Server-side programming

CS147L Lecture 6
Mike Krieger

Intro

Welcome back!

By the end of today...

- Server-side programming basics
- What to implement for this class
- PHP basics & syntax
- Calling PHP scripts from your Javascript
- Talking to a local database using PHP

Reminder: Getting stuff onto the Stanford space

With an FTP program

- Sign on to **cardinal.stanford.edu**
- Transmit static files to WWW/ folder, cgi scripts to cgi-bin/ folder
- If you still haven't, request CGI setup at **http://www.stanford.edu/services/web/cgi/personal.html** (can take a few hours)

Then…

- files will be available at:

  http://www.stanford.edu/~yoursunetid

- server scripts at:

  http://www.stanford.edu/~yoursunetid/cgi-bin

Server-side

The 4 components

HTML

content

hello world

CSS

style

div { font-weight: bold}

Javascript

action

$('div').click(function...

Server-side

communication

onRequest: return {"hello":"world"}

Options

Languages

- Python *(frameworks like Django; robust library support)*
- Ruby *(fmwrks like Rails; easy to pick up)*
- PHP *(ubiquitous server support, pretty easy to pick up)*
- Perl, ASP.net, ...

For this class…

- We'll use PHP, easiest to get going
  - *Recommendation*: just use Stanford web hosting, instead of local
  - But...see tech group or CS147L page for instructions for local running of PHP

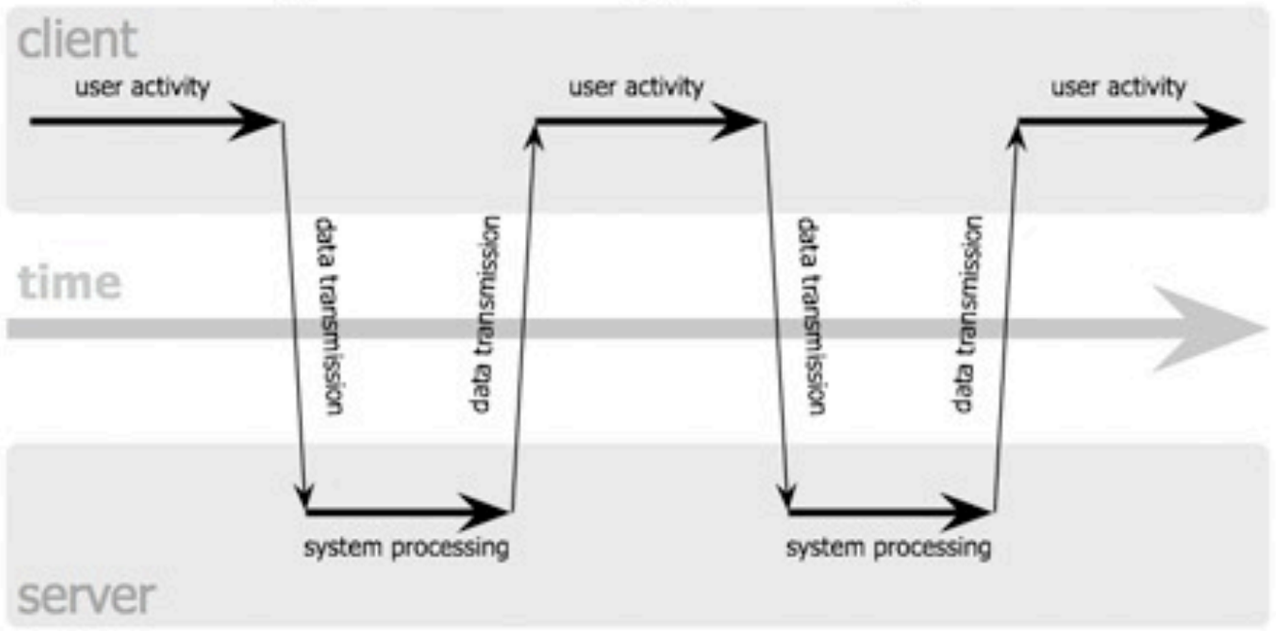
In the future…

- Frameworks like **Django** and **Rails** provide great ways to get projects off the ground & let you focus on interaction

