

# Project 2 Feedback

## Project 2 Feedback

Grades: think of + as "perfect", ✓ as "good," and ✓– as "not good".

Correctness: +

Design: ✓

Style: ✓

Scope: +

Comments

## Correctness Audit

Can users select different routes?

Are polylines drawn accurately?

Is each station displayed in the correct location?

Are arrival/departure times accurate?

Does zooming/panning the map break functionality?

Are any Javascript errors or warnings thrown?

Technical Requirements

Correctness Grade: +

## Design Audit

Is any logic convoluted or overly complex?

Is code copy/pasted where refactoring into functions would be more appropriate?

How is static data cached?

What is the runtime of a cache lookup?

How much space is consumed by the cache?

How robust is the cache structure?

How is data from the API fetched and parsed?

How efficiently are stations added and polylines drawn?

If applicable, how effectively were third-party libraries used?

Design Grade: ✓

## Style Audit

Is the HTML well-indented?

Is the PHP well-indented and commented?

Are variable and function names appropriate and descriptive?

Style Grade: ✓

## Scope Audit

Feature Requirements

Scope Grade: +

**Grades: think of ✓+ as "perfect", ✓ as "good," and ✓– as "not good".**

## **Correctness: ✓+**

Correctness: To what extent is your code consistent with our specifications and free of bugs?

## **Design: ✓**

Design. To what extent is your code written well (i.e., clearly, efficiently, elegantly, and/or logically)?

## **Style: ✓**

Style. To what extent is your code readable (i.e., commented and indented with variables aptly named)?

## **Scope: ✓+**

Scope. To what extent does your code implement the features required by our specification?

## **Comments**

A pretty good submission! Thank you!

# Correctness Audit

## Can users select different routes?

Users can select different routes just fine.

## Are polylines drawn accurately?

Polylines are drawn accurately.

## Is each station displayed in the correct location?

Each station is displayed in the correct place.

## Are arrival/departure times accurate?

Arrival and departure times are accurate.

## Does zooming/panning the map break functionality?

Zooming and panning does not break the functionality of the application.

## Are any Javascript errors or warnings thrown?

No javascript errors or warnings are thrown by the application.

## Technical Requirements

- You're welcome to develop your site on any computer using any IDE or text editor, but you must ensure that it works within the CS50 Appliance at a URL of `http://localhost/~jharvard/project2/` when uploaded (as via SCP or SFTP) to `/home/jharvard/public_html/project2/`.
- Any PHP files must be `chmod'd 600`.
- Your site must use version 3 of the Google Maps JavaScript API.
- It suffices to use only the Real BART API (<http://api.bart.gov/docs/overview/>), but you are welcome to use the Simple ETA Feed (<http://www.bart.gov/schedules/developers/etas.aspx>) and/or the GTFS feed ([http://www.bart.gov/dev/schedules/google\\_transit.zip](http://www.bart.gov/dev/schedules/google_transit.zip)).
- You should cache locally (on disk or in a MySQL database) data that does not change every minute (e.g., routes and their stations). Your mashup should only query the BART API or (Simple ETA Feed) for real-time departure (or arrival) times.
- You are welcome, but not required, to use any of the JavaScript libraries recommended in Lecture 6's slides.
- Your markup language should be valid (or "tentatively" valid) HTML 4.01 Strict, HTML 4.01 Transitional, HTML 5, XHTML 1.0 Strict, or XHTML 1.0 Transitional (or XHTML) should be valid, as per <http://validator.w3.org/>, unless some feature of your site requires

otherwise (for the sake of some browser); explain in HTML (or XHTML) comments any intentional invalidities. Your HTML (or XHTML) should also be as pretty---printed as possible. Your CSS need not be valid.

*Your page is almost valid, but you left out one small thing that causes over 80 errors on your page. In the script tag where you embed the JSON, you forgot to include CDATA sections:*

```
<script type="text/javascript">
var coord1 = // json data here
//]]&gt;</pre>
</div>
<div data-bbox="143 255 879 526" data-label="List-Group">
<ul>
<li>• Any JavaScript or PHP code that you write must be extensively commented and be as pretty---printed as possible.</li>
<li>• You may use a WYSIWYG editor to generate XHTML and/or CSS that you would like to use in your site.</li>
<li>• If you incorporate or adapt snippets of code from the Web into your project (e.g., examples from php.net), cite the code's origins with PHP comments.</li>
<li>• If you incorporate images from the Web into your project, cite the images' with XHTML comments.</li>
<li>• Your website must appear and behave the same on at least two major browsers, namely:
      <ul>
<li>• Chrome 12 or higher</li>
<li>• Firefox 4 or higher</li>
<li>• Internet Explorer 8 or higher</li>
<li>• Opera 11 or higher</li>
<li>• Safari 5 or higher</li>
</ul>
</li>
<li>• Your project submission must have a readme in a file called README that lives in the same folder as the rest of your project.</li>
</ul>
</div>
<div data-bbox="111 550 425 572" data-label="Section-Header">
<h2>Correctness Grade: ✓+</h2>
</div>
<div data-bbox="111 578 859 633" data-label="Text">
<p>Correctness: To what extent is your code consistent with our specifications and free of bugs? Assume a default score of ✓. If code is truly 100% bug-free, upgrade to ✓+. If code manifest many bugs and really leaves lots of room for improvement, downgrade to ✓−.</p>
</div>
<div data-bbox="111 647 886 889" data-label="Text">
<p><i>Thank you for your care with this project. Your code is almost completely consistent with our specifications and free of bugs. However, the the project 2 specification states that your markup should be valid or tentatively valid, "unless some feature of your site requires otherwise (for the sake of some browser)," in which case you are to "explain in HTML (or XHTML) comments any intentional invalidities." I read your Readme.txt, and it does not mention problems with validation. I even looked at <a href="http://help.cs75.net">help.cs75.net</a> and in my email inbox to see if you had any questions related to validation; there were none. There is technically only one mistake in your code that causes your validation errors, and if I had any indication that you had tried to validate your code as per the specification, I would probably would have let it slide. As it is, though, I have no indication that you put any work into validation, so I cannot vouch that your code is truly 100% consistent with our specification and upgrade the score to ✓+ for this axis. Thanks for your hard work, though, and keep in mind that the scores will be amortized across TF, Grade Axis, and Class, and I am just as careful about adhering to the spec and rubric with other students as I</i></p>
</div>
```

have been with you.

*EDIT: I discovered inside your index.php that you undoubtedly did validate, and somehow must have just validated without the rendered javascript. This is an important mistake: you should use a validator such as [Firefox's HTML Validator](#) so that you can validate using the complete rendered HTML on each page. I can tell that you attempted to validate, and overlooked only one thing, so I have adjusted your correctness grade to ✓+.*

## Design Audit

### Is any logic convoluted or overly complex?

The logic is not convoluted or overly complex.

