***Mobile Web Development***

***Module Code: IT3032FP***

***Project Guide***

**Table of Content**

(A) Preparation 1

(B) Limit count to max 10 4

(C) Make the first client draggable 5

(D) Placement of cars 7

(E) Placement of exit and cashier 12

(F) Make the client droppable to the correct brand of car – Porsche only 13

(G) Additional processing after dropping – Porsche only 15

(H) Similar processing for other brands of car 18

(I) Code reuse using function 21

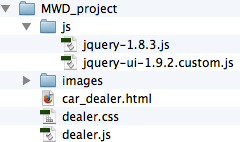
(J) Drop client to Exit 23

(K) Drop client to cashier 29

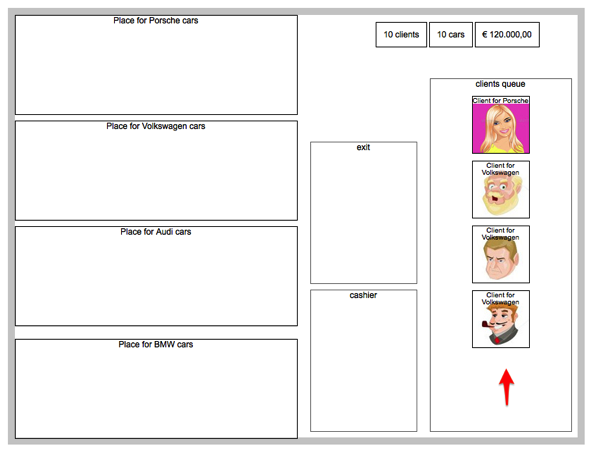
(L) Update the numbers 33

## (A) Preparation

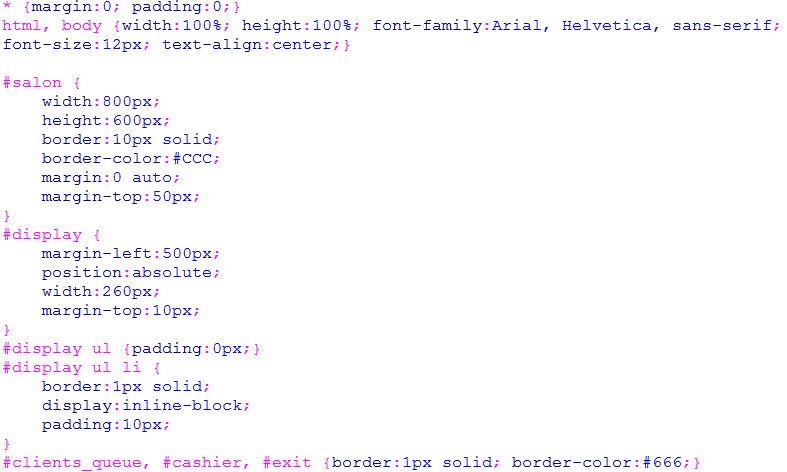
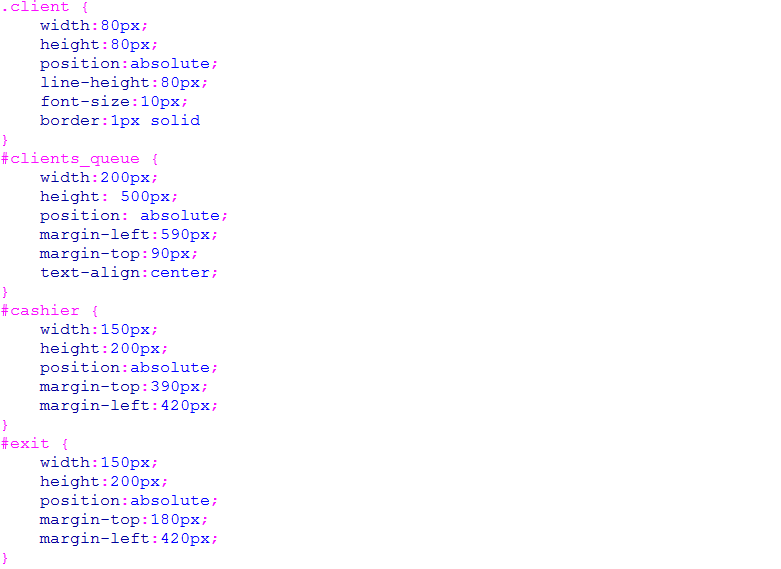
Step Unzip and copy the asset file to htdoc.

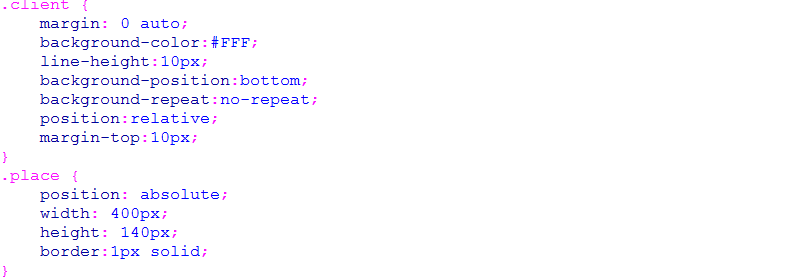


Step View the given car\_dealer.html in FireFox browser.

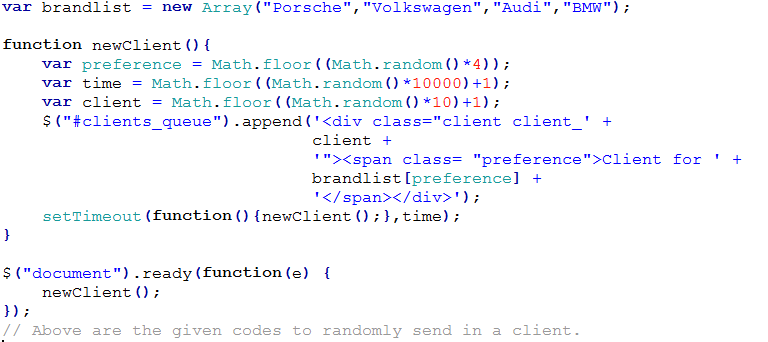


Step Observe the code in dealer.css.

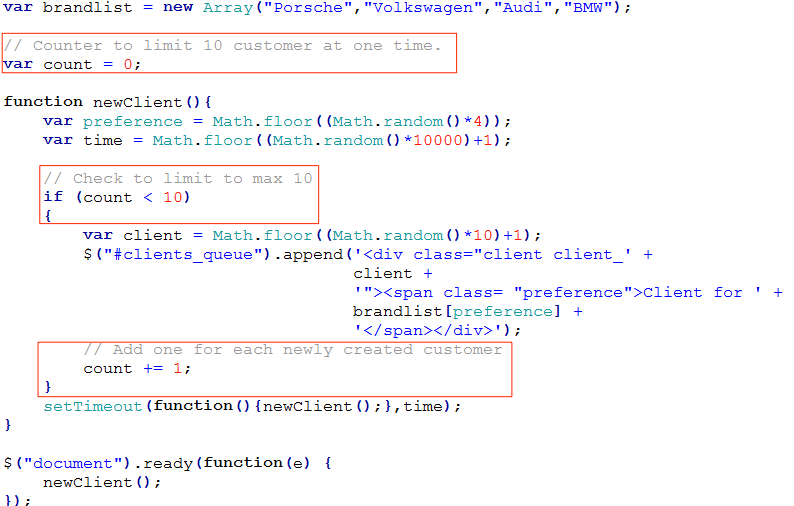
 

Step Observe the code in dealer.js.

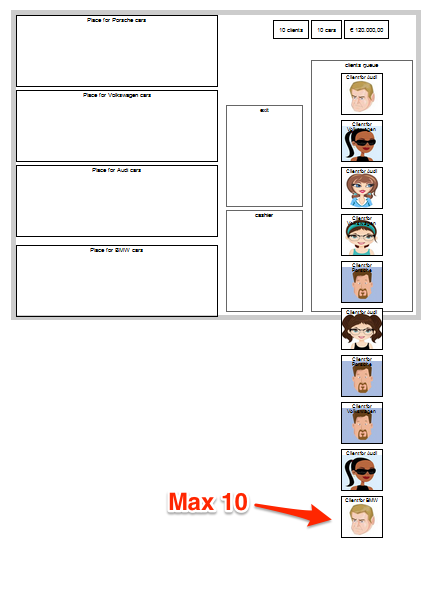


## (B) Limit count to max 10

Step Modify dealer.js to monitor the number of client created and stop adding new customer after max 10 clients are added.