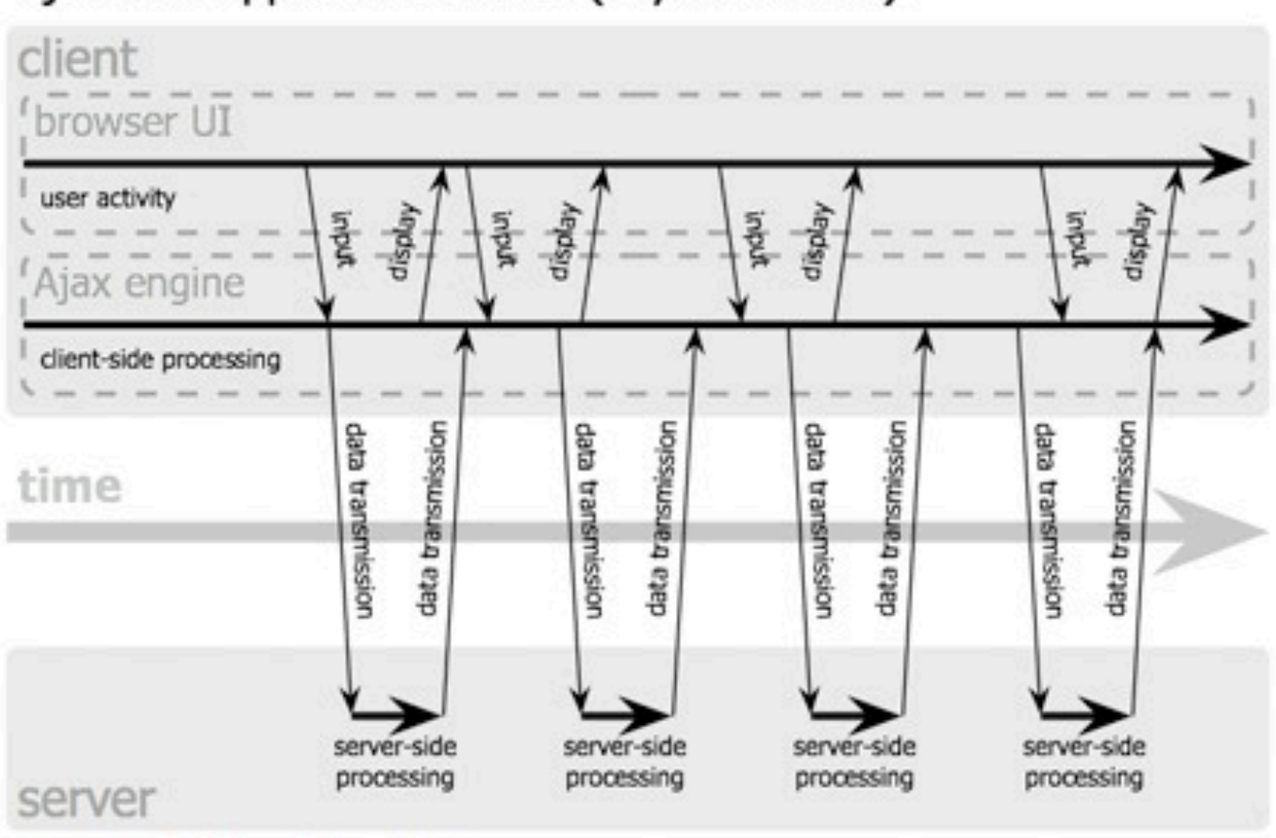
When server-side runs

- When rendering the page
- As an asynchronous request from the page itself

classic web application model (synchronous)



Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

General Strategy

- GET content, POST user data
- Server responds using JSON

Saving time

For CS147

- Only do server-side when it helps a part of your app feel more “real” or if it’s the fastest way to get something done
- If you need to persist user data, or let people exchange messages, etc.
- Don’t worry much about scalability, performance, security...