### Is code copy/pasted where refactoring into functions would be more appropriate?

Code is refactored into functions where it would be more appropriate. However, one function per php file is pretty sparse, and makes the directory unnecessarily clogged with files that are essentially function names.

### How is static data cached?

Static data is cached in a database.

### What is the runtime of a cache lookup?

The runtime of a cache lookup is about 20 seconds. This does not feel like a cache lookup. Why?

### How much space is consumed by the cache?

The cache consumes about 12 kb.

### How robust is the cache structure?

*The cache structure is not robust. If any of the routes or stations had to be changed, the entire thing would have to be repopulated. It is also not really a relational model; routes contain a csv string that refers to station ids, instead of adding a third column for ordinality.*

### How is data from the API fetched and parsed?

The data from the API is fetched and parsed and put into the MySQL cache from updaterroute.php and updatecache.php. When the route is selected and drawn to the page, the

Student: Hsieh, Crystal Jenfei: crystalhsieh7@gmail.com, crystalhsieh7

cache data is pulled from the database along with the real-time queries for each station. This causes route selection to be quite lengthy, sometimes taking thirty seconds or more. Real-time updates are snappy for all stations on a route once loaded, but the information gets stale and the user must wait for a whole page reload to get new info.

## **How efficiently are stations added and polylines drawn?**

There are serious inefficiency issues with the design. The stations and polylines are added in javascript when the page for a certain route loads, without ajax. The php compiles javascript variable names that refer to the station itself.

## **If applicable, how effectively were third-party libraries used?**

Third party libraries were not used.

### **Design Grade: ✓**

Design. To what extent is your code written well (i.e., clearly, efficiently, elegantly, and/or logically)? Assume a default score of ✓. If code's design leaves no room whatsoever for improvement, upgrade to ✓+. If code manifests more than a few poor design decisions, downgrade to ✓−.

# Style Audit

## Is the HTML well-indented?

The HTML is reasonably well-indented.

## Is the PHP well-indented and commented?

The PHP is reasonably well-indented and commented, but the comments don't always explain what is going on.

## Are variable and function names appropriate and descriptive?

PHP variable names and function names are reasonably descriptive (example: function `get_coord($abbrev){` in `coord_funct`) most of the time, but sometimes, one must read a comment to know that `$number` signifies route number; why not just call it `$route_id`?

## Style Grade: ✓

Style. To what extent is your code readable (i.e., commented and indented with variables aptly named)? Assume a default score of ✓. If code looks beautiful (i.e., perfectly indented, thoroughly commented), upgrade to ✓+. If code is a mess or uncommented, downgrade to ✓−.

# Scope Audit

## Feature Requirements

- Your site's homepage must display an embedded Google Map, centered and zoomed in on San Francisco.
- Your site's homepage must provide the user with a way of selecting one BART route at a time.
- Once selected, a route should be drawn as polylines on the map in the route's official color, with markers representing each of that route's stations.
- Each station, when clicked, should trigger an info window that summarizes the next trains departing from (or arriving at) that station.



## Scope Grade: ✓+

Scope: To what extent does your code implement the features required by our specification?

Assume a default score of ✓+. If code fails to satisfy one or some of the spec's requirements, downgrade to ✓. If code fails to satisfy most of the spec's requirements, downgrade further to ✓-.

*Your site matches the specifications - thanks for your effort!*



1. Crystal Hsieh
2. CS-75
3. README FILE
- 4.
5. Hello, my name is Crystal Hsieh, and this is my BART project.
- 6.
7. Please use Mozilla Firefox and Chrome to view my project.
- 8.
9. Visuals
- 10.
11. When starting this project, I knew I wanted to have the Google Maps part of the page to be the majority of the page because I think that is the main part of the website. I decided to use CSS to make it basically 80% of the page, with a info bar to the left of it. Rather than using the BART logo from their website, I used MS Paint on my computer to create my own banner at the top of the control panel. I tried to use similar colors to the actual website, so that it looks cohesive with the actual BART system. I also added BART logos for all the markers on the website, so that it adds a more personal look then just the regular red markers. 
- 12.
13. I also noticed that because the map is at a certain zoom level, some of the markers overlap. At first I tried to make the markers look very thin so that they could all be seen at once. Later, I realized that when you zoom in to a particular area, the station markers spread out and are automatically smaller. After editing the BART logo in MS Paint so that it had a pointer at the bottom, I used a size that I thought was suitable for the map.
- 14.
15. Also, on the left side of the website is the control panel, so when a user selects the radio button of a particular route, the radio button automatically submits itself (via Javascript) and the page is reloaded with the chosen route. I chose to use radio buttons because unlike checkmarks, radio buttons imply that only one option is selected at a time. Also, I wanted the page to instantly change after the user has made their decision, so I used onclick to create that effect. With the use of the value attribute for radio buttons, I was also able to send information of which route was chosen. 
- 16.
17. Lastly, in order to show all my information in the info bar without having to shrink the font to a barely readable size, I decided to implement a scroll bar so that the information could be accessed with the Google-maps Map intact.
- 18.
19. Performance
- 20.
21. After loading the website, the user will first see all the different routes at once, but will not see stations. When I had originally created the code for the site, I loaded all the routes and stations at once, and then basically I would just hide the other stations that weren't needed for a specific route. The load time for my code would take a lot longer than was necessary.
- 22.
23. Now I have decided that when a user clicks a station, the polyline for the route is made, and then all the station markers for that particular route are made as well. This still takes a short while to load, but it is significantly faster to how long it would take to load all the stations every time the radio button was clicked.
- 24.
25. Design Decisions
- 26.
27. Being more familiar with PHP than Javascript, I decided to use PHP to create the Javascript necessary for this assignment. With the use of SimpleXML and XPath, I was able to parse all the data from the BART API into the website.
- 28.
29. The specifications stated that static info had to be cached either locally or in a MYSQL database, so I created updatestation.php and updaterroute.php, which take the information given from XML documents in the BART API and input it into the MYSQL database tables called routes and stations. In the php documents, it checks whether or not there is input inside the tables. If there isn't, then it will input the

I can tell you care about the project!

awesome

information. If it is, then using the php's last modified date, it will update itself weekly after that time. This is implemented using modulo. I also realized I would need the coordinates of the stations in order to plot them, so I used the XML file for each station to cache the coordinates in the MYSQL database table called 'stations'. This will also be updated weekly because station information like their address don't change very often.

