pset8: CS50 Shuttle

Tommy MacWilliam

tmacwilliam@cs50.net

November 6, 2011

Today's Music

- Tommy's Choice
  - Never Let Me Down (Kanye West)
  - Bulletproof (La Roux)
  - E.T. (Katy Perry ft. Kanye West)
  - Follow me Down (3OH!3)
  - Comedy Tragedy History (Akala)

Today

- JavaScript <333333333333333333333333333333
- Distro code
- `populate()`
- `pickup()`
- `dropoff()`

JavaScript

- JavaScript is the best programming language ever
- other people will try to tell you otherwise
    - they are wrong

JavaScript

- JavaScript is NOT Java
    - Java is not the best programming language ever
- marketing ploy by Sun and Oracle
    - the “hot new web-programming language”

JavaScript

- PHP: server-side
    - runs on server, produces output, browser downloads
- JavaScript: client-side
    - browser downloads, runs code

JavaScript

- syntax (also) very similar to C and PHP
  - `if`, `else`, `for`, `while`, etc.
  - strings are built in (just like PHP)
  - variables don't need dollar signs (yay!)
- no types for variables or functions
  - `x = 5;`
  - `function increment(x) { return ++x; }`

JavaScript

- insert into page using `<script>`
- `<script>alert("oh hi, mark!")</script>`
- `<script src="file.js"></script>`

JavaScript

- example time!
  - `simple.html`

Arrays

- lists created with the `Array` function

```
var a = Array(5);
a[0] = 5;
a[1] = "tommy";
```

Arrays

- can also be created with `[]`
    - JavaScript arrays are dynamically-sized!

```
var b = [];
b[3] = 3.14159;
var c = [1, 2, 3, 4];
```

Associative Arrays

- just like PHP, JavaScript also has associative arrays (hashtables) built-in

```
var a = {};
hash["key"] = "value";
var tf = { name: "tommy", coolness: 100 };
```

Iterating

- `for in` can iterate over both array and associative array

```
var array = [1, 2, 3];
for (var i in array)
    alert(array[i]);
```

Objects

- associative arrays and objects (remember `Stock`?) are the same

```
var tf = { name: "tommy", coolness: 100 };
tf["name"] == "tommy"
tf.coolness == 100
```

Objects

- members can also be functions

```
var tf = { name: "tommy", grade: function() { retu
tf.grade();
```

Arrays

- example time!
  - `arrays.html`

Scope

- in C, loops, conditions, and functions limit the scope of variables
- in JavaScript, only functions limit the scope of variables

Scope

- `var` creates a local variable
  - where local means to the current function only
- without `var`, variable is global

Scope

- example time!
  - `scope.html`

Distro Code

- `buildings.js`, `houses.js`: arrays containing objects representing buildings
- `passengers.js`: array containing objects representing staff
- `math3d.js`: math, I'm no good at math
- `shuttle.js`: state of shuttle
- `service.js`: functions for implementing shuttle service
- `index.html`: brings it all together

Distro Code

- example time!

populate

- goal: remember location of each passenger
    - we need to know where they are to pick them up!

TODO

1. add passenger to passengers array

Arrays, again

- need to keep track of each passenger's `name`, `house`, `placemark`, `marker`, etc.
- there are `PASSENGERS.length` total passengers in the world
    - sounds like a good size for an array

Objects, again

- when using `for (var i in array)`, `i` is the current index in the array
- creating objects in JavaScript is easy!

```
var o = { key: "value" };
var p = { something: o };
```

TODO

1. ~~add passenger to passengers array~~

pickup

- goal: add passengers to the shuttle
  - also need to remove them from the world
  - also need to remove them from the map

TODO

1. detect if in range
2. add passenger to shuttle (if possible)
3. remove placemark
4. remove marker

Distance

- `shuttle.distance(lat, lng)` calculates distance from current position of shuttle to an arbitrary point
  - `lat`: latitude of point to get distance to
  - `lng`: longitude of point to get distance to

Finding Passengers

- we already know where passengers are from `populate()`
  - since we remembered a passenger's `placemark`, `marker`, etc.
- need to check if we are near any passenger
  - loop through all passengers and calculate `shuttle.distance`

Locating Passengers

- location information given by `placemark` associated with passenger
  - from that `placemark`, need to `getGeometry()` associated with it
  - from geometry, you can `getLatitude()` and `getLongitude()`

TODO

1. ~~detect if in range~~
2. add passenger to shuttle (if possible)
3. remove placemark
4. remove marker

Adding Passengers

- can only add passengers if `shuttle.seats` has room
  - has a fixed number of seats, given by `shuttle.seats.length`
- iterate through `seats` and look for an empty seat!
  - store passenger in the seat
- in range of multiple passengers? add as many as possible!

Displaying Passengers

- `chart()` displays position of passengers in shuttle
- iterates over each seat, but shouldn't say `TODO`!

TODO

1. ~~detect if in range~~
2. ~~add passenger to shuttle (if possible)~~
3. remove placemark
4. remove marker

Placemark

- remember, placemark is photo on the 3D world
- to get everything in world, need to `var features = earth.getFeatures()`
- once you have `features`, you can `features.removeChild(p)`
  - where `p` is a placemark on the world

TODO

1. ~~detect if in range~~
2. ~~add passenger to shuttle (if possible)~~
3. ~~remove placemark~~
4. remove marker

Marker

- remember, marker is an icon on the 2D map
- can remove a marker `m` with `m.setMap(null);`

TODO

1. ~~detect if in range~~
2. ~~add passenger to shuttle (if possible)~~
3. ~~remove placemark~~
4. ~~remove marker~~

dropoff

- goal: remove passengers from shuttle
    - only if in range of destination

TODO

1. check if in range of any houses
2. drop off all passengers going to current location

Houses

- don't forget about `HOUSES` array!
  - gives latitude and longitude of each house
- good thing we remembered which `house` each passenger was going to!

TODO

1. ~~check if in range of any houses~~
2. drop off all passengers going to current location

Dropping off

- can use `shuttle.distance()` again to calculate distance
- to remove passenger, just set position in array to `null`
  - can technically resize the array dynamically, but that's harder!
  - `chart()` assumes a fixed number of seats
- make sure to check all passengers in shuttle

TODO

1. ~~check if in range of any houses~~
2. ~~drop off all passengers going to current location~~

Extra Features

- points
- timer
- group passengers by house
- flying
- teleportation
- change speed
- fuel
- make your own!

Thanks!

These were walkthroughs.