In other words…

- Do the simplest thing you can to prototype your idea, & focus on the interactive components and variations instead

PHP intro

A brief history

- Originally stood for “personal home page”
- Later: “PHP=*PHP: Hypertext Processor*”
- Current version: PHP5 (available on Stanford servers)

Large PHP-based sites

- Facebook
- Flickr
- Wikipedia

Two ways of using PHP

- Rendering web pages (mixed in with HTML)
- Just responding to requests (handling form submission, etc)

Our first PHP script

hello.php

```
<?php
    print "hello!";
?>
```

Demo

`hello.php`

<?php ?>

- Anything inside will be executed as PHP
- Anything outside will just be rendered to the browser

Testing PHP

- Load up the page in browser, or:
- From command line, run: php5 yourfilename.php (will act as if it's rendering to the browser)

Example: Running hello.php from cmd

helloinhtml.php

```
<body>
<div>Now I'm going to print some PHP stuff:</div>
<?php
   print "this is coming from PHP, can you tell?";
?>
<div>this is some other stuff....</div>
</body>
```

Demo

`helloinhtml.php`

Some more basics

```
<?php
    // variables have $ before them, don't get mixed up with jQuery :)
    $hello = "this is a string";
    $another = 5; /* PHP is weak-typed like Javascript */
    // for loops look familiar:
    for ($i=0; $i < 10; $i++) {
        // concatenate strings using a dot (.):
        print "this is iteratation number ".$i."<br/>";
    }
    // semicolons are mandatory!
?>
```

Arrays

- Ordered (like JS arrays) or associative (like JS objects)
- Can have mixed types inside them, like JS

Ordered Arrays

```
>> $arr = array('zero', 'one', 'two');
array(3) { [0]=> string(4) "zero" [1]=> string(3) "one" [2]=>
string(3) "two" }

>> $arr2 = array();
array(0) { }

>> $arr2[] = 'zero';
array(1) { [0]=> string(4) "zero" }

>> $last = array_pop($arr2);
string(4) "zero

>> var_dump($arr2); // useful debugging command
array(0) { }
```

Iterating through array

```
>> $arr = array('red', 'green', 'yellow');
>> foreach ($arr as $key => $value) {
>>  print "index: ".$key." ";
>>  print "value: ".$value."<br/>";
>> }
index: 0 value: red
index: 1 value: green
index: 2 value: yellow
```

Associative Array

```
>> $assoc = array("red"=>"angry", "blue"=>"sad");
array(2) { ["red"]=> string(5) "angry" ["blue"]=> string(3) "sad" }

>> $assoc["red"];
string(5) "angry"

>> $assoc["green"] = "envy";
array(3) { ["red"]=> string(5) "angry" ["blue"]=> string(3)
"sad" ["green"]=> string(4) "envy" }

>> unset($assoc["red"]);
array(2) { ["blue"]=> string(3) "sad" ["green"]=> string(4) "envy" }
```

Functions

```
<?php
// functions can have default arguments:
function printStuff($what_to_print, $linebreak=true) {
    print $what_to_print;
    if ($linebreak == true) print "<br/>";
}

printStuff("hello");
printStuff("how's life?");
printStuff("I'm doing well.", false);
printStuff(" Very well, actually");

?>
```

Demo

`functions.php`

Today's demo app:

cafés on campus

Features

- Get information about 3 different cafés on campus
- Post a review about food at each
- View other people's messages
- If we have time: tell us which café is nearest to us

To follow along

- http://mkrieger.org/week06.zip or do an **svn update** and look inside the week6 folder
- Examples will work best in Safari or Chrome
  - Use FTP to cgi-bin files to your cgi-bin folder, and the WWW/ files into your WWW/ folder
  - Using SecureCRT or the Terminal, log into **cardinal.stanford.edu** to run from command line

Part 1 : Café List

For now...

- Hard-code 3 different café names
- All of our server responses will be JSON for parsing in the Javascript

JSON in PHP

```
// remember: JSON is just the Javascript object notation

>> $arr = array("CS"=>array("CS147", "CS108"), "ME"=>array
("ME101, ME310"));
array(2) { ["CS"]=> array(2) { [0]=> string(5) "CS147" [1]=>
string(5) "CS108" } ["ME"]=> array(1) { [0]=> string(12)
"ME101, ME310" } } //not suitable for Javascript!

>> $json_data = json_encode($arr);
{"CS":["CS147","CS108"],"ME":["ME101, ME310"]} //much better!
```

Exercise 1

Take this data (data/cafes.txt) and make it a PHP array:

```
Cafes:
    - NeXus Cafe
        - 318 Campus Drive West, Stanford, CA
        - Opens at 8am
        - Closes at 3pm
    - Bytes Cafe
        - 350 Serra Mall, Stanford, CA
        - Opens at 7am
        - Closes at 3pm
    - Cool Cafe
        - 328 Lomita Drive, Stanford, CA
        - Opens at 11am
        - Closes at 1pm
```

