158.258 Web Development

AJAX - Doing Things in the Background

Computer Science & Information Technology
School of Mathematical & Computational Sciences
College of Sciences, Massey University
(AKLI, DISD & MTUI)

Running a Script in the Background

It can be useful to be able to run scripts and update a page element WITHOUT refreshing the page.

This can enable you to:

- Dynamically update page element while the user is watching/using the page
    - *e.g. search box completions*
- create 'Single Page Apps' where the user is never aware of full page refreshes

This enables *desktop style* Web Apps

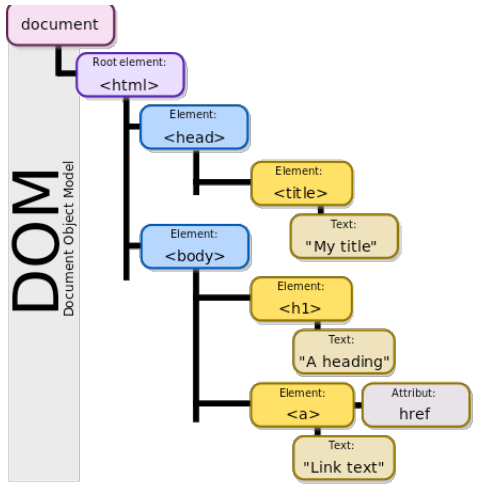
Page elements are redrawn by the browser

- effectively instantaneous redraws (no partial redraws)
- no *request* - *wait* - *redraw* lag

Ideally, user isn't aware of whether the app is web-based or not

This leads to things like *Chromebooks*

- hardware designed to just run web-apps inside Chrome browser
- virtually no local storage
- both apps and data is stored in the cloud.



DOM excerpt

Need:

- Mechanism to fetch data asynchronously (in the background)
- A way to continue to function when offline

This leads to what is known as AJAX.

AJAX - Asynchronous Javascript And XML:

- A set of techniques using many Web technologies on the client side to create *asynchronous* Web applications.
- Web applications can send data to and retrieve from a server asynchronously, in the background, without interfering with the display and behavior of the existing page.
- By decoupling the data interchange layer from the presentation layer, Ajax allows for Web pages, and, by extension, Web applications, to change content dynamically without the need to reload the entire page.
- In practice, modern implementations commonly substitute JSON for XML due to the advantages of being native to JavaScript.
  – See AJAX Programming on Wikipedia.

Querying Apple's iTunes Store using JSON

Apple are rather generous in making the iTunes Store searchable **without requiring an API key**.

Details of various sorts of media can be retrieved:

- music
- movies
- eBooks
- audio books

Fetching data using JSON

This is a Python program that uses the Python 'requests' module to search Apple's iTunes Store

- we're goint to turn it into a web page

```
    # Search Apple's iTunes store and get results as JSON data,
# which gets turned into a list of dictionaries
# See https://www.apple.com/itunes/affiliates/resources/documentation/itunes-store-web-service-search-api.html#understand
  author    = '158258 Dev Team'
# Make sure you have imported the following modules
import json, requests
print("\n*** 159.171 - Search Apple's iTunes Store ***\n")
searchTerm  = input("1st Term: ")
searchTerm2 = input("2nd Term: ")
if searchTerm2:
searchTerm = searchTerm + '+' + searchTerm2
#search_url = "https://itunes.apple.com/search?entity=audiobook&term=" + searchTerm
search_url = "https://itunes.apple.com/search?term=" + searchTerm
response = requests.get(search_url, timeout=10)
print(response.text)
data = response.json()   # unpack JSON encoded data
print("Unpacked JSON data\n", data)
print ("No of results", len(data["results"]))
for entry in range(15):             # only display the first 15 results
    item = data["results"][entry]
    print("-"*80)
    print (item["artistName"])
    if "trackName" in item:
       print('>>> ', item["trackName"])
    print("Preview Track    ", item["previewUrl"],
          "\nArtwork          ", item["artworkUrl100"])
    if "trackViewUrl" in item:
```

```
        print("View iTunes page ", item["trackViewUrl"])
    if "description" in item:
        description = item["description"].replace("<br />", "\n").replace(". ", ".\n")
        print(description)
```