- 30.
31. When a user clicks on a radio button, it sends the information of which route was chosen to clicked.php, which takes the route chosen, selects it from the database, and helps create the particular polyline and station markers.
- 32.
33. The specifications also required real-time data from BART, and I chose to use the simple XML feed, or ETA. I chose to use ETA because I liked how the ETA XML file gave the times for the next three arriving trains. The ETD XML file only gave one time, which didn't seem as helpful. I almost implemented both, but when I was checking the info windows, the ETA information and the ETD information tended to clash (e.g. Train A arriving at 7pm, Train A departing at 7pm). It also seemed like two estimates of arrivals was overkill.
- 34.
35. Trying to make my code easier to manage, I tried to keep the functions in separate php files. Because my website is basically one large page, I included all my php files/functions in the header.php file.
- 36.
37. Quick Summary of my Files and their Purpose:
- 38.
39. arrivals.php --> is a function that takes abbreviation of the station, queries the XML from the BART API and returns real-time arrival times
- 40.
41. clicked.php --> takes in the number of the chosen route, and creates the routes and stations, placig them on the map.
- 42.
43. coord\_funcnt.php --> php function made to query XML on the BART API and find out the coordinates of specific stations
- 44.
45. database.php --> basically a php document to be used for connecting to the database
- 46.
47. getlatlon.php --> takes a specific abbreviation of a station and returns their latitude and longitude from the MYSQL database
- 48.
49. header.php --> provides the main header for index.php, also contains the printed javascript from the php documents
- 50.
51. index.php --> main page
- 52.
53. markers.php --> holds the function that creates the markers based on the station string given from the route chosen.
- 54.
55. routefunctions.php --> creates the routes and sets them to the map (for homepage)
- 56.
57. routes.php --> creates the radio buttons to the left that are clicked on to show specific routes
- 58.
59. updaterroute.php --> If nothing is in the database, inserts XML data to the database. Else, updates the routes in the MYSQL database once a week after the last modified date of the php file
- 60.
61. updaterroute.php --> If nothing is in the database, inserts XML data to the database. Else, updates the stations in the MYSQL database once a week after the last modified date of the php file
- 62.
- 63.
64. I hope you enjoy, and thank you for grading all my code!
- 65.
- 66.

67. Crystal Hsieh  
68.  
69.

```
1. <?
2.
3. //function that takes abbreviation of the station, queries the XML from the BART API and returns real-time arrival times
4.
5. function getArrivals($abbrev){
6.
7.     //query XML file from BART
8.     $xml = new SimpleXMLElement(file_get_contents('http://www.bart.gov/dev/eta/bart_eta.xml'));
9.     $stations = $xml->xpath("station");
10.
11.     //because the arrival file gives ALL the stations,
12.     //I must use an "if statement" to find the correct station's times
13.
14.     foreach($stations as $station){
15.         $abbr = $station->abbr;
16.         if($abbr == $abbrev){
17.             $eta = $station->eta;
18.             $endstring = "";
19.             foreach($eta as $arrival){
20.                 $dest = $arrival->destination;
21.                 $est = $arrival->estimate;
22.                 $endstring = $endstring."#".$dest." ".$est;
23.             }
24.         }
25.     }
26.
27.     //I decided to take each arrival destination and time and separate them by "."
28.     //but still keeping them in their pairs with "#", so it could easily be separated with explode() later
29.
30.
31.     return $endstring;
32. }
33.
34. ?>
```

```
1. <? include("database.php");
2.
3. //after a radio button from the routes.php document has been clicked,
4. //this file will take the route number of that radio button and will print out the javascript
5. //for that particular route
6.
7.     $number = $_POST["route"];
8.
9. //if the value for $number exists,
10. if(isset($number)){
11.
12.     //use htmlspecialchars for added security because it came from a $_POST[] <- input from the user
13.     //prevention against injection attacks
14.     //as shown in https://github.com/codekiln/S75-Sections/blob/master/opencyclemap/map08.php
15.     $number = htmlspecialchars($number);
16.
17.     //clear the map of any remaining polylines or markers
18.     echo "clearPaths()";
19.     echo "clearMarkers()";
20.
21.     //create new polylines/markers
22.     $sqlselect = sprintf("SELECT * FROM routes WHERE number = '$number'");
23.     $select = mysql_query($sqlselect, $connect);
24.
25.     while($rows = mysql_fetch_array($select)){
26.         $stat = $rows['stations'];
27.         $pathname = "path".$number;
28.
29.         //sets it to the map (so it can be seen)
30.         echo $pathname.".setMap(map)";
31.
32.         //uses this method to set the stations for this particular route
33.         getmarkers($stat);
34.     }
35. }
36.
37.
38. ?>
```

why not call it  
\$routeId or  
something that  
describes it?

you should provide  
the user with some  
way to know which  
route is currently  
selected in the list  
after they have  
clicked on one

```
1. <?php
2.
3. //this is a function that takes the abbreviation of a station and queries the XML document provided by BART API
4. //so that it will return the latitude and longitude of that specific station.
5.
6. //it should only be used once, when it's updating the MYSQL database
7.
8. function get_coord($abbrev){
9.
10.     //creating url
11.     $url = "http://api.bart.gov/api/stn.aspx?cmd=stninfo&orig=".$abbrev."&key=MW9S-E7SL-26DU-VV8V";
12.
13.     //retrieving info from XML
14.     $xml = new SimpleXMLElement(file_get_contents($url));
15.     $lat = $xml->xpath("stations/station/gtfs_latitude");
16.     $lon = $xml->xpath("stations/station/gtfs_longitude");
17.
18.
19.     //I would have separated them by a period, but that would break up the decimals!
20.     //it will later be parsed with explode()
21.
22.     $latlon = $lat[0].'p'.$lon[0];
23.
24.     return $latlon;
25. }
26.
27. ?>
```

```
1. <?
2.
3.     //in order to save the effort of always starting the session and checking the connection to the database,
4.     //I have a separate php file to connect to the database.
5.
6.
7.     //start the session
8.     session_start();
9.
10.    //it checks if it can connect to the database server.
11.    //if not, return error message
12.    if(($connect = mysql_connect("192.168.56.50","jharvard","crimson")) == FALSE){
13.        die("Error! Unable to connect to the database!");
14.    }
15.
16.    // taken from login example from Malan, check if it can select database
17.    if(mysql_select_db("jharvard_bart", $connect) == FALSE){
18.        die("Error! Unable to select the database!");
19.    }
20. ?>
```

```
1. <?php
2.
3.
4. //takes in the route number and returns a string that holds all the stations that the route goes through
5. function get_stations($number){
6.     //creates the URL to access the BART API (using the public key) and getting the stations for the particular route #
7.
8.     $url = "http://api.bart.gov/api/route.aspx?cmd=routeinfo&route=".$number."&key=MW9S-E7SL-26DU-VV8V";
9.
10.    $xml = new SimpleXMLElement(file_get_contents($url));
11.    $stations = $xml->xpath("routes/route/config/station");
12.
13.    $statstring = $stations[0];
14.
15.    //taking the stations and separating them by a comma, to put in the MYSQL database
16.    //I put them in commas so that I can separate them later by explode()
17.
18.    for($i = 1; $i < count($stations); $i++){
19.        $statstring = $statstring.", ".$stations[$i];
20.    }
21.
22.    return $statstring;
23. }
24.
25. ?>
```