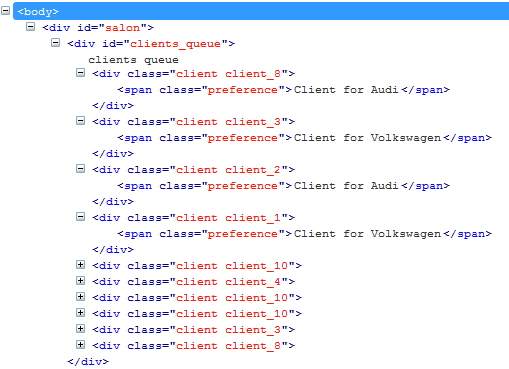


Step Test to confirm.

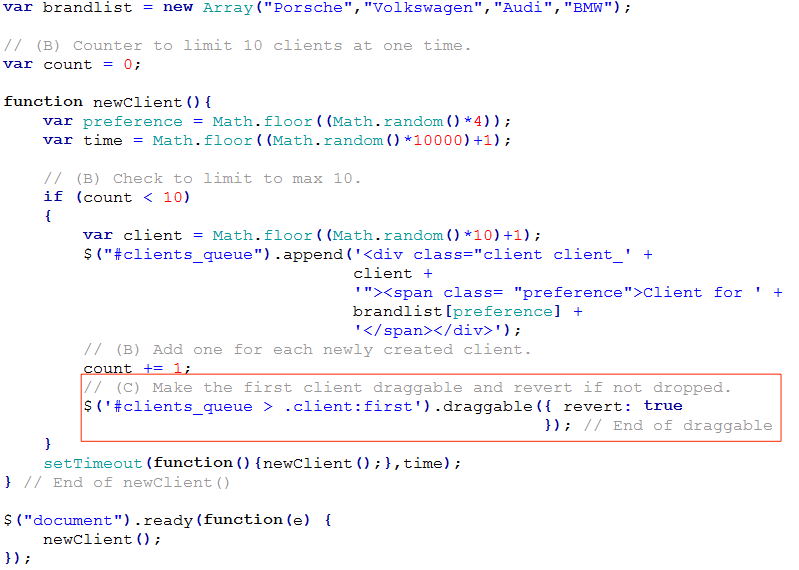


## (C) Make the first client draggable

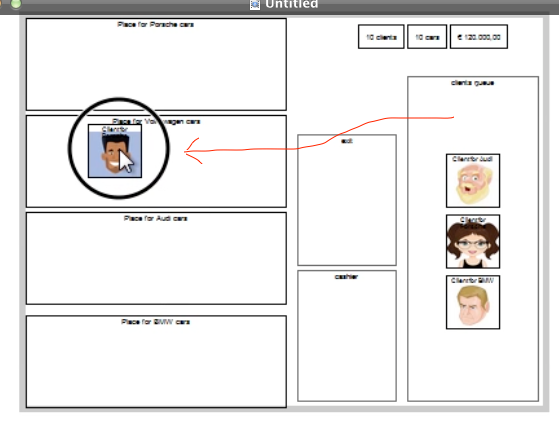
Step Observe the HTML code generated for the client queue, using Firebug.



Step Add the code to select the first client and make it draggable.



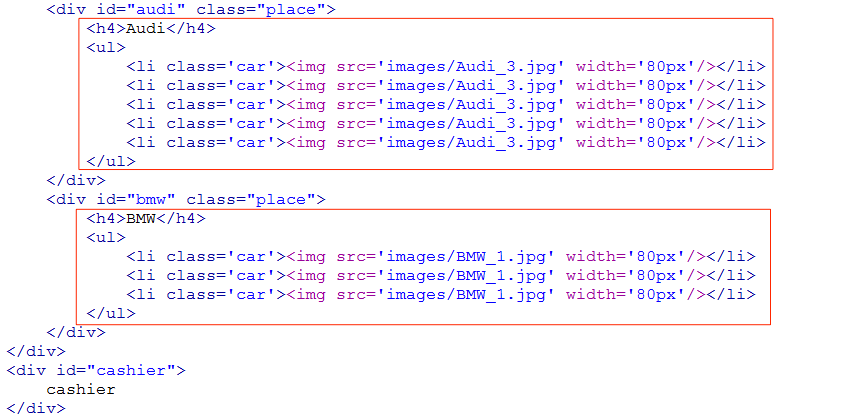
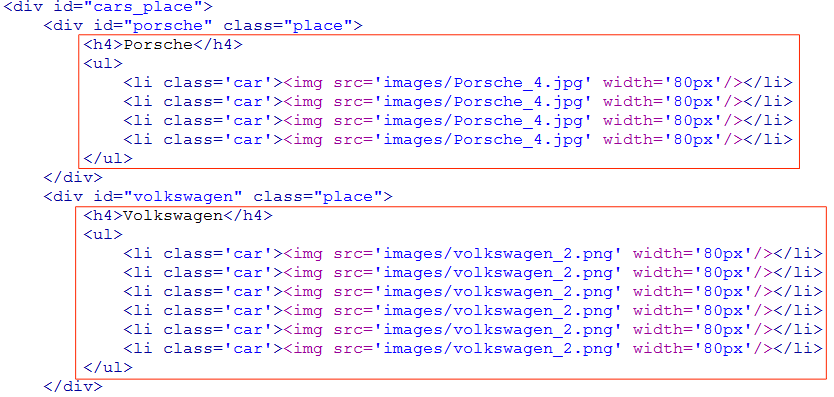
Step Test code. Only the first client is draggable and it will revert back if we drop it anywhere outside for now.



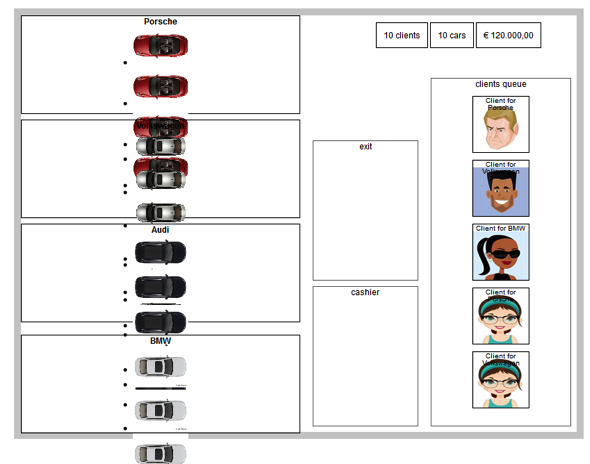
## (D) Placement of cars

Step There is no requirement to generate the car into the interface via code, then the easiest way to do is via HTML and CSS. In car\_dealer.html, add title and cars.

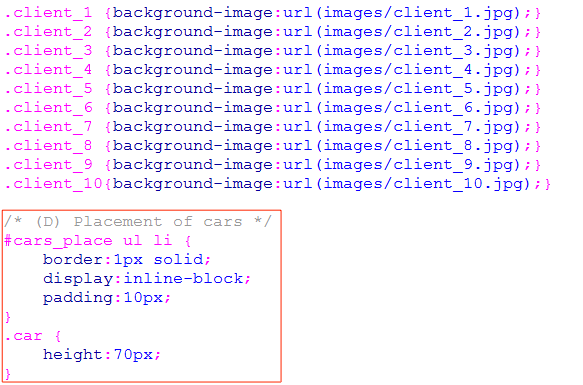
Note that the width of image is set to ‘80px’ to be the same as the width of the client image.



