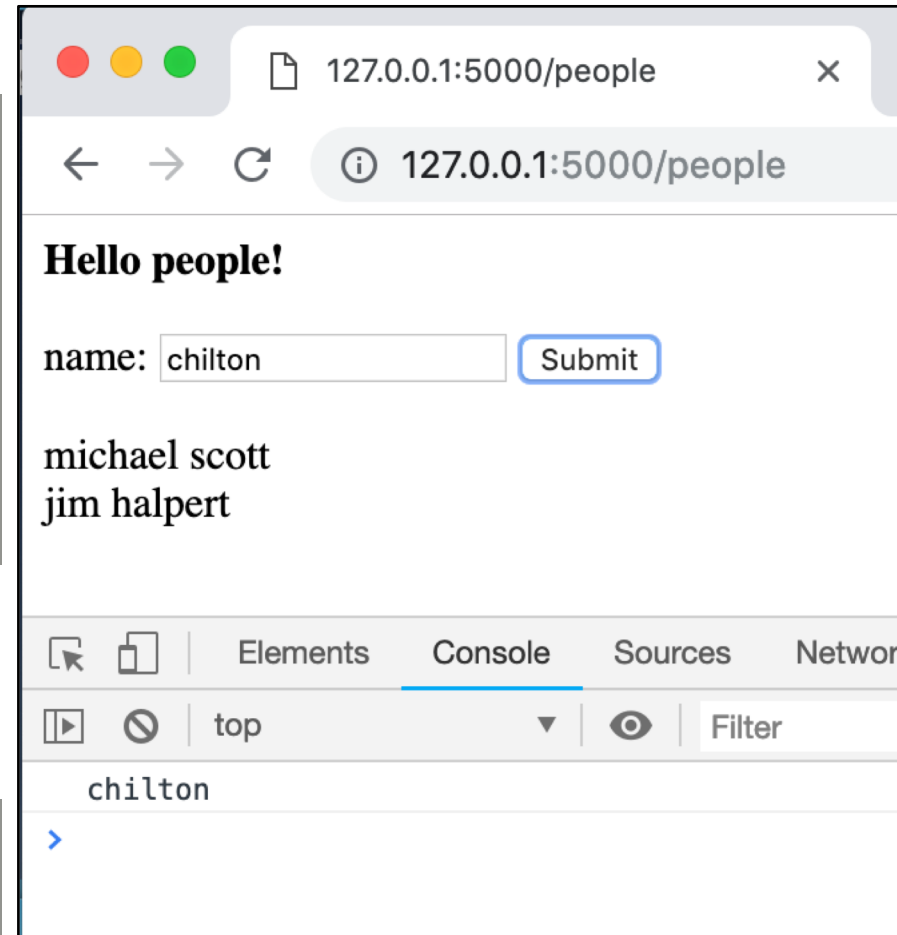
```
people.html
server.py
hello.html

1 <html>
2 <head>
3   <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4   <script>
5     var data = [{ data|tojson }];
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9
10      $.each(data, function(i, datum){
11        var new_name= $("<div>" + datum["name"] + "</div>")
12        $("#people_container").append(new_name)
13      })
14
15      $("#submit_name").click(function(){
16        console.log("new name")
17      })
18    })
19  }
20
21 </script>
22
23 </head>
24 <body>
25
26 <b>Hello people!</b>
27 <br>
28 <br>
29
30 name: <input id="new_name"></input> <button id="submit_name">Submit</button>
31 <br>
32
```

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/people`. The page content includes the heading **Hello people!**, a form with a text input field labeled "name:" and a "Submit" button. Below the form, the text "michael scott" and "jim halpert" is displayed. The browser's developer console is open, showing a log entry with the text "new name".