you need to specify  
that it is csv

this is not really the  
relational database  
way to solve this  
problem; you should  
have a stations  
table and a routes  
table, but your  
routes should have  
route number,  
station number, and  
ordinal number.  
Then you can pull  
out all the station  
ids and sort them by  
ordinal number.



```
1. <?
2.     //function made to parse the stations string that is made for a particular route
3.     //and query the MYSQL database for the latitudes and longitudes of all the stations
4.     //and then echo the creation of a latLng object.
5.
6.     //it was basically created to simplify the process of parsing the station string and inputting the javascript in
7.     //so that the path for the route can be created.
8.     function getlatlon($stat){
9.
10.         include("database.php");
11.
12.         $stations = explode(',', $stat);
13.
14.         foreach($stations as $station){
15.             $sqlselect = sprintf("SELECT `lat`, `lon` FROM `stations` WHERE abbr='$station'");
16.             $select = mysql_query($sqlselect, $connect);
17.             $array = mysql_fetch_array($select);
18.             $lat = $array['lat'];
19.             $lon = $array['lon'];
20.
21.             if(isset($lat) && isset($lon)){
22.                 echo "new google.maps.LatLng(".$lat.", ".$lon."),";
23.             }else{
24.                 echo "";
25.             }
26.         }
27.     }
28.
29.
30. ?>
```

```

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.
5. <head>
6.
7. <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
8.
9. <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
10.
11. <link href="css/mapstyle.css" rel="stylesheet" type="text/css" />
12.
13. <title>Bay Area Rapid Transit Map</title>
14.
15. <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
16. <script type="text/javascript">

```

try your javascript  
out at [jslint.com](http://jslint.com) to  
get some feedback  
on code style

you need `//<![CDATA[`  
here to avoid validation errors.

```

18. //initialize() will be performed onload
19.
20. function initialize() {
21.
22. //I found the center coordinate on google maps by finding a station close to the center (on the BART website),
23. //then looking up the address for that coordinate,
24. //then estimating the "center" of the routes,
25. //then I right-clicked the map and selected "What's Here?" to get the coordinate
26.
27. var myLatLng = new google.maps.LatLng(37.828226,-122.143478);
28.
29. //Used mostly from the google maps API samples page
30. var myOptions = {
31. zoom: 10,
32. center: myLatLng,
33. mapTypeId: google.maps.MapTypeId.ROADMAP
34. }
35.
36.
37. //These are the arrays that will hold the path objects and the marker objects
38.
39. var pathArray = [];
40. var markerArray = [];
41.
42. //I found the BART logo on google documents, and then I edited it with MSPaint
43. //I thought it would look more appealing then just the regular pointers.
44. //Also, I am aware that the BART API said that their logos should not be used,
45. //but since this project is most likely not going to be in public, regular use
46. //I thought I would add it just to add more style to the page.
47.
48. var image = 'image/bart_icon.gif';

```

```

// Consider omitting the repeated "Var" Keyword
var myLatLng = new google.maps.LatLng(37.828226, -122.143478),
// Used mostly from the google maps API samples page
myOptions = {
    zoom: 10,
    center: myLatLng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
},
// These will hold the path and marker objects
pathArray = [],
markerArray = [],
image = 'image/bart_icon.gif',
clearPaths = function() {
    // ... function body
},
clearMarkers = function() {
    // ... function body
},
map = new google.maps.Map(
    document.getElementById("map_canvas"), myOptions);
// ... or, decide to do something different
// http://uxebu.com/blog/2010/04/02/one-var-statement-for-one-
variable/

```

either way, realize that in javascript, all variables are hoisted to the top of the function they are declared in, without regard to order. Douglas Crockford suggests making variables alphabetical at the top of a function. A lot of people disagree, but then again, a lot of companies actually use his jslint to validate their code!

```

49.
50.    //referenced by http://code.google.com/apis/maps/documentation/javascript/overlays.html
51.    //to clear any polylines or markers currently on the map
52.    function clearPaths() {
53.        if (pathArray) {
54.            for (i in pathArray) {
55.                pathArray[i].setMap(null);
56.            }
57.        }
58.    }
59.
60.    function clearMarkers(){
61.        if (markerArray) {
62.            for (i in markerArray) {
63.                if(markerArray[i].getVisible()){
64.                    markerArray[i].setMap(null);
65.                }
66.            }
67.        }
68.    }
69.
70.    //added the map to the "map_canvas" div
71.    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
72.
73.
74.    <?php
75.    //adding all the different functions and php files that pertain to the markers and paths
76.    //more info within each php document
77.        include("getlatlon.php");
78.        include("coord_func.php");
79.        include("arrivals.php");
80.        include("markers.php");
81.        include("routefunctions.php");
82.        include("clicked.php");
83.    ?>
84.    }
85. </script>
86. </head>
87. <body onload="initialize()">
88.    <?php include("updatestation.php");
89.        include("updateroute.php");
90.
91.    //once initialized, the two php files are there to weekly update the station
92.    //more info about them inside the php files
93.    ?>
94.    <div id="map_canvas"></div>
95.    <div id="menu">
96.    <!--below is a banner that I created with only MSPaint. I used "Big Bimbo" font from dafont.com (free font site) -->

```

It is less than clear here that you are speaking about javascript functions

close the cdata with //]]>

thanks for the attribution

97. 

```
1. <!-- this is the actual page that will be shown to the user. It's basically the template for the php files to be put in one file -->
2.
3. <?php include("header.php"); ?>
4. <p>
5. All Available Routes
6. <br/>
7. Choose a route below to see it on the map!
8. <br/>
9. <!-- I could only make a div tag if I used object tags, according to the validator -->
10. <object>
11. <div id="routes">
12. <?php include("routes.php"); ?>
13. </div>
14. </object>
15. </p>
16. </div>
17. </body>
18. </html>
```

you DID validate! I wondered if you did; this is the only sign I have seen!