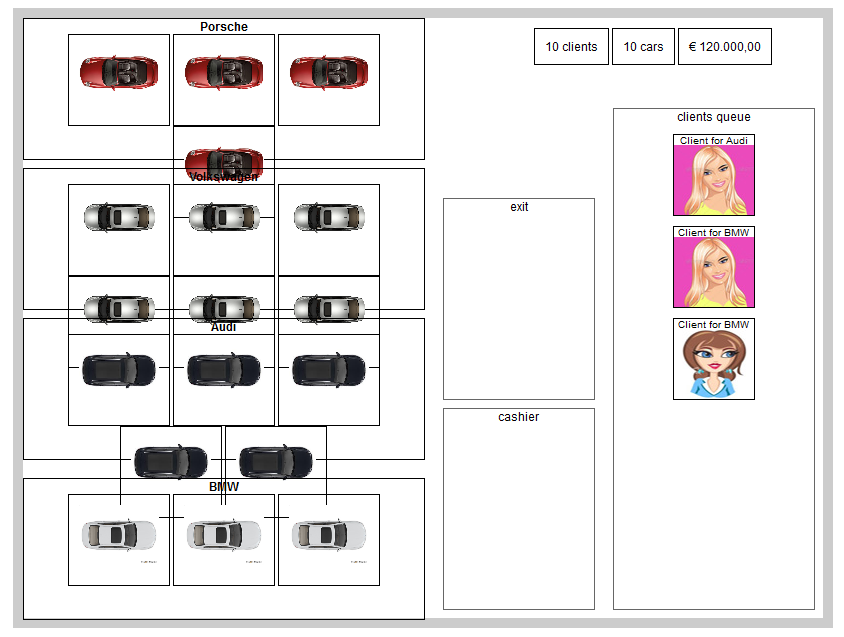
Step Test the code.



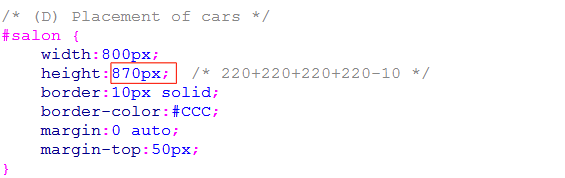
Step Modify dealer.css. The height for .car is set to 70px to be same as the client’s image height.



Step Test the code.



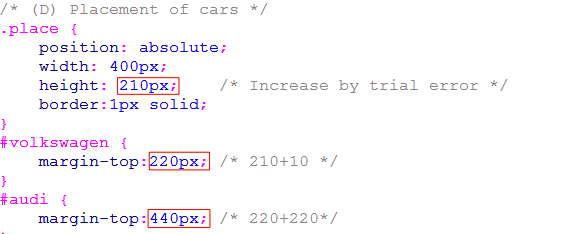
Step Modify dealer.css.



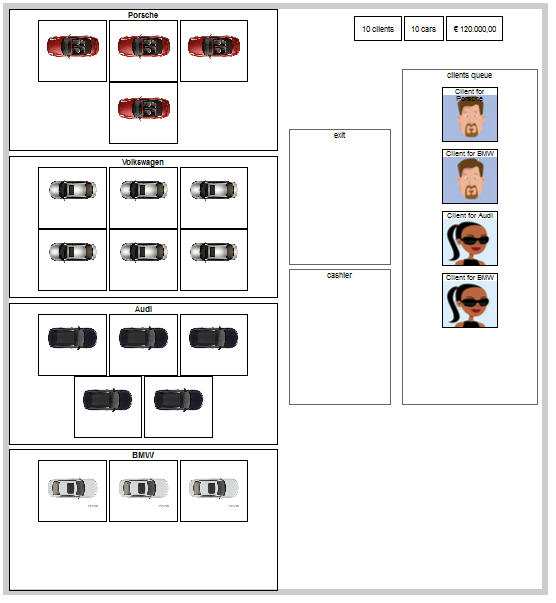
:

:

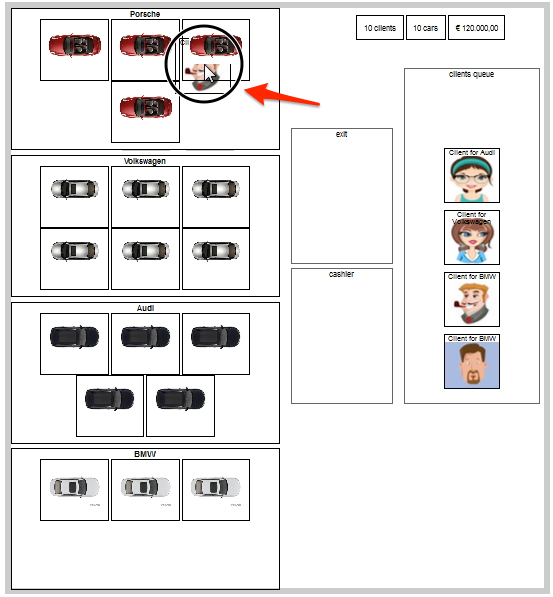
:



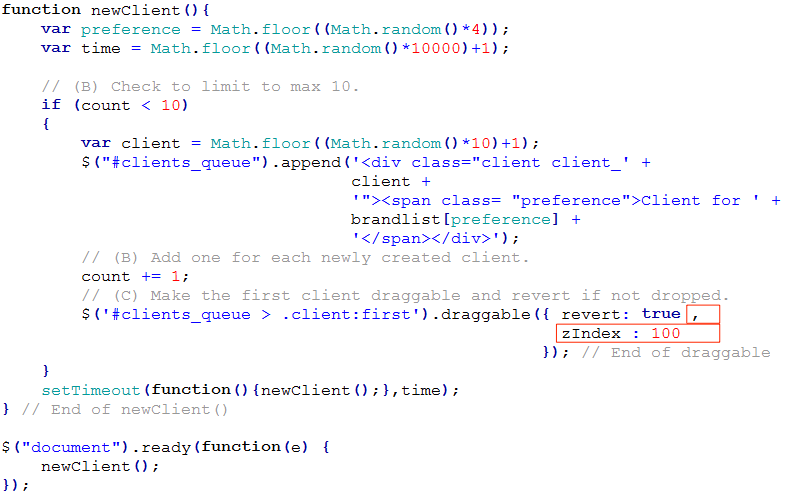
Step Test the code.



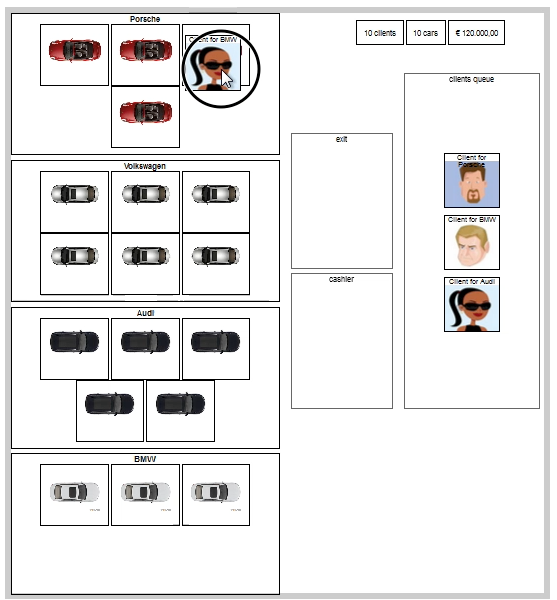
Step However, you will notice that for the draggable first client, the image goes below the cars.



Step Modify dealer.js to change the z index property (zIndex).

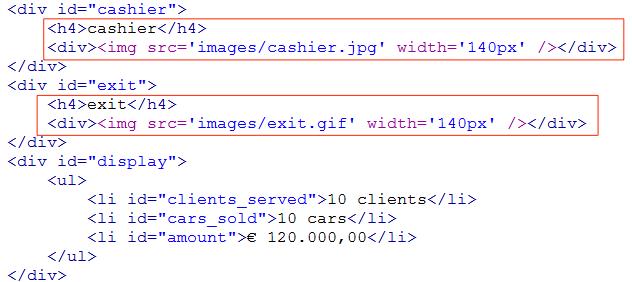


Step Test code.