Now what does the click handler do?

```
1 <html>
2 <head>
3   <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4   <script>
5     var data = [{ data|tojson }];
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9
10      $.each(data, function(i, datum){
11        var new_name= $("#div">"+datum["name"]+"</div>")
12        $("#people_container").append(new_name)
13      })
14
15      $("#submit_name").click(function(){
16
17        var name = $("#new_name").val()
18        console.log(name)
19
20      })
21    })
22  </script>
23
24
25
```



This is MVC updating. It will add names to the list.
Will it save? **NO. The changes are only on the client**

```
1 <html>
2 <head>
3   <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4   <script>
5     var data = [{ data|tojson }];
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9       //when the page loads, display all the names
10      displayNames(data)
11
12      $("#submit_name").click(function(){
13        var name = $("#new_name").val()
14        console.log(name)
15
16        var new_id = data.length + 1
17        var new_name = name
18        var new_data = {
19          "id": new_id,
20          "name": new_name
21        }
22        data.push(new_data)
23        displayNames(data)
24      })
25    }
26  </script>
```

127.0.0.1:5000/people

127.0.0.1:5000/people

Hello people!

name:

michael scott
jim halpert
chilton

Elements Console Sources Network

top Filter

chilton

> data

< (3) [{...}, {...}, {...}]

- ▶ 0: {id: 1, name: "michael scott"}
- ▶ 1: {id: 2, name: "jim halpert"}
- ▶ 2: {id: 3, name: "chilton"}

length: 3

▶ __proto__: Array(0)

Save the data to the server

```
people.html x server.py x hello.html x
1 <html>
2 <head>
3 <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4 <script>
5   var data = {{ data|tojson }};
6
7   // Shorthand for $( document ).ready()
8   $(document).ready(function(){
9     //when the page loads, display all the names
10    displayNames(data)
11
12    $("#submit_name").click(function(){
13      var name = $("#new_name").val()
14      console.log(name)
15
16      ??????
17
18
19
20
21
22
23
24
25
```

Save the data to the server

```
people.html x server.py x hello.html x
1 <html>
2 <head>
3 <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
4 <script>
5     var data = {{ data|tojson }};
6
7     // Shorthand for $( document ).ready()
8     $(document).ready(function(){
9         //when the page loads, display all the names
10        displayNames(data)
11
12        $("#submit_name").click(function(){
13            var name = $("#new_name").val()
14            console.log(name)
15
16            saveName(name)
17
18
19
20
21
22
23
24
25
```

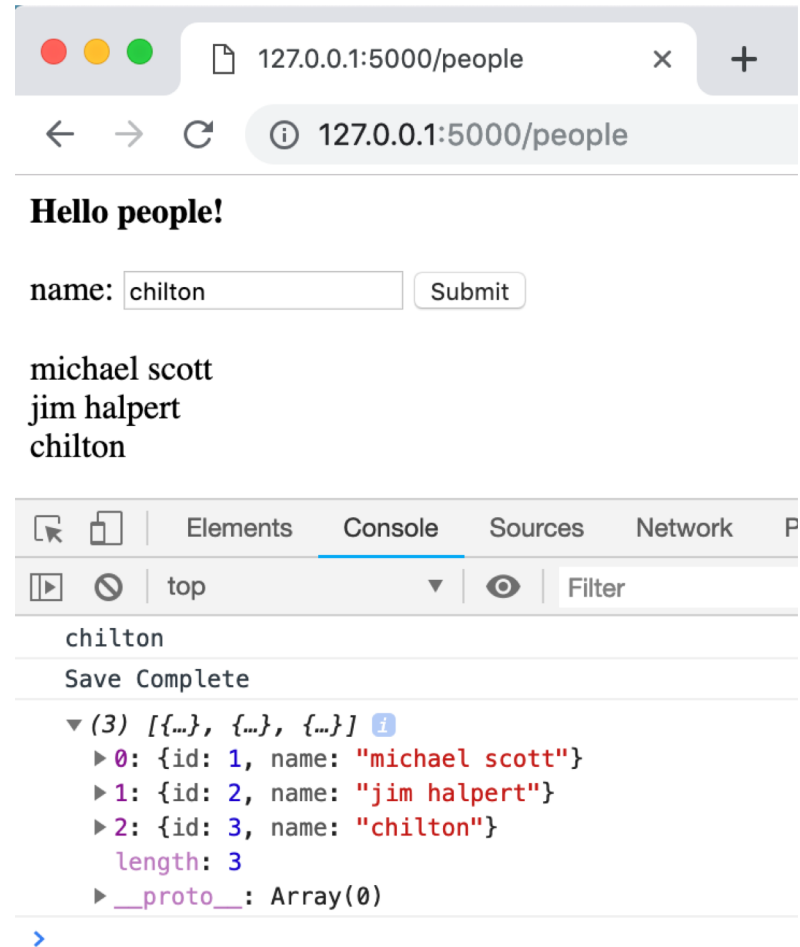
```
var saveName = function(name){
    var data_to_save = {"name": name}
    $.ajax({
        type: "POST",
        url: "add_name",
        dataType: "json",
        contentType: "application/json; charset=utf-8",
        data: JSON.stringify(data_to_save),
        success: function(result){
            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)
        }
    });
}
```

```
server.py hello.html x people
1 from flask import Flask
2 from flask import render_template
3 from flask import Response, request, jsonify
4 app = Flask(__name__)
5
6
7 current_id = 2
8 data = [
9     {
10         "id": 1,
11         "name": "michael scott"
12     },
13     {
14         "id": 2,
15         "name": "jim halpert"
16     },
17 ]
18
19
20 @app.route('/people')
21 def people(name=None):
22     return render_template('people.html', data=data)
23
24
25 @app.route('/add_name', methods=['GET', 'POST'])
26 def add_name():
27     global data
28     global current_id
29
30     json_data = request.get_json()
31     name = json_data["name"]
32
33     # add new entry to array with
34     # a new id and the name the user sent in JSON
35     current_id += 1
36     new_id = current_id
37     new_name_entry = {
38         "name": name,
39         "id": current_id
40     }
41     data.append(new_name_entry)
42
43     #send back the WHOLE array of data, so the client
44     return jsonify(data = data)
45
```