validator

id="menu" from header.php

Object tags are the wrong things to use. Object tags are used for embedding objects that require external libraries to render properly, such as flash and java. I know the error you are probably mentioning: 'document type does not allow element "div" here; missing one of "object", "ins", "del", "map", "button" start-tag.' Unfortunately, that validation message is a little vague. If you had used the html 5 doctype instead of the html 4 strict doctype, you would have gotten a better error message: "XHTML element div not allowed as child of XHTML element p in this context," which indicates the real problem: you may not have a div element that is the child of a p element.

```

1. <?
2. //this is a function so that when a route is chosen, the stations for that route show up too.
3. //it takes in the station string (all the stations for that route) and creates a marker for each station
4.
5. function getmarkers($stat){
6.
7.     //parse the station string
8.     $stations = explode(',', $stat);
9.
10.    //for each station abbreviation
11.    foreach($stations as $station){
12.
13.        include("database.php");
14.
15.        //select the row with that particular abbreviation
16.        $sqlselect = sprintf("SELECT SQL_CACHE * FROM stations WHERE abbr='$station'");
17.        $select = mysql_query($sqlselect, $connect);
18.
19.        while($row = mysql_fetch_assoc($select)){
20.            $abbr = $row['abbr'];
21.            $name = $row['name'];
22.            $lat = $row['lat'];
23.            $lon = $row['lon'];
24.            //each marker name needs to be unique,
25.            //so I used the station's unique abbreviation to differentiate them
26.            $marker = "station".$abbr;
27.
28.            //print the proper javascript to create the marker for this station
29.            $coordstring = "var a".$abbr." = new google.maps.LatLng(".$lat.", ".$lon.");";
30.
31.            echo $coordstring;
32.            echo "var ".$marker." = new google.maps.Marker({"
33.                echo "position: a".$abbr.",";
34.                echo "map: map,";
35.            echo "icon: image,";
36.
37.            $title = 'title:'.$name.'}';
38.
39.            echo $title;
40.
41.            //afterwards, put the finished marker in the markerArray
42.            echo "markerArray.push(".$marker.");";
43.
44.            //also add the info window for the marker
45.            echo "var info".$abbr." = new google.maps.InfoWindow({"
46.
47.            //using arrivals.php, it will access the real-time arrivals
48.            //from the BART API

```

you should clearly define what the \$stat variable should be here - what kind of string do you mean? a string can hold any character data

nowhere in this description do you explain that the result of running this function is outputting javascript; quite confusing

it's a best practice to use a space between separate parameters for enhanced visibility.

I prefer php functions that return a value to php functions that perform an action such as echoing items to the page, because then the caller can always decide to echo the result to the page OR to do something else with the compiled data. You could have \$str .= \$coordstring, etc. here, then return \$str at the end. That would give the caller the flexibility to put the output through a pretty-printer, for example.

this is not good design. If anyone ever wanted to change the variable your map is stored in, they would have to dig through / interpret your PHP code; not to mention that it makes it harder to debug

```

49.     $arrivals = getArrivals($abbr);
50.     $array = explode('#',$arrivals);
51.     $arrive = "";
52.
53.     //parsing each estimate
54.     foreach($array as $group){
55.         $split = explode('.', $group);
56.         $dest = $split[0];
57.         $time = $split[1];
58.         if(isset($dest) && isset($time)){
59.             $arrive = $arrive."<p> Destination: ".$dest."<br/> Estimated Time of Arrival of the Next Three Trains: <br/>".
$time."</p>";
60.         }
61.     }
62.
63.     echo "content: '<strong>Station: ".$name."<br/>Arrivals:</strong>".$arrive."'";
64.     echo "}}";
65.
66.     //add the listener to this specific marker
67.     //so when someone clicks on the marker, the info window will appear.
68.     echo "google.maps.event.addListener(".$marker.", 'click', function() {";
69.     echo "info".$abbr.".open(map, ".$marker.");";
70.     echo "}}";
71. }
72. //add the marker to the map
73. echo $marker.".setMap(map);";
74. }
75. }
76.
77. ?>

```

You are missing a fundamental point about javascript here. You do not need a separate variable name for each marker in order have them lay "in waiting" for a click. Google "javascript scope" or "javascript closures," or watch this: <http://developer.yahoo.com/yui/theater/video.php?v=crockonjs-3>

Here inside this function there is a separate scope that presents the opportunity to declare a variable that will store data private to that function. Even if there are thousands of markers, and each has an info window named "info," there will be no conflict - as long as you declare it within the function.

it's not really "real-time," because you are parsing the remote data only when the route is drawn. If the page was open for a long time and you clicked on a marker, the information would be wrong. You should be using ajax to fetch only what is needed here, not the data that already exists on the client.

```

1. -- MySQL dump 10.13  Distrib 5.5.14, for Linux (i686)
2. --
3. -- Host: localhost    Database: jharvard_bart
4. -- -----
5. -- Server version    5.5.14
6.
7. /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8. /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9. /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10. /*!40101 SET NAMES utf8 */;
11. /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12. /*!40103 SET TIME_ZONE='+00:00' */;
13. /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14. /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15. /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16. /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17.
18. --
19. -- Table structure for table `routes`
20. --
21.
22. DROP TABLE IF EXISTS `routes`;
23. /*!40101 SET @saved_cs_client      = @@character_set_client */;
24. /*!40101 SET character_set_client = utf8 */;
25. CREATE TABLE `routes` (
26.   `name` varchar(255) NOT NULL,
27.   `abbr` varchar(10) NOT NULL,
28.   `routeID` varchar(255) NOT NULL,
29.   `number` int(255) NOT NULL,
30.   `color` varchar(255) NOT NULL,
31.   `stations` varchar(255) NOT NULL,
32.   UNIQUE KEY `abbr` (`abbr`)
33. ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
34. /*!40101 SET character_set_client = @saved_cs_client */;
35.
36. --
37. -- Dumping data for table `routes`
38. --
39.
40. LOCK TABLES `routes` WRITE;
41. /*!40000 ALTER TABLE `routes` DISABLE KEYS */;
42. INSERT INTO `routes` VALUES ('Pittsburg/Bay Point - SFIA/Millbrae','PITT-SFIA','ROUTE 1',1,'#ffff33',
  'PITT,NCON,CONC,PHIL,WCRK,LAFY,ORIN,ROCK,MCAR,19TH,12TH,WOAK,EMBR,MONT,POWL,CIVC,16TH,24TH,GLEN,BALB,DALY,COLM,SSAN,SBRN,SFIA,MLBR'),('Daly
  City - Dublin/Pleasanton','DALY-DUBL','ROUTE 12',12,'#0099cc',
  'DALY,BALB,GLEN,24TH,16TH,CIVC,POWL,MONT,EMBR,WOAK,LAKE,FTVL,COLS,SANL,BAYF,CAST,WDUB,DUBL'),('Daly City - Fremont','DALY-FRMT','ROUTE 6',6,
  '#339933','DALY,BALB,GLEN,24TH,16TH,CIVC,POWL,MONT,EMBR,WOAK,LAKE,FTVL,COLS,SANL,BAYF,HAYW,SHAY,UCTY,FRMT'),('Dublin/Pleasanton - Daly City',
  'DUBL-DALY','ROUTE 11',11,'#0099cc','DUBL,WDUB,CAST,BAYF,SANL,COLS,FTVL,LAKE,WOAK,EMBR,MONT,POWL,CIVC,16TH,24TH,GLEN,BALB,DALY'),('Fremont -
  Daly City','FRMT-DALY','ROUTE 5',5,'#339933','FRMT,UCTY,SHAY,HAYW,BAYF,SANL,COLS,FTVL,LAKE,WOAK,EMBR,MONT,POWL,CIVC,16TH,24TH,GLEN,BALB,DALY'),