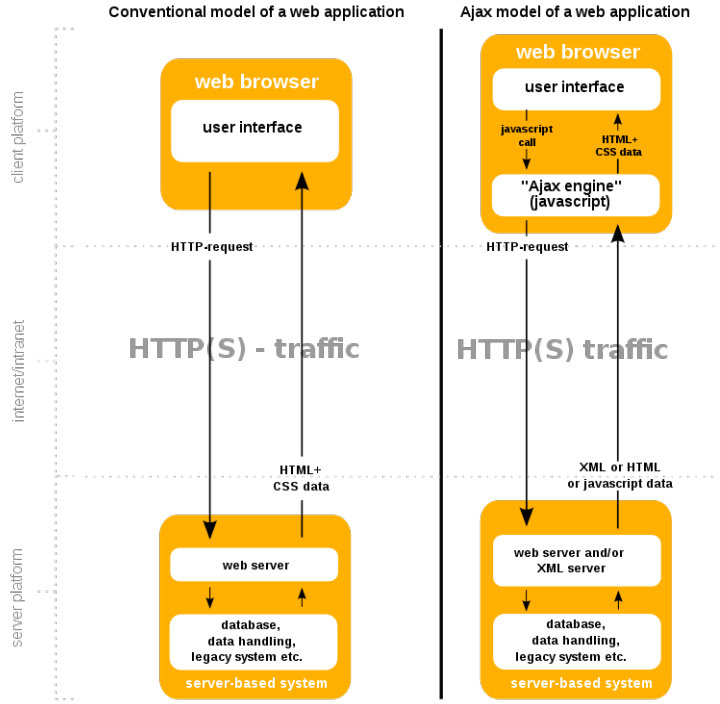
Limitations with the above implementation

The above Python program certainly works, but is limited:

- no pictures (album covers, artist photos) are shown
- the links to audio sample are indirect
- the searching is somewhat 'old-style'
  –> type the whole query and press <enter>

AJAX client-server model

The control flow differs from the usual *'request then display'* page model:



By DanielSHaischt, via Wikimedia Commons [CC BY-SA 3.0], via Wikimedia Commons

AJAX - XMLHttpRequest()

The Javascript technology uses the **XMLHttpRequest()** object

This name is a misnomer

- the item being transferred isn't necessarily XML
- it can be JSON instead of XML
    - *XML is an older more structured method of textualising data (like JSON)*
    - *it's widely used*

Setting up an AJAX request

1. create an **XMLHttpRequest()** object
2. define the server URL and transfer method (*get* or *post*)
    - *if using get, append any data to send*
3. define a callback function to handle the response when/if it arrives
4. invoke the *send()* method to start the process

carry on …

At some state the response should arrive and the callback function will be called

XMLHttpRequest: methods

Methods

1. **xhr = new XMLHttpRequest()**
2. **xhr.open('get', Server-URL, async?)** or **xhr.open('post', Server-URL, async?)**
    - *async should be true or false but may be omitted*
3. **xhr.onreadystatechange** = function() { define callback }
4. **xhr.send()** - called after 1 - 3, start request …

These are involved in setting up the request

XMLHttpRequest: properties

When the callback function is invoked, these properties will be available:

Properties

- **xhr.readyState** - valid in callback
    - *the value we want is **4** (DONE) - request has completed*
- **xhr.status** - HTTP status code
    - *the value should be **200** for success*
- **xhr.responseText** - data received from the server
    - *this will be a string*

XMLHttpRequest skeleton

```
    <script>
let xhr = new XMLHttpRequest();
xhr.open('get', 'send-ajax-data.php');

// Track the state changes of the request.
xhr.onreadystatechange = function () {
    let DONE = 4;   // readyState 4 means the request is done.
    let OK = 200;   // status 200 is a successful return.
    if (xhr.readyState === DONE) {
        if (xhr.status === OK) {
            console.log(xhr.responseText);       // This is the returned text
        } else {
            console.log('Error: ' + xhr.status); // A problem, show erro msg
        }
    }
};
xhr.send()  // Start the process
</script>
```

Adapted from https://en.wikipedia.org/wiki/Ajax_(programming)