the server?

```
var saveName = function(name){
  var data_to_save = {"name": name}
  $.ajax({
    type: "POST",
    url: "add_name",
    datatype: "json",
    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)
    }
  });
}
```

How do we test if the data saves to the server?



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/people`. The page content includes the heading **Hello people!**, a form with a text input containing `chilton` and a `Submit` button, and a list of names: `michael scott`, `jim halpert`, and `chilton`.

The browser's developer console is open, showing the `Console` tab. The console output includes the text `chilton` and `Save Complete`. Below this, there is a log entry for an array of three objects:

```
(3) [{"id": 1, "name": "michael scott"}, {"id": 2, "name": "jim halpert"}, {"id": 3, "name": "chilton"}]
```

The array structure is expanded to show the following details:

- `0`: `{id: 1, name: "michael scott"}`
- `1`: `{id: 2, name: "jim halpert"}`
- `2`: `{id: 3, name: "chilton"}`
- `length`: `3`
- `__proto__`: `Array(0)`

Refresh the page to see if the new data stays

Next Homework

Problem 1. Get the Flask sample code to run

```
server.py
1 from flask import Flask
2 from flask import render_template
3 from flask import Response, request, jsonify
4 app = Flask(__name__)
5
6
7 current_id = 2
8 data = [
9     {
10         "id": 1,
11         "name": "michael scott"
12     },
13     {
14         "id": 2,
15         "name": "jim halpert"
16     },
17 ]
18
19
20 @app.route('/people')
21 def people(name=None):
22     return render_template('people.html', data=data)
23
24
25 @app.route('/add_name', methods=['GET', 'POST'])
26 def add_name():
27     global data
28     global current_id
29
30     json_data = request.get_json()
31     name = json_data["name"]
32
33     # add new entry to array with
34     # a new id and the name the user sent in JSON
35     current_id += 1
36     new_id = current_id
37     new_name_entry = {
38         "name": name,
39         "id": current_id
40     }
41     data.append(new_name_entry)
42
43     #send back the WHOLE array of data, so the client
44     return jsonify(data = data)
45
```

127.0.0.1:5000/people

127.0.0.1:5000/people

Hello people!

name:

michael scott
jim halpert
chilton

Elements Console Sources Network P

top Filter

chilton

Save Complete

(3) [{"id": 1, "name": "michael scott"}, {"id": 2, "name": "jim halpert"}, {"id": 3, "name": "chilton"}]