Live TextMateing

getcafes.php

```
<?php
    $cafes = array(
        "nexus"=>array(
            "address" => "318 Campus Drive West, Stanford, CA",
            "full_name" => "NeXus Cafe",
            "opens" => 8,
            "closes" => 15
        ),
        "bytes"=>array(
            "address" => "350 Serra Mall, Stanford, CA",
            "full_name" => "Bytes Cafe",
            "opens" => 7,
            "closes" => 15
        ),
        "coolcafe"=>array(
            "address" => "328 Lomita Drive, Stanford, CA",
            "full_name" => "Cool Cafe",
            "opening_hour" => 11,
            "closing_hour" => 13
        )
    );

    print json_encode($cafes);

?>
```

Client-side

- Strategy: use $.getJSON to get the café list
- Build some s, use jQTouch to navigate between them

$.getJSON recap

```
Usage:
$.getJSON(url, data, callback); or
$.getJSON(url, callback);

so:
$.getJSON("/~mkrieger/doStuff.php",
          function(response) {
             console.log(response);
          });
```

Starting point

```
<head>
    <style type="text/css" media="screen">@import "../jqt/jqtouch.css";</style>
    <style type="text/css" media="screen">@import "../jqt/theme.css";</style>
    <script src="../jquery.js" type="text/javascript" charset="utf-8"></script>
    <script src="../jqt/jqtouch.js" type="text/javascript" charset="utf-8"></
script>
    <script type="text/javascript" charset="utf-8">
    var jQT = new $.jQTouch();
    var cgiPath = '/~mkrieger/cgi-bin/';
    $(document).ready(function(){
        // do stuff here
    });
</script>
</head>
<body>
    <div id="home" class="current">
        <ul id="cafelist"></ul>
    </div>
</body>
```

Demo

`index-1.html`

Exercise 2

Start from index-1 and have it fetch the data & write to the console

TextMate...

Fetching data

```
$(document).ready(function(){
    $.getJSON(cgiPath + 'getcafes.php', function(response){
        for (var cafe in response) {
            console.log(response[cafe]);
        }
    })
})
```

Demo

`index-2.html (watch the console)`

Exercise 3

For each café:

1. Create a with an id of the café's shortname
2. Create a <ul> inside that div for jQTouch's list view
3. Create two s inside each <ul>:
    1. Full name of the café
    2. Address
4. Append that <ul> to the
5. Append the to the <body>
6. Create an <li> with a link like such: Full Café Name and add it to the cafélist on the front page

Doing something interesting

```
$(document).ready(function(){
    $.getJSON(cgiPath + 'getcafes.php', function(response){
        for (var cafe in response) {
            var fullname = response[cafe].full_name;
            var address = response[cafe].address;
            var newDiv = $("<div id='" + cafe + "'></div>");
            var newList = $("<ul></ul>").appendTo(newDiv);
            $("<li>" + fullname + "</li>").appendTo(newList);
            $("<li>at: " + address + "</li>").appendTo(newList);
            newDiv.appendTo(document.body);
            $('<li><a href="#' + cafe + '">' + response[cafe].full_name
+ '</li></a>').appendTo('#cafelist');
        }
    })
})
```

Demo

`index-3.html`

Posting a Message

jQTouch & Forms

- By default, will AJAX-ify all forms
- Place an <a> with class="submit" inside the form and it will work to submit as AJAX
- The response will be loaded as the next page
- Override these defaults by passing in: formSelector: false to jQTouch initializer

Exercise 4

1. Add a new div that will be our submission form
2. Create a <form> inside that
3. Add form fields for a username and a review score
4. Add a type="hidden" input field so that we can also POST which café is being reviewed
5. Add a link at the bottom that has class="submit" so jQTouch knows that it should POST that form