```



```

('Fremont - Richmond','FRMT-RICH','ROUTE 3',3,'#ff9933',
'FRMT,UCTY,SHAY,HAYW,BAYF,SANL,COLS,FTVL,LAKE,12TH,19TH,MCAR,ASHB,DBRK,NBRK,PLZA,DELN,RICH'),('Millbrae/Daly City - Richmond','MLBR-RICH',
'ROUTE 8',8,'#ff0000','MLBR,SBRN,SSAN,COLM,DALY,BALB,GLEN,24TH,16TH,CIVC,POWL,MONT,EMBR,WOAK,12TH,19TH,MCAR,ASHB,DBRK,NBRK,PLZA,DELN,RICH'),(
'Richmond - Fremont','RICH-FRMT','ROUTE 4',4,'#ff9933',
'RICH,DELN,PLZA,NBRK,DBRK,ASHB,MCAR,19TH,12TH,LAKE,FTVL,COLS,SANL,BAYF,HAYW,SHAY,UCTY,FRMT'),('Richmond - Daly City/Millbrae','RICH-MLBR',
'ROUTE 7',7,'#ff0000','RICH,DELN,PLZA,NBRK,DBRK,ASHB,MCAR,19TH,12TH,WOAK,EMBR,MONT,POWL,CIVC,16TH,24TH,GLEN,BALB,DALY,COLM,SSAN,SBRN,MLBR'),(
'Millbrae/SFIA - Pittsburg/Bay Point','SFIA-PITT','ROUTE 2',2,'#ffff33',
'MLBR,SFIA,SBRN,SSAN,COLM,DALY,BALB,GLEN,24TH,16TH,CIVC,POWL,MONT,EMBR,WOAK,12TH,19TH,MCAR,ROCK,ORIN,LAFY,WCRK,PHIL,CONC,NCON,PITT');
43. /*!40000 ALTER TABLE `routes` ENABLE KEYS */;
44. UNLOCK TABLES;
45.
46. --
47. -- Table structure for table `stations`
48. --
49.
50. DROP TABLE IF EXISTS `stations`;
51. /*!40101 SET @saved_cs_client      = @@character_set_client */;
52. /*!40101 SET character_set_client = utf8 */;
53. CREATE TABLE `stations` (
54.   `name` varchar(255) NOT NULL,
55.   `abbr` varchar(255) NOT NULL,
56.   `address` varchar(255) NOT NULL,
57.   `city` varchar(255) NOT NULL,
58.   `state` varchar(2) NOT NULL,
59.   `zipcode` int(32) NOT NULL,
60.   `lat` float NOT NULL,
61.   `lon` float NOT NULL,
62.   UNIQUE KEY `name` (`name`),
63.   UNIQUE KEY `abbr` (`abbr`)
64. ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
65. /*!40101 SET character_set_client = @saved_cs_client */;
66.
67. --
68. -- Dumping data for table `stations`
69. --
70.
71. LOCK TABLES `stations` WRITE;
72. /*!40000 ALTER TABLE `stations` DISABLE KEYS */;
73. INSERT INTO `stations` VALUES ('16th St. Mission','16TH','2000 Mission Street','San Francisco','CA',94110,37.7651,-122.42),('19th St. Oakland',
'19TH','1900 Broadway','Oakland','CA',94612,37.8079,-122.269),('24th St. Mission','24TH','2800 Mission Street','San Francisco','CA',94110,
37.7523,-122.418),('Ashby','ASHB','3100 Adeline Street','Berkeley','CA',94703,37.853,-122.27),('Balboa Park','BALB','401 Geneva Avenue','San
Francisco','CA',94112,37.722,-122.447),('Bay Fair','BAYF','15242 Hesperian Blvd.','San Leandro','CA',94578,37.6972,-122.127),('Castro Valley',
'CAST','3301 Norbridge Dr.','Castro Valley','CA',94546,37.6908,-122.076),('Civic Center/UN Plaza','CIVC','1150 Market Street','San Francisco',
'CA',94102,37.7795,-122.414),('Coliseum/Oakland Airport','COLS','7200 San Leandro St.','Oakland','CA',94621,37.754,-122.197),('Colma','COLM',
'365 D Street','Colma','CA',94014,37.6846,-122.466),('Concord','CONC','1451 Oakland Avenue','Concord','CA',94520,37.9737,-122.029),('Daly
City','DALY','500 John Daly Blvd.','Daly City','CA',94014,37.7061,-122.469),('Downtown Berkeley','DBRK','2160 Shattuck Avenue','Berkeley','CA',
94704,37.8699,-122.268),('Dublin/Pleasanton','DUBL','5801 Owens Dr.','Pleasanton','CA',94588,37.7017,-121.9),('El Cerrito del Norte','DELN',
'6400 Cutting Blvd.','El Cerrito','CA',94530,37.9257,-122.317),('El Cerrito Plaza','PLZA','6699 Fairmount Avenue','El Cerrito','CA',94530,