Modifying above code to query Apple's iTunes Store

```
   <script>

function queryiTunesStore v1() {
   let xhr = new XMLHttpRequest();
   let searchTerm = 'Madonna'
   let searchReqURL = "https://itunes.apple.com/search?term=" + searchTerm
   xhr.open('get', searchReqURL);

   xhr.onreadystatechange = function () {
       let DONE = 4; // readyState 4 --> request completed
       let OK = 200; // HTTP status code of 200 --> success
       if (xhr.readyState === DONE) {
           if (xhr.status === OK) {
              console.log(xhr.responseText); // 'This is the returned text.'
              document.getElementById('ajax1').innerHTML = xhr.responseText;
           } else {
               alert('Error: ' + xhr.status);        // Show error msg.
               console.log('Error: ' + xhr.status);  // and add to log
           }
       }
   };

   xhr.send()   // Run the process
}
</script>
<button onClick="queryiTunesStore v1()"> Find songs by Madonna</button>
JSON Result: <p id=ajax1> Show Ajax Response</p>
```

Find songs by Madonna JSON Result:

Show Ajax Response

Cleaning up the code (v2)

Having the callback function as part of the assignment is ugly.

- refactor the code

```
    <script>
// Start a background request
function AJAX_queryiTunesStore_v2(searchTerm, callbackFunction) {
   xhr = new XMLHttpRequest();  // IMPORTANT - 'xhr' IS GLOBAL
   let searchReq = "https://itunes.apple.com/search?term=" + searchTerm
   xhr.open('get', searchReq);
   xhr.onreadystatechange = callbackFunction
   xhr.send()   // Run the process
}

function iTunesResponseCallback_v2() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          document.getElementById('ajax2').innerHTML = xhr.responseText;
       } else {
          alert('Error: ' + xhr.status); // An error occurred during the request.
       }
   }
}

function searchITunesStore_v2(){
  searchTerm = 'Madonna'
  AJAX_queryiTunesStore_v2(searchTerm, iTunesResponseCallback_v2)
}
</script>
<button onClick="searchITunesStore_v2()"> Find songs by Madonna - tidied</button>
JSON Result: <p id=ajax2> Show Ajax Response</p>
```

Find songs by Madonna - tidied JSON Result:

Show Ajax Response

Livening up the search - interactive input (v3)

Modify the code to allow the user to input the search terms

```
    <script>
// Setup and callback omitted ...

function searchITunesStore v3(){
  searchTerm = document.getElementById('searchbox').value
  AJAX_queryiTunesStore_v3(searchTerm, iTunesResponseCallback_v3)
}
</script>


<input type='text' id='searchbox'
         placeholder='Search for what'
         autofocus
         onChange="searchITunesStore_v3()">

<p id='resp3'>Last update ...</p>
<p>JSON Result</p>
<p id=ajax3> Show Ajax Response here</p>
```

Don't run this live - search keys interact with Slidy's Presentation mode

use code from Stream:

AJAX_demos/3_livening_the_search_InputBox.html

but if you insist! (press <enter> after entering the search term):

Search for what

Last update ...

JSON Result

Show Ajax Response

OnChange() event handler isn't the correct one

onChange() event fires only when the input loses focus

- for text box, this is when <enter> key is pressed
- no change until the end

This isn't the desired behaviour - to update on each typed character …

Using onInput() fires after a change

- this is per character for text boxes

>> AJAX_demos/3a_livening_the_search_InputBox_JSON_onInput.html

```
    <script>
// Start a background request
function AJAX_queryiTunesStore_v3(searchTerm, callbackFunction) {
   xhr = new XMLHttpRequest();  // IMPORTANT - 'xhr' IS GLOBAL
   let searchReq = "https://itunes.apple.com/search?term=" + searchTerm
   xhr.open('get', searchReq);
   xhr.onreadystatechange = callbackFunction
   xhr.send()   // Run the process
}

//
function iTunesResponseCallback_v3() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          document.getElementById('resp3').innerHTML = new Date()
          document.getElementById('ajax3').innerHTML = xhr.responseText;
       } else {
          alert('Error: ' + xhr.status); // An error occurred during the request.
       }
   }
}

function searchITunesStore_v3(){
  searchTerm = document.getElementById('searchbox').value
  AJAX_queryiTunesStore_v3(searchTerm, iTunesResponseCallback_v3)
}
</script>
<h2>Searching iTunes Store using onChange() event handler</h2>
```

```
<input type='text' id='searchbox'
         onInput="searchITunesStore v3()"
         placeholder='Search for what'>
<p id='resp3'>Last update ...</p>
<p>JSON Result</p>
<p id=ajax3> Show Ajax Response here</p>
```

What's in the JSON object - RAW?