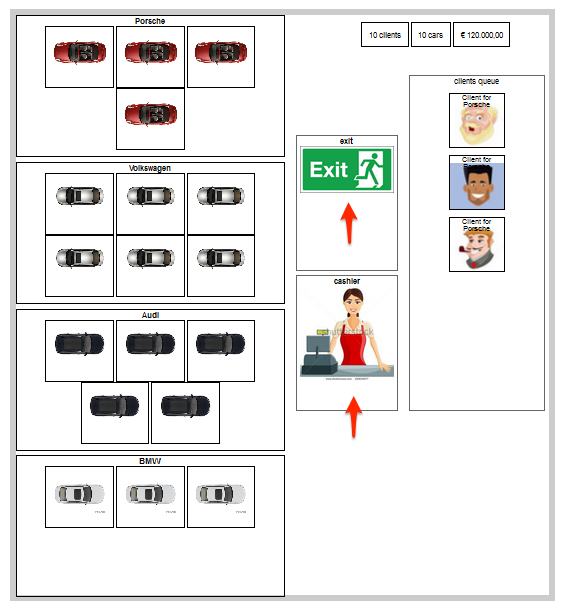


## (E) Placement of exit and cashier

Step Modify car\_dealer.html to add the images of cashier and exit.

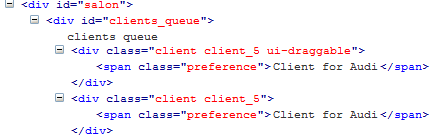


Step



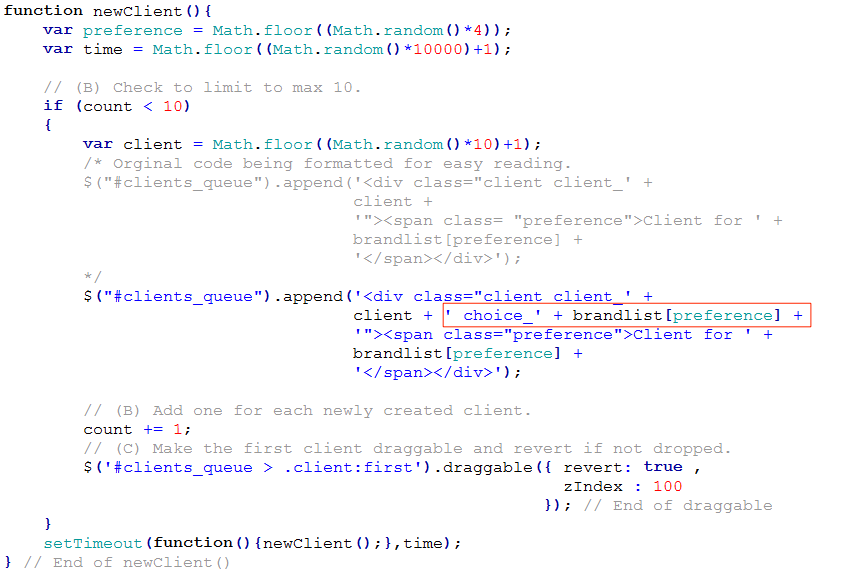
## (F) Make the client droppable to the correct brand of car – Porsche only

Step Observe via Firebug the code generated for the clients.

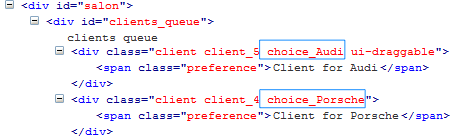


Step It is possible to find the choice of the client by looking for the text in the preference, but why not we just generate the choice in the class itself for easy checking.

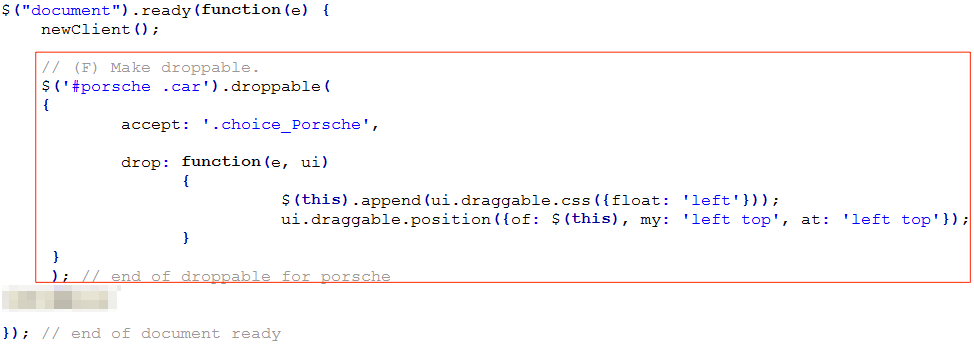
Take note of the blank space at the beginning of the string ‘ choice\_’ and the underscore at the end.



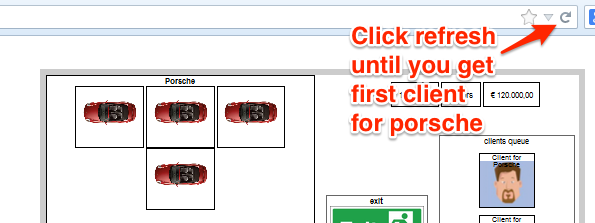
Step Test the code and observe via Firebug.

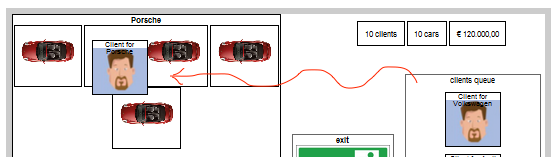


Step In dealer.js, add processing to use droppable with two parameters (accept and drop).



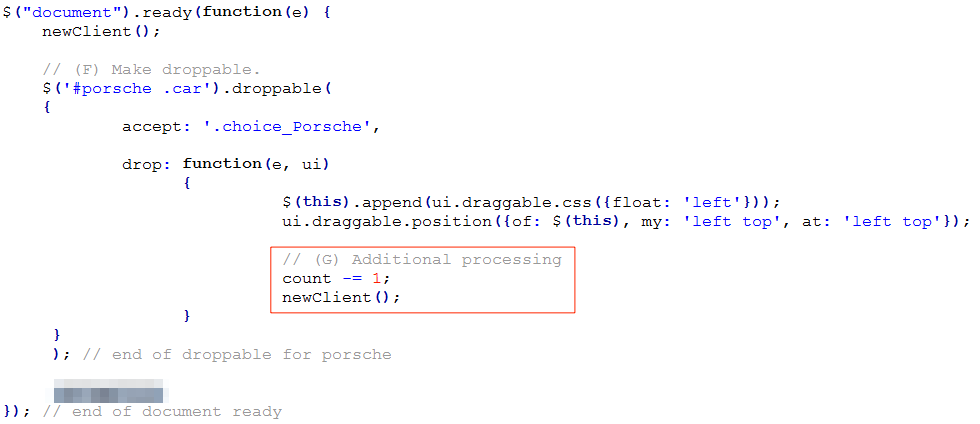
Step Test the code.



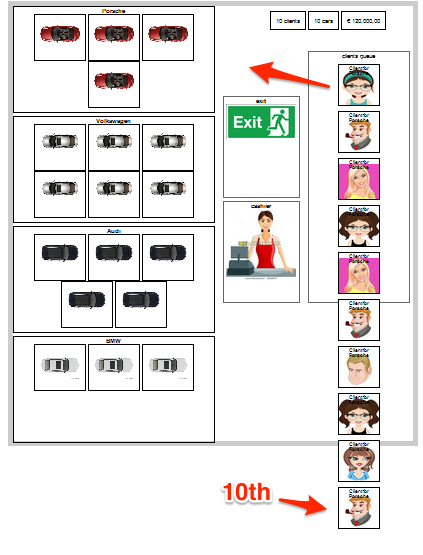


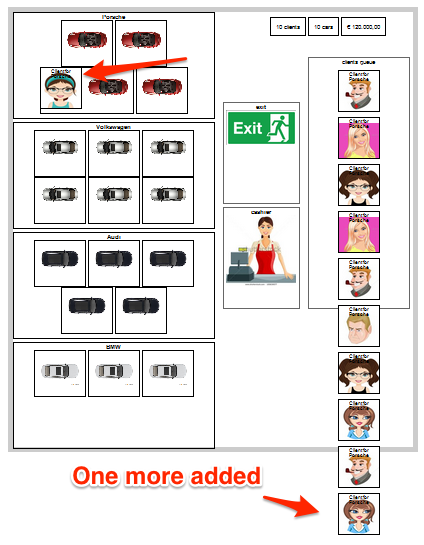
## (G) Additional processing after dropping – Porsche only

Step We need to decrease the queue count and add a new client. In dealer.js:

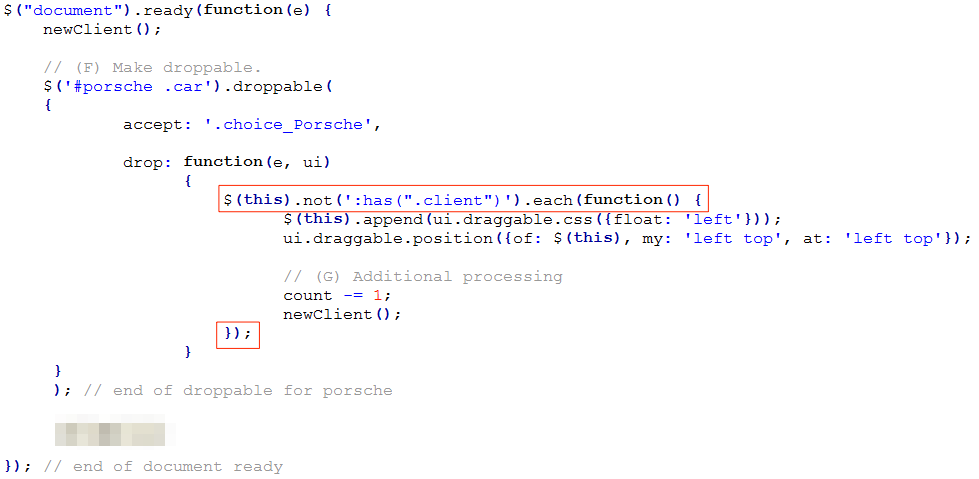


Step Test the code.

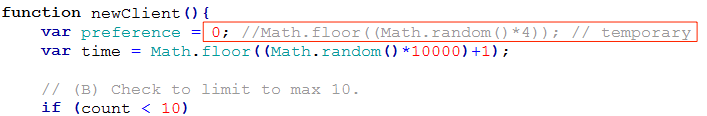




Step After one car is occupied by a client, then another client should not be able to choose the same car. In dealer.js, check that the element does not contain ‘.client’ before calling the subsequent processing:

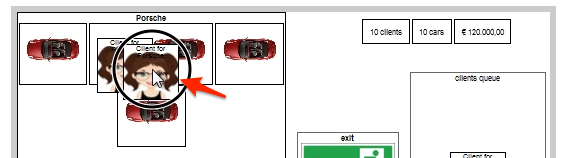


Step To simplify our testing for now, make the following temporary change to dealer.js, so that only client for porsche will be generated.

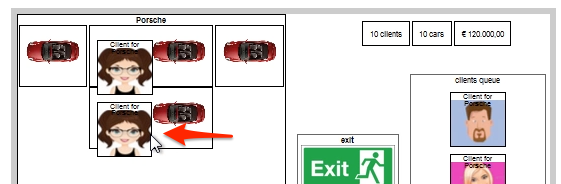


Step Test the code.

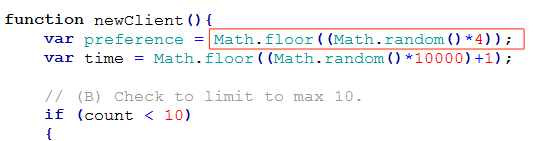
(Cannot drop into the same car slot)



(Can drop to another slot)

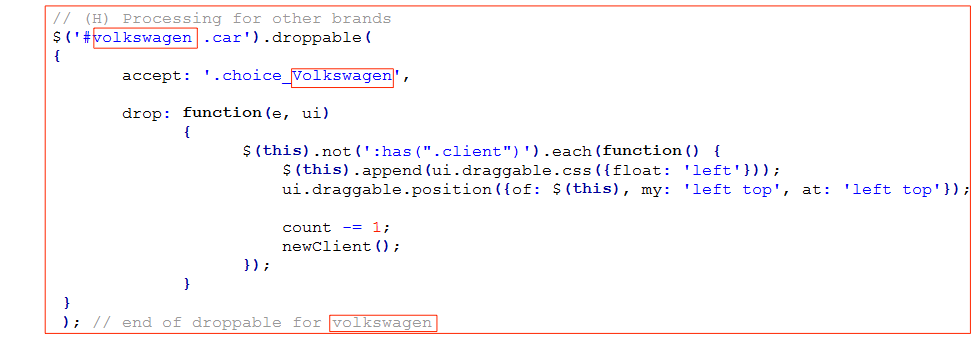
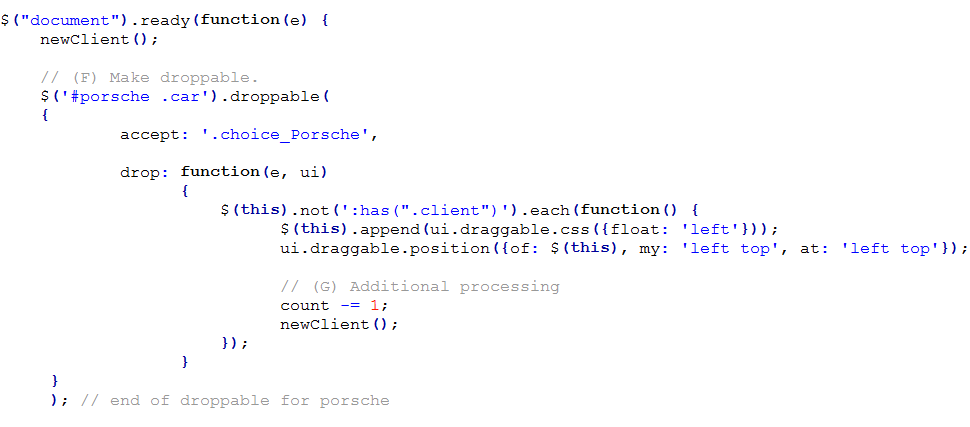


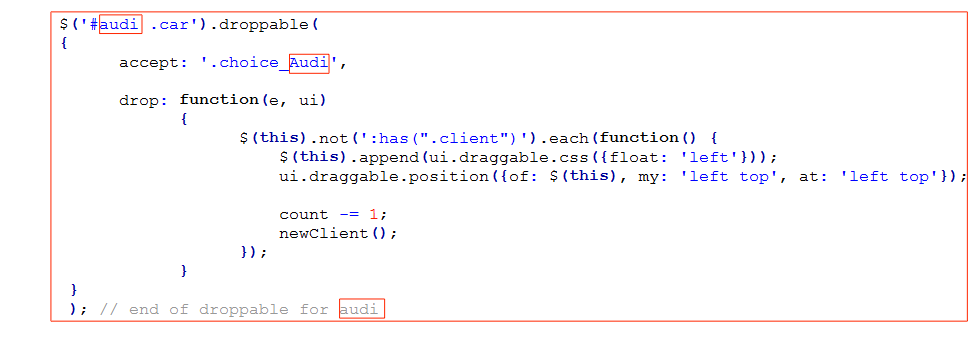
Step Change the temporary code back. In dealer.js:

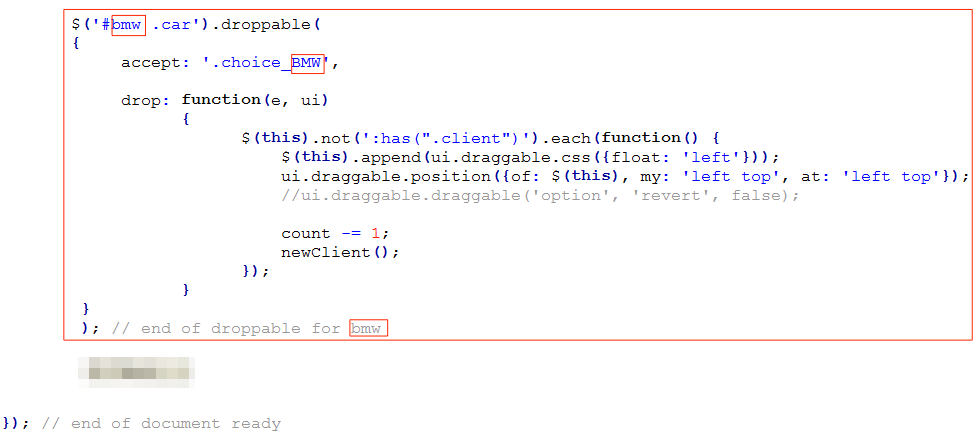


## (H) Similar processing for other brands of car

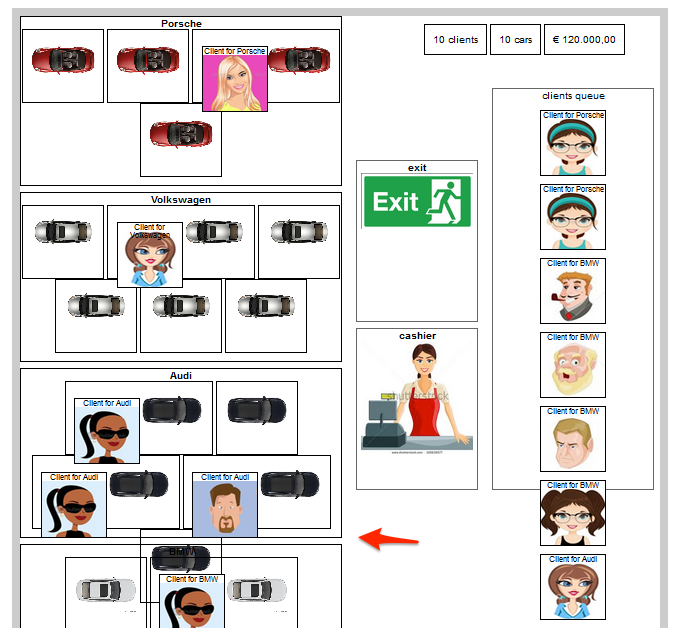
Step Copy and paste the code for the other brands. In dealer.js:



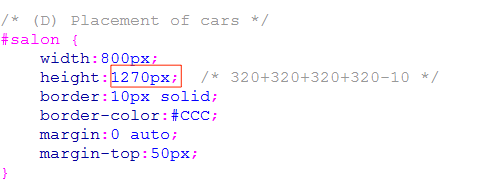
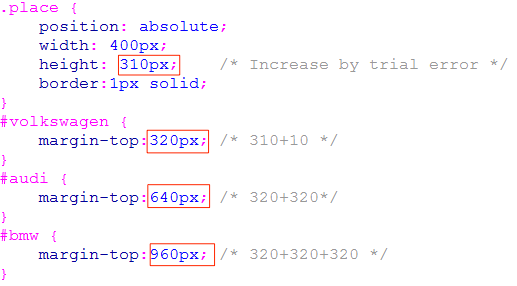




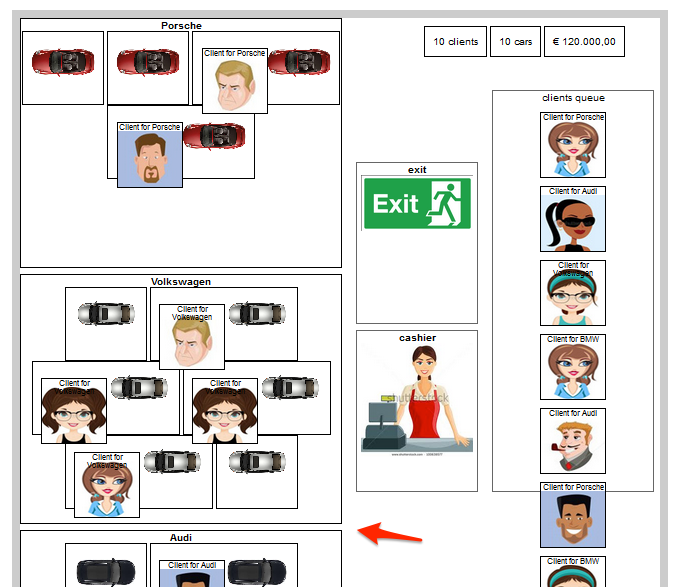
Step Test the code.



Step Observe that the layout is not sufficient and look messy. Make changes in dealer.css:

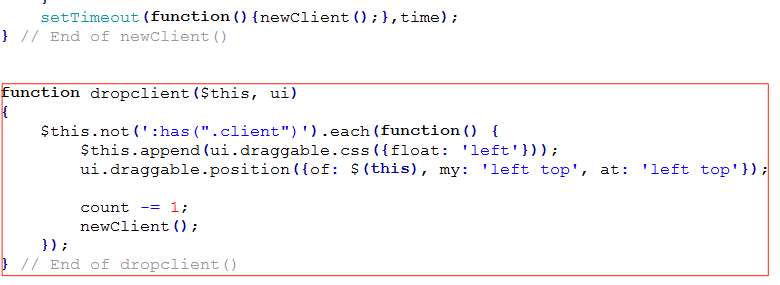
Step Test the code.

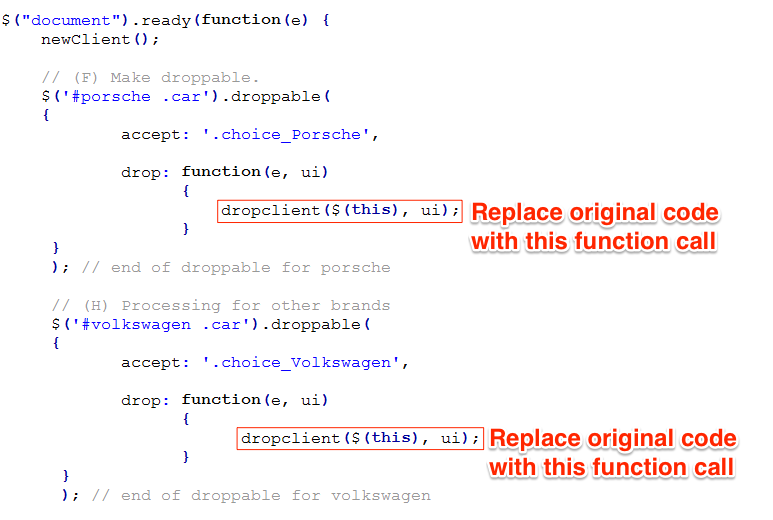


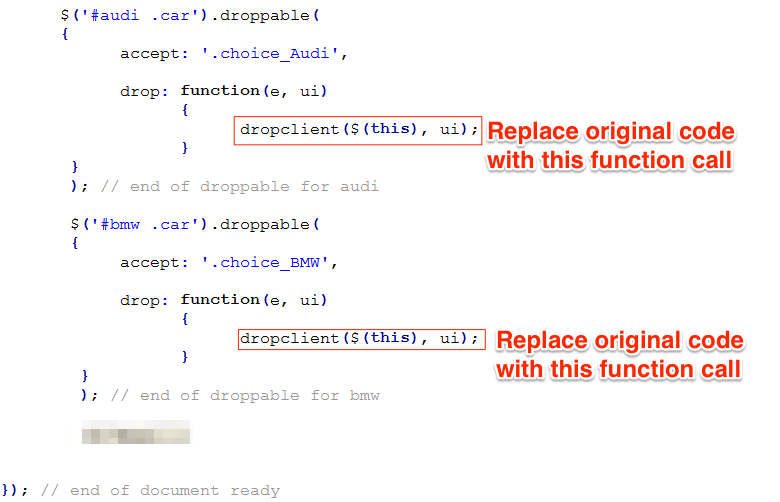
## (I) Code reuse using function

Step Since the same processing are repeated in the code we copy and paste and later on we will need to add more processing into this segment of code, it is better for us to group a segment into a new function **dropclient( )** and pass in relevant parameters.

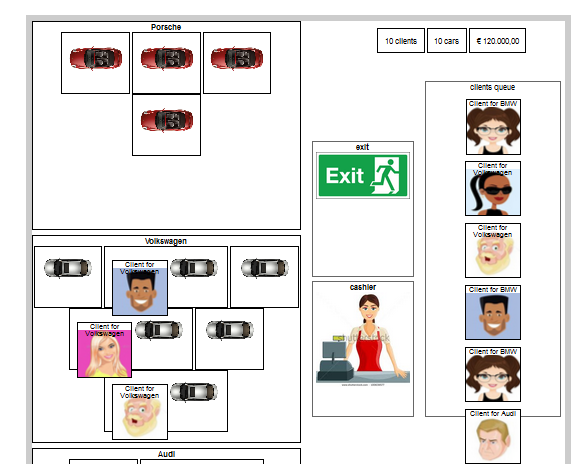
$this is just a variable name (since variable can start with letters, \_ and $). The reason why we name it this way is following convention since the content of $this is actually from $(this). While $(this) is jquery selecting from ‘this’ which is the current object.