The code

```
<form id='review-form' action="/~mkrieger/cgi-bin/savereview.php" method='POST'>
        <input type="hidden" name='cafeid' id="reviewing-cafe-id" />
        <ul class="edit rounded">
        <li>Review for: <span id="reviewing-cafe-name"></span></li>
              <li><input type="text" name="username" placeholder="Your name"
id="some_name" /></li>
        <li>
        <select name="score">
            <option value="none">How good was the food?</option>
            <optgroup label="Positive">
                <option value="5">Amazing</option>
                <option value="4">Pretty Good</option>
            </optgroup>
            <optgroup label="Negative">
                <option value ="2">Not very good</option>
                <option value ="1">Terrible</option>
            </optgroup>
        </select>
        </li>
    </ul>
    <a style="margin:0 10px;color:rgba(0,0,0,.9)" href="#" class="submit
whiteButton">Submit</a>
</form>
```

Let's add a form

`Demo: index-4.html`

Side note: dynamically setting the action=

```
$('#review-form').attr('action', cgiPath + 'savereview.php');
```

(in the $(document).ready function)

Saving the review

$_GET, $_POST, $_REQUEST

- PHP provides three arrays in a request, $_GET, $_POST, and $_REQUEST, which combines the two

Getting all values

```
<?php
    foreach ($_REQUEST as $key => $value) {
       print $key.": ".$value."<br/>";
    }
?>
```

demo: request.php

Checking if all fields are present

```
<?php
    $required_fields = array("firstname", "lastname", "age");
    $missing_fields = array();
    $response = array();
    foreach ($required_fields as $key => $value) {
        if (!isset($_REQUEST[$value])) {
            $missing_fields[] = $value;
        }
    }
    if (count($missing_fields) > 0) {
        $response['status'] = 'error';
        $response['errors'] = $missing_fields;
    } else {
        $response['status'] = 'success';
    }
    print json_encode($response);
?>
```

Demo

`required_fields.php`

Importing other PHP files

```
<?php include_once ('file.php'); ?>
```

Quick task: Make the $cafes object imported

```
// cafes.php:
$cafes = array(
    "nexus"=>array(
        "address" => "318 Campus Drive West, Stanford, CA",
        "full_name" => "NeXus Cafe",
        "opens" => 8,
        "closes" => 15
    ),
    "bytes"=>array(
        "address" => "350 Serra Mall, Stanford, CA",
        "full_name" => "Bytes Cafe",
        "opens" => 7,
        "closes" => 15
    ),
    "coolcafe"=>array(
        "address" => "328 Lomita Drive, Stanford, CA",
        "full_name" => "Cool Cafe",
        "opening_hour" => 11,
        "closing_hour" => 13
    )
```

getcafes2.php

```
<?php require_once ('cafes.php'); ?>

<?php
    print json_encode($cafes);

?>
```

Exercise 4

1. Grab the username, score, and which café was reviewed from the $_REQUEST
2. Also print out some HTML that will be shown as the success/error page

Saving the Review, beginning

```
<?php
    // in a real app, check these are present & valid
    $username = $_REQUEST['username'];
    $cafeid = $_REQUEST['cafeid'];
    $score = $_REQUEST['score'];
    $full_cafe_name = $cafes[$cafeid]['full_name'];
?>
```

Provide feedback

```
<div id="submit-results" style="padding: 20px">
Thanks for submitting your review of <?php print $full_cafe_name; ?>

<a style="margin:0 10px;color:rgba(0,0,0,.9)" href="#home"
class="backHome whiteButton">Back</a>
</div>
```

Demo so far

`savereview-beginning.php`

Database

SQLite

- File-based database (doesn't require running a separate server)
- PHP & Python have built-in support
- Supported on Stanford servers!
- Works fine for small apps & prototypes
- Docs: http://www.sqlite.org/lang.html

If you get file permission errors

- In Terminal or SecureCRT, do:

chmod 755 yourdbname.db

SQLite in PHP

- SQLiteDatabase class
- Takes a filename as its parameter

SQLite in PHP

```
$db = new SQLiteDatabase('cafe.db');
$all_reviews = $db->query("SELECT * FROM reviews");
//inserting
/* reviews' schema is: id, username, cafeid, score
$db->queryExec("INSERT INTO reviews(username,cafeid,score)
VALUES('mikeyk', 'nexus', 5)");
```

Check if table exists

```
// check if table exists, create if not

// use query when you need a result back,
// queryExec when you don't expect one back

if ($db = new SQLiteDatabase('cafe.db')) {
    $result = $db->query("SELECT name FROM sqlite_master WHERE
type='table' AND name='reviews'");
    if ($result->numRows() == 0) {
        $db->queryExec('CREATE TABLE reviews (id int, username text,
cafeid text, score int, PRIMARY KEY (id))');
    }
} else {
    die($err);
}
```