Usually the contents of the JSON data will be defined by it's provider (the server)

However, it can be found by inspection if the application isn't critical:

Raw

{ "resultCount":50, "results": [{"wrapperType":"track", "kind":"song", "artistId":217015, "collectionId":963405017, "trackId":963405018, "artistName":"Andrea Bocelli", "collectionName":"Bocelli", "trackName":"Con Te Partiro", "collectionCensoredName":"Bocelli", "trackCensoredName":"Con Te Partiro", "artistViewUrl":"https://itunes.apple.com/us/artist/andrea-bocelli/id217015?uo=4", "collectionViewUrl":"https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4", "trackViewUrl":"https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4", "previewUrl":"https://audio-ssl.itunes.apple.com/apple-assets-us-std-000001/Music5/STUFF-DELETED-HERE", "artworkUrl30":" 30x30bb.jpg", "artworkUrl60":" 60x60bb.jpg", "artworkUrl100":" 100x100bb.jpg", "collectionPrice":9.99, "trackPrice":1.29, "releaseDate":"1995-03-17T08:00:00Z", "collectionExplicitness":"notExplicit", "trackExplicitness":"notExplicit", "discCount":1, "discNumber":1, "trackCount":10, "trackNumber":1, "trackTimeMillis":250200, "country":"USA", "currency":"USD", "primaryGenreName":"Pop", "isStreamable":true}, {"wrapperType":"track", "kind":"song", "artistId":217015, "collectionId":266952045, "trackId":266952347, "artistName":"Andrea Bocelli & Céline Dion", "collectionName":"The Best of Andrea Bocelli - Vivere (Bonus Track Version)", "trackName":"The Prayer", "collectionCensoredName":"The Best of Andrea Bocelli - Vivere (Bonus Track Version)", "trackCensoredName":"The Prayer", "collectionArtistName":"Andrea Bocelli", "artistViewUrl":"https://itunes.apple.com/us/artist/andrea-bocelli/id217015?uo=4", "collectionViewUrl":"https://itunes.apple.com/us/album/the-prayer/id266952045?i=266952347&uo=4", "trackViewUrl":"https://itunes.apple.com/us/album/the-prayer/952347&uo=4", "previewUrl":"https://audio-ssl.itunes.apple.com/apple-assets-us-std-000001/Music/STUFF-DELETED-HERE", "artworkUrl30":" 30x30bb.jpg", "artworkUrl60":" 60x60bb.jpg", "artworkUrl100":" 100x100bb.jpg", "collectionPrice":11.99, "trackPrice":1.29, "releaseDate":"1998-10-30T08:00:00Z",

What's in the JSON object?

The result is an object, containing a count, and a list of objects - each one has details about one music track.


```
 { "resultCount":50,
"results": [
 {"wrapperType":"track",
  "kind":"song",
  "artistId":217015,
  "collectionId":963405017,
  "trackId":963405018,
  "artistName":"Andrea Bocelli",
  "collectionName":"Bocelli",
  "trackName":"Con Te Partiro",
  "collectionCensoredName":"Bocelli",
  "trackCensoredName":"Con Te Partiro",
  "artistViewUrl":"https://itunes.apple.com/us/artist/andrea-bocelli/id217015?uo=4",
  "collectionViewUrl":"https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4",
  "trackViewUrl":"https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4",
  "previewUrl":"https://audio-ssl.itunes.apple.com/apple-assets-us-s/XXXXX- REALLY LONG - Chopped",
  "artworkUrl30":"http://is1.mzstatic.com/image/thumb/Music3/XXXXXXXXXX8-c04c-e3b/source/30x30bb.jpg",
  "artworkUrl60":"http://is1.mzstatic.com/image/thumb/Music3/XXXXXXXXXX8-c04c-e3b/source/60x60bb.jpg",
  "artworkUrl100":"http://is1.mzstatic.com/image/thumb/Music3/XXXXXXXXX8-c04c-e3/source/100x100bb.jpg",
  "collectionPrice":9.99,
  "trackPrice":1.29,
  "releaseDate":"1995-03-17T08:00:00Z",
  "collectionExplicitness":"notExplicit",
  "trackExplicitness":"notExplicit",
  "discCount":1,
  "discNumber":1,
  "trackCount":10,
  "trackNumber":1,
  "trackTimeMillis":250200,
```

```
      "country":"USA",
      "currency":"USD",
      "primaryGenreName":"Pop",
      "isStreamable":true
     },

     {"wrapperType":"track",         <<<< START OF NEXT OBJECT IN THE LIST
     }
     ... many more
   ] // end of 'results' list
}    // end of the returned object
```