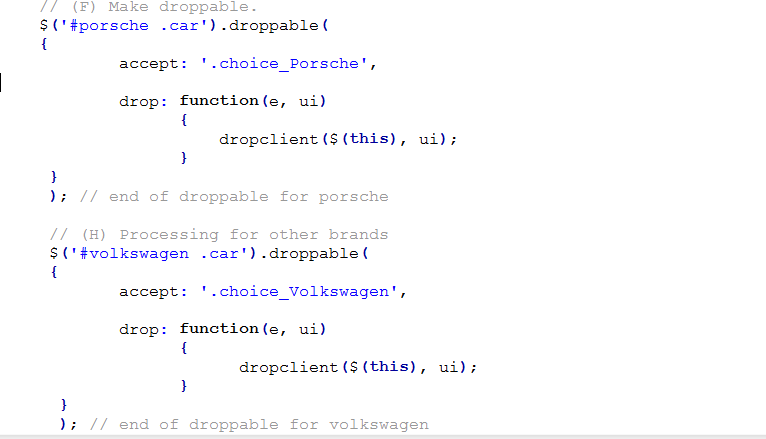


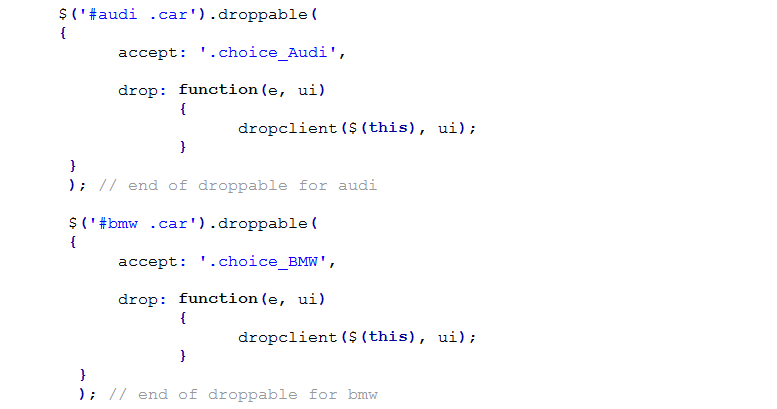
Step Test to confirm that the app still works as before.

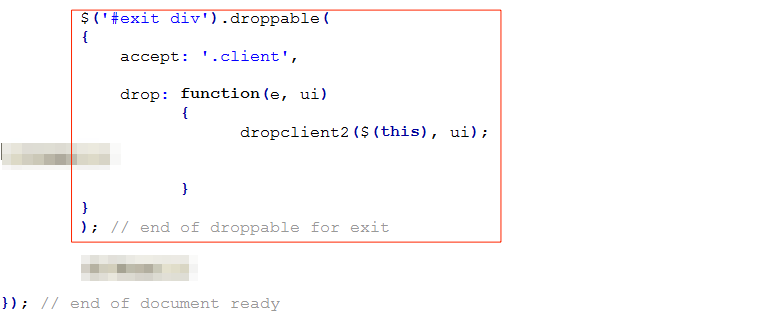


## (J) Drop client to Exit

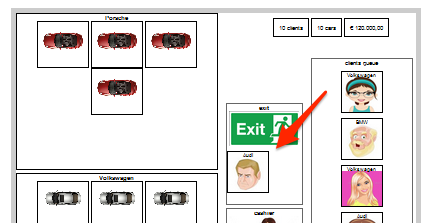
Step Try to reuse dropclient() but considering that the will be slight difference in the processing in the next few step, copy and paste the code to create a new function dropclient2():

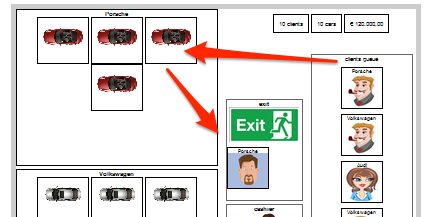




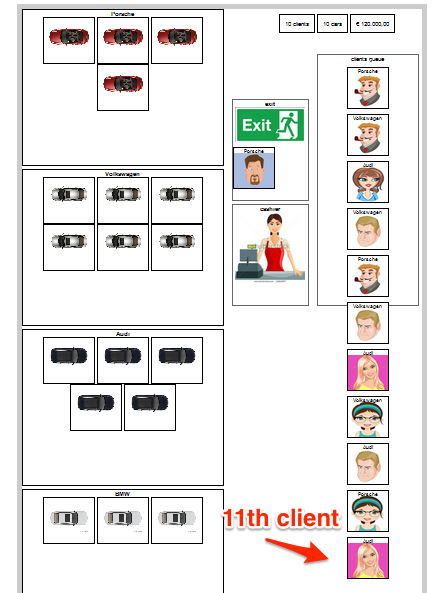


Step Test the code. Both ways should work.

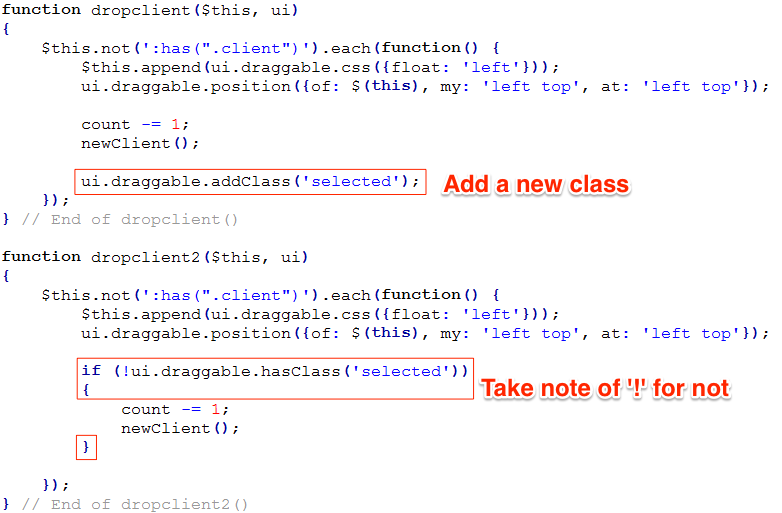




Step But observe that for the second way, one extra client is created in the queue since we add one client when drop to the car slot, then we created one more to the queue when we drop at exit.



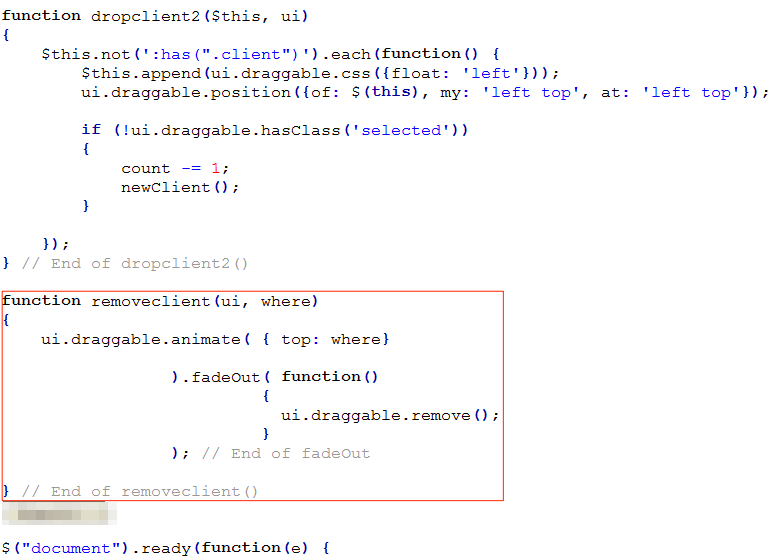
Step Modify dropclient() to add indication that the client has selected a car, then check in dropclient2() to add new client only if not coming from selected car.



Step Test code to confirm that no new client will be added when drag and drop after selected a car.

Step After dropping the client at exit, the client should be removed.