A brief note on SQL injection

- If input to SQL isn't properly escaped, user-submitted data could have bad side effects

Make input safe

```
$safe_name = sqlite_escape_string($_REQUEST['username']);
// prevent SQL injection attack!
```

Exercise 5

1. Take the data we received from the client and make it DB-safe
2. Insert it into the database

Actually insert

```
$username = sqlite_escape_string($_REQUEST['username']);
$cafeid = sqlite_escape_string($_REQUEST['score']);
$score = sqlite_escape_string($_REQUEST['score']);

$command = "INSERT INTO reviews(username, cafeid, score) VALUES('%s',
'%s', %d)";
$replaced = sprintf($command, $username, $cafeid, $score);
$db->queryExec($replaced);
```

Demo

`index-4.html`

Loading data

Strategy

- When a user clicks into a café, asynchronously fetch the reviews using jQuery's load function
- Have PHP read the Database and write the HTML to return

Event

- We want to watch for **pageAnimationEnd** and use jQuery's **.bind** function to listen for it

Exercise 6

1. Watch for pageAnimationEnd event to see that a café page has loaded
2. Use jQuery's **.load** function to populate a with data from the server

jQuery.load

```
$(selector).load(url, data, callback);
```

Live Editing

Javascript

```
$("<h2>Reviews</h2><div class='reviews'></div>").appendTo(newDiv);

newDiv.bind("pageAnimationEnd", function(){
    var cafeID = $(this).attr("id");
    $(".reviews", $(this)).load(cgiPath + 'loadreviews.php', {'cafeid':
cafeID});
})

// .load() fetches an HTML page and places its content into the element
// that called it. the first parameter is the URL,
// second are the GET parameters
```

Demo

`index-5.html`

PHP: Getting Request

```
$cafeid = sqlite_escape_string($_REQUEST['cafeid']);
```

Iterating through results in PHP

```
$result = $db->query("SELECT * FROM reviews WHERE cafeid='".
$cafeid."'");

while ($result->valid()) {
    $cur = $result->current();
    // do something with $cur, which is the result row
    $result->next();
}
```

Printing results

```
if ($db = new SQLiteDatabase('cafe.db')) {
    print "<ul class='rounded'>";
    $result = $db->query("SELECT * FROM reviews WHERE
cafeid='".$cafeid."'");
    while ($result->valid()) {
        $cur = $result->current();
        print "<li>".$cur['username']." rated it ".$cur
['score']."</li>\n";
        $result->next();
    }
    print "</ul>";
}
```

Bonus: Distance from each cafe

Geocoding Review

- Geocoding: taking real-world address, give latitude & longitude
- Reverse Geocoding: given lat/lon, guess a real-world address

In this case…

- We have addresses for cafés
- We have lat/lon for user using geolocation
- We need lat/lon for cafés

Demo

- Tinygeocoder's geocoder (we used their reverse geocoder last week):

http://tinygeocoder.com/create-api.php?q=353+Serra+Stanford+CA

Get lat/lon for cafe

```
// if we haven't already fetched the distance for this café
if (!$("#distance", cafeID).length) {
    $.getJSON("http://tinygeocoder.com/create-api.php?callback=?", {'q':
thisCafe.address}, function(data){
        var lat = data[0];
        var lon = data[1];
        getLocationAndDistance(lat, lon, cafeId);
    })
}
```

Now, user location

```
/* this should look familiar from last week */
function getLocationAndDistance(lat, lon, cafeId) {
    if(navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position){
            var dist = distanceBetweenPoints(lat, lon,
position.coords.latitude, position.coords.longitude);
            addDistanceInformation(dist, cafeId);
        });
    } else {
        // fake it
        window.setTimeout(function(){
            var dist = distanceBetweenPoints(lat, lon, 37.428337,
-122.175822);
            addDistanceInformation(dist, cafeId);
        }, 1000);
    }
}
```

Now, populate the data

```
function addDistanceInformation(distance, cafeId) {
    $("<li id='distance'>" + Math.floor(distance * 1000) + "m
away</li>").insertBefore($(".postreview", "#"+cafeId));
}
```

Demo

`index-6.html`

Q's?