```

```

37.9031,-122.299),('Embarcadero','EMBR','298 Market Street','San Francisco','CA',94111,37.793,-122.397),('Fremont','FRMT','2000 BART Way',
'Fremont','CA',94536,37.5574,-121.976),('Fruitvale','FTVL','3401 East 12th Street','Oakland','CA',94601,37.775,-122.224),('Glen Park','GLEN',
'2901 Diamond Street','San Francisco','CA',94131,37.7329,-122.434),('Lafayette','LAFY','3601 Deer Hill Road','Lafayette','CA',94549,37.8934,-
122.124),('Lake Merritt','LAKE','800 Madison Street','Oakland','CA',94607,37.7975,-122.266),('MacArthur','MCAR','555 40th Street','Oakland',
'CA',94609,37.8284,-122.267),('Millbrae','MLBR','200 North Rollins Road','Millbrae','CA',94030,37.5998,-122.387),('Montgomery St.','MONT','598
Market Street','San Francisco','CA',94104,37.7893,-122.401),('North Berkeley','NBRK','1750 Sacramento Street','Berkeley','CA',94702,37.874,-
122.283),('North Concord/Martinez','NCON','3700 Port Chicago Highway','Concord','CA',94520,38.0033,-122.025),('Orinda','ORIN','11 Camino
Pablo','Orinda','CA',94563,37.8784,-122.184),('Pittsburg/Bay Point','PITT','1700 West Leland Road','Pittsburg','CA',94565,38.0189,-121.945),('
Pleasant Hill/Contra Costa Centre','PHIL','1365 Treat Blvd.','Walnut Creek','CA',94597,37.9284,-122.056),('Powell St.','POWL','899 Market
Street','San Francisco','CA',94102,37.785,-122.407),('Richmond','RICH','1700 Nevin Avenue','Richmond','CA',94801,37.9369,-122.353),('
Rockridge','ROCK','5660 College Avenue','Oakland','CA',94618,37.8446,-122.252),('San Bruno','SBRN','1151 Huntington Avenue','San Bruno','CA',
94066,37.6378,-122.416),('San Leandro','SANL','1401 San Leandro Blvd.','San Leandro','CA',94577,37.7226,-122.161),('South Hayward','SHAY',
'28601 Dixon Street','Hayward','CA',94544,37.6348,-122.058),('South San Francisco','SSAN','1333 Mission Road','South San Francisco','CA',94080,
37.6642,-122.444),('Union City','UCTY','10 Union Square','Union City','CA',94587,37.5912,-122.018),('Walnut Creek','WCRK','200 Ygnacio Valley
Road','Walnut Creek','CA',94596,37.9056,-122.067),('West Dublin/Pleasanton','WDUB','6501 Golden Gate Drive','Dublin','CA',94568,37.6998,-
121.928),('West Oakland','WOAK','1451 7th Street','Oakland','CA',94607,37.8047,-122.295),('12th St. Oakland City Center','12TH','1245
Broadway','Oakland','CA',94612,37.8037,-122.272);
74. /*!40000 ALTER TABLE `stations` ENABLE KEYS */;
75. UNLOCK TABLES;
76. /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
77.
78. /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
79. /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
80. /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
81. /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
82. /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
83. /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
84. /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
85.
86. -- Dump completed on 2011-08-09 2:55:35

```

```

1. <?
2. //this php file helps create the actual polylines that will be displayed.
3. //it queries the MYSQL database and prints the javascript for each route
4.
5.
6. include("database.php");
7.
8. //selects ALL routes
9. $sqlselect = sprintf("SELECT * FROM routes");
10. $select = mysql_query($sqlselect, $connect);
11.
12. while($rows = mysql_fetch_array($select)){
13.
14.     $num = $rows['number'];
15.     $stat = $rows['stations'];
16.     $color = $rows['color'];
17.
18.     //in order to distinguish between each particular route's method/variable names,
19.     //while keeping them all fairly organized and easy to maintain,
20.     //I added their function to the beginning of the name,
21.     //and the route's number to the end.
22.
23.     //I also did it because the variable names cannot start with a number.
24.
25.     $methodname = "poly".$num;
26.     $varname = "coord".$num;
27.     $pathname = "path".$num;
28.
29.     echo "var ".$varname." = [";
30.
31.     getlatlon($stat);
32.
33.     echo "];";
34.     echo "var ".$pathname." = new google.maps.Polyline({";
35.     echo "path: ".$varname.", ";
36.     echo "strokeColor: '".$color."', ";
37.     echo "strokeOpacity: 1.0, ";
38.     echo "strokeWeight: 4";
39.     echo "});";
40.
41.     //add the path to the pathArray so they are all accessible by one source
42.     echo "pathArray.push('".$pathname."');";
43.
44.     //place the path on the map
45.     echo $pathname.".setMap(map);";
46. }
47.
48.

```

no need for the  
sprintf here, it does  
not do anything if  
you do not have  
more parameters

```
49.
```

```
50. ?>
```

```

1. <?
2. //this is basically where all the routes are queried from the MYSQL database,
3. //and then they are printed on the left of the index.php document as buttons for the user
4. //when a radio button is clicked, it sends the information back to itself, telling clicked.php
5. //which route number was chosen so that it can display the correct route.
6.
7.
8. include("database.php");
9.
10. //selects all routes from the table
11. $sqlselect = sprintf("SELECT SQL_CACHE * FROM routes");
12. $select = mysql_query($sqlselect, $connect);
13.
14. //after getting feedback on project1, I'm sanitizing the PHP_SELF variable in clicked.php !
15. //I'm using htmlspecialchars(), as specified in:
16. //https://github.com/codekiln/S75-Sections/blob/master/opencyclemap/map08.php
17. ?>
18. <form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
19. <?
20. while($row = mysql_fetch_assoc($select)){
21.
22. //for each route in the database, it displays the name, abbreviation, and route id to the user.
23. //in the <input> tag, however, php is used to display the value of the route (the route's specific number)
24.
25. $name = $row['name'];
26. $abbr = $row['abbr'];
27. $routeid = $row['routeID'];
28. $num = $row['number'];
29. ?>
30. <p>
31. <label for="<? echo $abbr; ?>"><? echo $name; ?></label>
32.
33. <!-- I also used javascript with "this.form.submit();" so that as SOON as the user clicks the radio button, it will create that
particular route. -->
34. <input type="radio" value="<? echo $num; ?>" id= "<? echo $abbr; ?>" name="route" onclick="this.form.submit();"/>
35. <br/>
36. <? echo $abbr; ?>
37. <br/>
38. <? echo $routeid; ?>
39. </p>
40. <?
41. }
42. ?>
43. </form>

