Plant a Seed

- New way to think about a webpage.
- Initially, only a ‘seed’ is sent from the server to the client browser.
- In that seed is the AJAX code to then grow a ‘tree’ – in other words a full page.
- Not always the best answer, BUT you should know how to do this for cases where it is appropriate.
When loading, the animals make their sounds in some random order with delays between 1 and 10 seconds.
It Starts With a Seed

- Table ‘animals’ contains only header

```html
<body>
  <div style="padding-left: 8px">
    <h3 style="text-align: center">Watch this Page Build</h3>
    <p>The table will be filled in the order of AJAX responses for all the animals returned from a first AJAX call. The individual animals have artificially introduced random delays in responding in order to make obvious the asynchronis nature of the AJAX calls by which animals make noise.</p>
    <table id="animals">
      <tr>
        <th>Name</th>
        <th>Sound</th>
      </tr>
    </table>
    <p>Status of Animalia AJAX call: <span id="outp1">...</span></p>
  </div>
</body>
```
Where Animals Come From

- From an AJAX call of course.

Get used to the idea that the seed is general. As in this example, the seed has no idea how many items may be added to the table/page/list etc.
Need a JavaScript function to kickoff the process of ‘growing the tree’.

Here is that function using jQuery:

```javascript
var alist;

jQuery(document).ready(function() {
    jQuery.post("animalia.php", {}, function(data, status) {
        addRows(data);
        jQuery("#outp1").html(status);
    });
});
```
function addRows(lst) {
    var rt = "";
    var tab = document.getElementById('animals');
    var i = tab.rows.length;
    var len = lst.length;
    for (j = len - 1; j >= 0; j--) {
        rt = "<tr><td>");
        rt += "<a href=" + lst[j].url + ">"><" + lst[j].name + "</a>";
        rt += "</td>");
        rt += "<td id=" + lst[j].name + "_sound" > ... </td> </tr>";
        var rr = tab.insertRow(i);
        rr.innerHTML = rt;
    }
    for (j = 0; j < len; j++) {
        getSound(lst[j].name);
    }
}
AJAX begets AJAX

- AJAX is used to get the list of animals
- AJAX to get each animal sound.

Getting used to the result of one AJAX call in turn causing a variable number of additional AJAX calls may take just a little getting used to.
To Return a Sound

```php
<?php

header ( 'Content-Type: text/plain' );
header ( "Access-Control-Allow-Origin: *" );

$n = $_REQUEST ["a"];  

switch ($n) {
   case "ardvark":
      $s = "Grunt";
      break;
   case "bison":
      $s = "Mumble";
      break;
   default:
      $s = "Shhh";
      break;
}

sleep(rand(1,10));

echo $s;
?>
```

Which animal is being passed into the call.

To highlight asynchronous nature of AJAX animal sounds wait a random interval before responding.
Inside AJAX Sound Call

```javascript
function getSound(n) {
    jQuery.post("asound.php", {a : n}, function(data, status) {
        target = "#" + n + "_sound";
        jQuery(target).text(data);
    })
}
```

Target is the table data element ID that is updated by AJAX call.