What's in the JSON object - finding the property names

Removing quotes around keys, and reformatting helps:

```
   { "resultCount":50,
  "results": [
    {wrapperType       :"track",
     kind              : "song",
     artistId          : 217015,
     collectionId      : 963405017,
     trackId           : 963405018,
     artistName        : "Andrea Bocelli",
     collectionName    : "Bocelli",
     trackName         : "Con Te Partiro",
     trackCensoredName : "Con Te Partiro",
     artistViewUrl     : "https://itunes.apple.com/us/artist/andrea-bocelli/id217015?uo=4",
     collectionViewUrl : "https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4",
     trackViewUrl      : "https://itunes.apple.com/us/album/con-te-partiro/id963405017?i=963405018&uo=4",
     previewUrl        : "https://audio-ssl.itunes.apple.com/apple-assets-us-s/XXXREALLY LONG - Chopped",
     artworkUrl30      : "http://is1.mzstatic.com/image/thumb/Music3/XXX8-c04c-e3b/source/30x30bb.jpg",
     artworkUrl60      : "http://is1.mzstatic.com/image/thumb/Music3/XXX8-c04c-e3b/source/60x60bb.jpg",
     artworkUrl100     : "http://is1.mzstatic.com/image/thumb/Music3/XXX8-c04c-e3/source/100x100bb.jpg",
     etc ...
    },
}    // end of the returned object
```

Unpacking the JSON - in detail - outer level

Using JSON.parse(response_Text) to get the Javascript object

```
    function handleiTunesResponseCallback_v4() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          let response  = JSON.parse(xhr.responseText);

          let numResults = response.resultCount

          let resultList = response.results

           >>>> 'resultList' is the array/list of track info objects
          }
       } else {
          alert('Error: ' + xhr.status); // An error occurred during the request.
       }
   }
}
```

Unpacking the details about each track

Using JSON.parse(response_Text) to get the Javascript object

```
 function handleiTunesResponseCallback_v4() {  //
if (xhr.readyState === 4 ) {  // DONE
    if (xhr.status === 200) { // OK
       let response  = JSON.parse(xhr.responseText);
       let numResults = response.resultCount
       document.getElementById('resp4').innerHTML = new Date() + ', #Results='+numResults

       let resultList = response.results
       let s = ''
       for (i=0; i<numResults; i++){
          let item = resultList[i]   //PICK OUT ONE TRACK

          s = s + item.artistName + ' -- ' + item.trackName + '<br>'

          document.getElementById('ajax4').innerHTML = s
       }
    } else {
       alert('Error: ' + xhr.status); // An error occurred during the request.
    }
 }
}
```