```

`$_SERVER["PHP_SELF"]` actually needs to be scrubbed with `htmlspecialchars` or `htmlentities` here.

```

1. <?
2.     // this page was created to either insert routes if there is nothing in the MYSQL database table,
3.     // or update the table once a week.
4.     // it is very similar to updatestation.php .
5.
6.     include("get_stations.php");
7.
8.     include("database.php");
9.
10.
11.     //query for stuff in table
12.     $sqlselect = sprintf("SELECT * FROM routes");
13.     $select = mysql_query($sqlselect, $connect);
14.     $array = mysql_fetch_array($select);
15.
16.     //check if anything is there first!
17.     //if there isn't anything there, then fill the database with information from BART API !
18.     if(!$array){
19.
20.         $xml = new SimpleXMLElement(file_get_contents('http://api.bart.gov/api/route.aspx?cmd=routes&key=MW9S-E7SL-26DU-VV8V'));
21.         $routes = $xml->xpath("routes/route");
22.
23.         //if nothing, then take each route and make a row
24.         foreach($routes as $route){
25.             //storing all the variables from each route
26.             $name = $route->name;
27.             $abbr = $route->abbr;
28.             $routeID = $route->routeID;
29.             $number = $route->number;
30.             $color = $route->color;
31.
32.             //function created to place all the stations into one string
33.             $stat = get_stations($number);
34.
35.             $insert = sprintf("INSERT INTO `routes`(`name`, `abbr`, `routeID`, `number`, `color`, `stations`) VALUES ('$name','$abbr','
$routeID','$number','$color','$stat')");
36.             $query = mysql_query($insert,$connect);
37.         }
38.         echo "Routes inserted!";
39.     }else{
40.         //get the last modified time
41.         $lmodified = getlastmod();
42.
43.         //subtract the unix time of last modified from the current time
44.         $time = time() - $lmodified;
45.
46.         //after each week past the last modified time, it will update itself.
47.         $nowtime = $time % 604800;

```

```
48.
49.     //after every 604800 seconds, it will update
50.     if($nowtime == 0){
51.         $xml = new SimpleXMLElement(file_get_contents('http://api.bart.gov/api/route.aspx?cmd=routes&key=MW9S-E7SL-26DU-VV8V'));
52.         $routes = $xml->xpath("routes/route");
53.
54.         //if nothing, then take each route and make a row
55.         foreach($routes as $route){
56.             //storing all the variables from each route
57.             $name = $route->name;
58.             $abbr = $route->abbr;
59.             $routeID = $route->routeID;
60.             $number = $route->number;
61.             $color = $route->color;
62.
63.             //function created to place all the stations into one string
64.             $stat = get_stations($number);
65.
66.             $update = sprintf("UPDATE `routes` SET `name`='%s', `abbr`='%s', `routeID`='%s', `number`='%s', `color`='%s'
        , `stations`='%s' WHERE `abbr`='%s'", $name, $abbr, $routeID, $number, $color, $stat);
67.             $query = mysql_query($update,$connect);
68.
69.         }
70.         echo "Routes are updated!";
71.     }
72. }
```

```

1. <?
2.
3. // this page was created to either insert stations if there is nothing in the MYSQL database table,
4. // or update the table once a week.
5. // it is very similar to updatroute.php .
6.
7. include("database.php");
8.
9.
10. //query for stuff in table
11. $sqlselect = sprintf("SELECT * FROM stations");
12. $select = mysql_query($sqlselect, $connect);
13. $array = mysql_fetch_array($select);
14.
15. //check if anything is there first!
16. //if there isn't anything there, then fill the database with information from BART API !
17. if(!$array){
18.
19.     $xml = new SimpleXMLElement(file_get_contents('http://api.bart.gov/api/stn.aspx?cmd=stns&key=MW9S-E7SL-26DU-VV8V'));
20.     $stations = $xml->xpath("stations/station");
21.
22.     //if nothing, then take each station and make a row
23.     foreach($stations as $station){
24.         //storing all the variables from each station
25.         $name = $station->name;
26.         $abbr = $station->abbr;
27.         $address = $station->address;
28.         $city = $station->city;
29.         //didn't add the county, I didn't think the information was necessary
30.         $state = $station->state;
31.         $zipcode = $station->zipcode;
32.
33.         $latlon = get_coord($abbr);
34.
35.         $array = explode('p', $latlon);
36.
37.         $lat = $array[0];
38.         $lon = $array[1];
39.
40.         $insert = sprintf("INSERT INTO `stations`(`name`,`abbr`, `address`, `city`, `state`, `zipcode`, `lat`, `lon`) VALUES ('$name', '$abbr', '$address', '$city', '$state', '$zipcode', '$lat', '$lon')");
41.         $query = mysql_query($insert,$connect);
42.     }
43.     echo "Stations inserted!";
44. }else{
45.     //get the last modified time
46.     $lmodified = getlastmod();
47.

```



```

48.      //subtract the unix time of last modified from the current time
49.      $time = time() - $lmodified;
50.
51.      //after each week past the last modified time, it will update itself.
52.      $nowtime = $time % 604800;
53.
54.      //after every 604800 seconds, or 7 days, it will execute an update.
55.
56.      if($nowtime == 0){
57.          $xml = new SimpleXMLElement(file_get_contents('http://api.bart.gov/api/stn.aspx?cmd=stns&key=MW9S-E7SL-26DU-VV8V'));
58.          $stations = $xml->xpath("stations/station");
59.
60.
61.          //update each station's information
62.          foreach($stations as $station){
63.              //storing all the variables from each station
64.              $name = $station->name;
65.              $abbr = $station->abbr;
66.              $address = $station->address;
67.              $city = $station->city;
68.              //didn't add the county, I didn't think the information was necessary
69.              $state = $station->state;
70.              $zipcode = $station->zipcode;
71.
72.              $latlon = get_coord($abbr);
73.
74.              $array = explode('p', $latlon);
75.
76.              $lat = $array[0];
77.              $lon = $array[1];
78.              //Update the info fetched from the xml document
79.
80.              $update = sprintf("UPDATE stations SET abbr='$abbr',address='$address',city='$city',state='$state',zipcode='$zipcode',lat='
$lat',lon='$lon' WHERE name='$name'");
81.              $query = mysql_query($update,$connect);
82.
83.          }
84.          echo "Stations are updated!";
85.      }
86.  }
87.
88.  ?>

```

```
1. html, body {
2.     height: 100%;
3.     margin: 0;
4.     padding: 0;
5.     font-family: Verdana, Arial, Helvetica, sans-serif;
6.     background-color: black;
7.     font-size: 95%;
8. }
9.
10. #routes {
11.     overflow: auto;
12.     height: 250px;
13.     width: 27%;
14.     margin: 20px;
15. }
16.
17. a: hover {
18.     color: yellow;
19. }
20.
21. #menu {
22.     padding: 15px;
23.     color: white;
24. }
25.
26. #map_canvas {
27.     float: right;
28.     height: 100%;
29.     width: 70%;
30. }
31.
32. @media print {
33.     html, body {
34.         height: auto;
35.     }
36. }
37.
```