- ▶ 0: {id: 1, name: "michael scott"}
- ▶ 1: {id: 2, name: "jim halpert"}
- ▶ 2: {id: 3, name: "chilton"}

length: 3

▶ __proto__: Array(0)

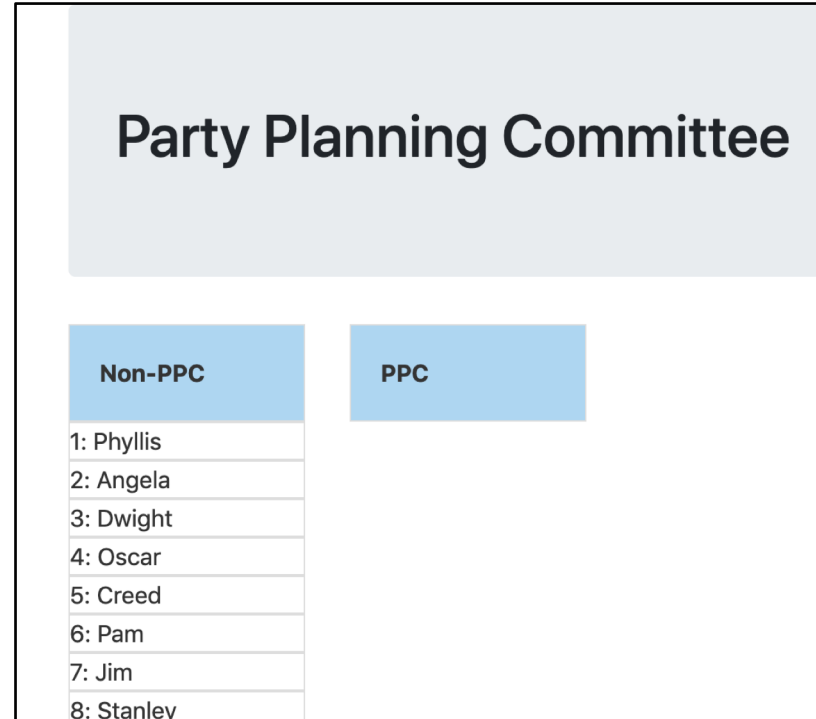
Problem 2. Put a backend behind Log Sales

Columbia Paper Infinity

Log your paper sales:

<input type="text" value="Client"/>	<input type="text" value="# Reams"/>	<input type="submit" value="Submit"/>	
James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

Problem 3. Put a backend behind PPC



HW5: Due Wednesday Feb 26th:

Problem 1.
Get the Flask sample code to run

127.0.0.1:5000/people

127.0.0.1:5000/people

Hello people!

name:

michael scott
jim halpert
chilton

```
chilton
Save Complete
(3) [{"id": 1, "name": "michael scott"}, {"id": 2, "name": "jim halpert"}, {"id": 3, "name": "chilton"}]
  length: 3
  __proto__: Array(0)
```

Problem 2.
Put a backend behind Log Sales

Columbia Paper Infinity

Log your paper sales: # Reams

James D. Halpert	Shake Shack	100	<input type="button" value="X"/>
Stanley Hudson	Toast	400	<input type="button" value="X"/>
Michael G. Scott	Computer Science Department	1000	<input type="button" value="X"/>

Problem 3.
Put a backend behind PPC

Party Planning Committee

Non-PPC	PPC
1: Phyllis	
2: Angela	
3: Dwight	
4: Oscar	
5: Creed	
6: Pam	
7: Jim	
8: Stanley	

Please get this done today.

Fill out participation now!
HW 5 is out.

Columbia University

User Interface Design

COMS 4170 · Spring 2020

Home Grading Syllabus **Piazza**

5	FEBRUARY 17	FEBRUARY 19	FEBRUARY 21
	Participation Form HW3 Review Menus and Navigation	Homework 4 due @ 4pm Contact Melanie for grading concerns. Participation Form Homework 5 out people.zip Saving Data on the Server	

Participation