Livening search - unpacking the JSON response - demo

Update a paragraph with artist and track name as letters are typed …

```
    <script>
// Start a background request
function AJAX queryiTunesStore v4(searchTerm, callbackFunction) {
   xhr = new XMLHttpRequest();  // IMPORTANT - 'xhr' IS GLOBAL
   let searchReq = "https://itunes.apple.com/search?term=" + searchTerm
   xhr.open('get', searchReq);
   xhr.onreadystatechange = callbackFunction
   xhr.send()   // Run the process
}

function handleiTunesResponseCallback v4() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          let response  = JSON.parse(xhr.responseText);
          let numResults = response.resultCount
          document.getElementById('resp4').innerHTML = new Date() + ', #Results='+numResults
          let resultList = response.results
          let s = ''
          for (i=0; i<numResults; i++){
             let item = resultList[i]
             s = s + item.artistName + ' -- ' + item.trackName + '<br>'
             document.getElementById('ajax4').innerHTML = s
          }
       } else {
          alert('Error: ' + xhr.status); // An error occurred during the request.
       }
   }
}
```

```
function searchITunesStore v4(){
  searchTerm = document.getElementById('searchbox').value
  AJAX_queryiTunesStore_v4(searchTerm, handleiTunesResponseCallback_v4)
}
</script>
<input type='text' id='searchbox' onInput="searchITunesStore_v4()"
       placeholder='Search for what' autofocus>
<p id='resp4'>Last update ...</p>
<p>JSON Result</p>
<p id=ajax4> Show Ajax Response</p>
```

>>> AJAX_demos/4_livening_search_unpackJSON.html (will be on Stream)

Livening search - with images

Use the **artworkUrl100** property build an 'img' element to display the album cover:

```
    <script>
function handleiTunesResponseCallback v4() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          let response  = JSON.parse(xhr.responseText);
          let numResults = response.resultCount
          document.getElementById('resp4').innerHTML = new Date() + ' - No. of results: '+numResults
          let resultList = response.results

          let s = ''
          for (i=0; i<numResults; i++){
             let item = resultList[i]
             s = s + '<img src="' + item.artworkUrl100 +'">';
             s = s + item.artistName + ' -- ' + item.trackName;
             s = s + '<p>'
          }
          document.getElementById('ajax4').innerHTML = s
       }
   }
}
```

>>> AJAX_demos/5_livening_search_with_links.html

Livening Search - with audio

Now add an HTML5 <audio> element so we can play the track previews

Create a function to build the audio link

```
    <script>
function audioPlayer(audioURL){
    // let s = '<a href="' + audioURL + '" > Preview</a>'

    let s = ''
    s += '<audio controls>\n'
    s +='   <source src="' + audioURL + '" type="audio/mp3">\n'
    s +=    'Browser does not support audio player\n'
    s += '</audio>'
    return s
}
```

Livening Search - with audio

Now use the function in the callback so we can play the track previews

```
    function audioPlayer(audioURL){
    let s = ''
    s += '<audio controls>\n'
    s +='   <source src="' + audioURL + '" type="audio/mp3">\n'
    s +=    'Browser does not support audio player\n'
    s += '</audio>'
    return s
}

function handleiTunesResponseCallback v4() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          let response  = JSON.parse(xhr.responseText);
          let numResults = response.resultCount
          let resultList = response.results

          let s = ''
          for (i=0; i<numResults; i++){
             let item = resultList[i]
             s = s+ '<img src="' + item.artworkUrl100 +'">'
             s = s + item.artistName + ' -- ' + item.trackName
             s = s + audioPlayer(item.previewUrl)+ '<br>'
          }
          document.getElementById('ajax4').innerHTML = s
       }
   }
}
</script>
```

>>> AJAX_demos/6_livening_search_with_audio.html

Refine the results page using a table layout

Divs would be better as they could flow for responsive layout, but for this example tables are simpler.

- create a table with two cells/row
- first cell has the cover image, the second has the text and the audio player

```
    <script>
function handleiTunesResponseCallback() {  //
   if (xhr.readyState === 4 ) {  // DONE
       if (xhr.status === 200) { // OK
          let response  = JSON.parse(xhr.responseText);
          let numResults = response.resultCount
          document.getElementById('resp4').innerHTML = new Date() + ', #Results='+numResults
          let resultList = response.results

          let s = '<table>'
          for (i=0; i<numResults; i++){
             let item = resultList[i]
             s +='<tr> <td style="padding:10px">'
             s += '<img src="' + item.artworkUrl100 +'">'
             s += '</td><td>'
             s += '<p><b>' +item.trackName + '</b> by ' + item.artistName +'</p>'
             s += audioPlayer(item.previewUrl)
             s += '</td></tr>'
          }
          s += '</table>'
          document.getElementById('ajax4').innerHTML = s
       }
   }
}
```

>>> AJAX_demos/7_livening_search_tablised.html