Use animate( ) t move it up a little and then use fadeOut( ) to fade and finally remove( ) to get rid of the client.

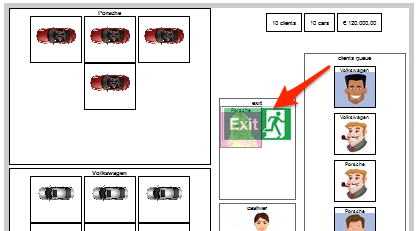


:

:

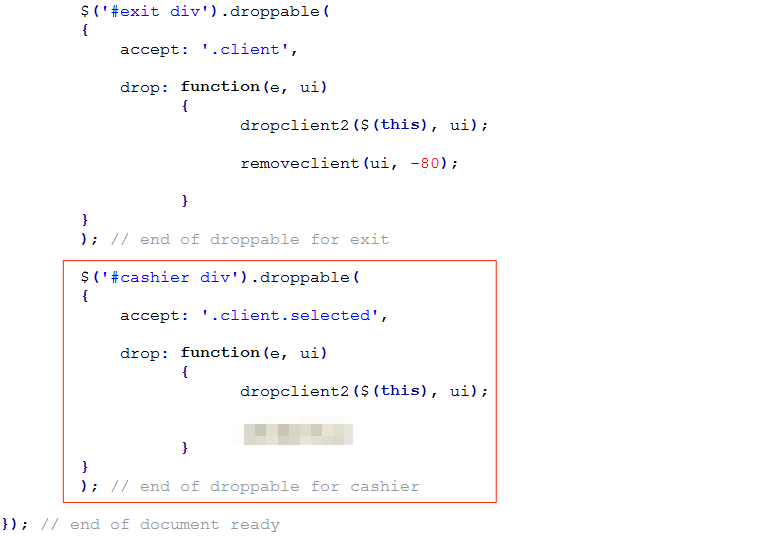


Step Test the code.

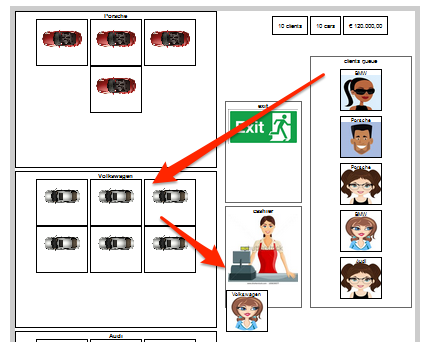


## (K) Drop client to cashier

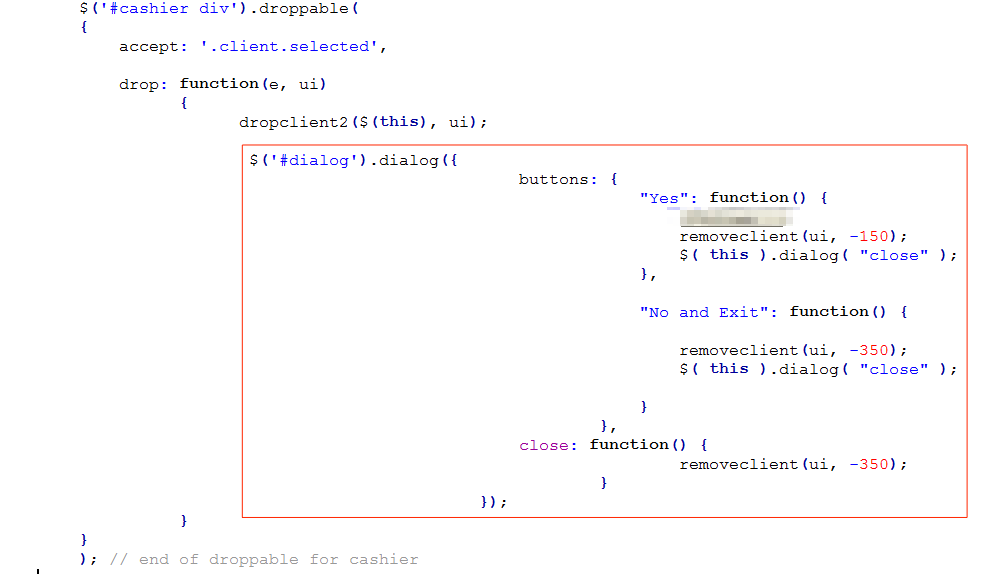
Step Reuse dropclient2( ) for dropping to cashier:



Step Test the code.

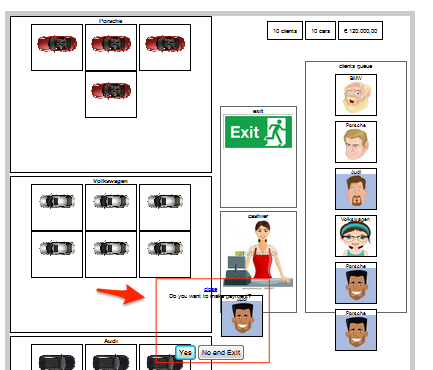


Step Use jQuery UI to add dialogue box with “YES” and “No and Exit”.

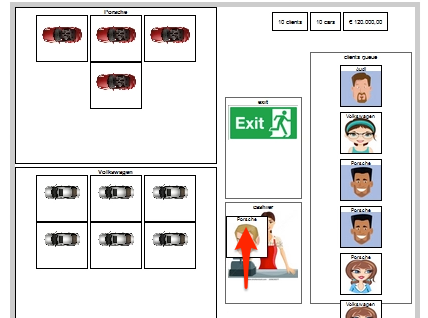


Step Test the code

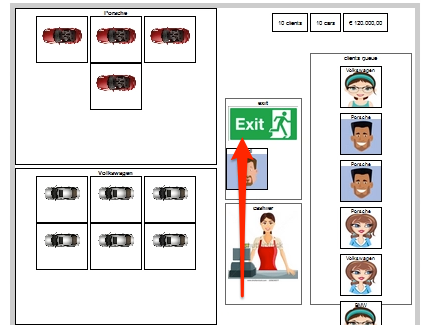
(Observe the dialogue box’s styles are missing – we will add in the next step)



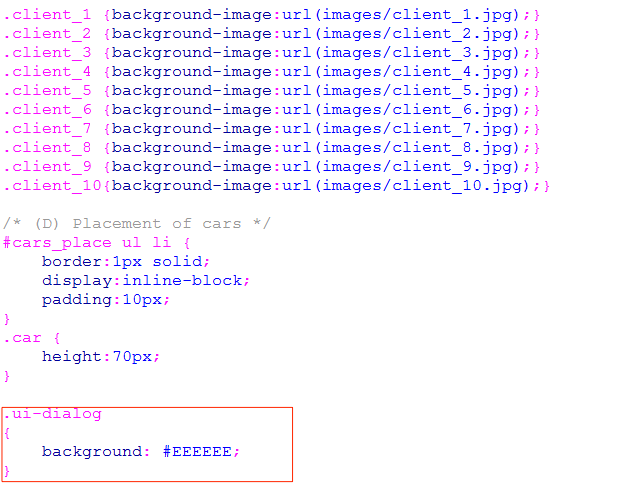
(When you click on YES)



(When you click on “No”)

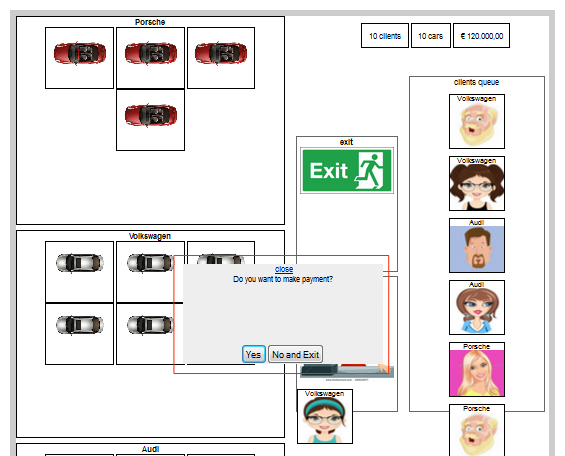


Step As the jQuery UI CSS is not included with the library, we need to incorporate minimum style to have minimum look and feel. In dealer.css:



Step Test the code.

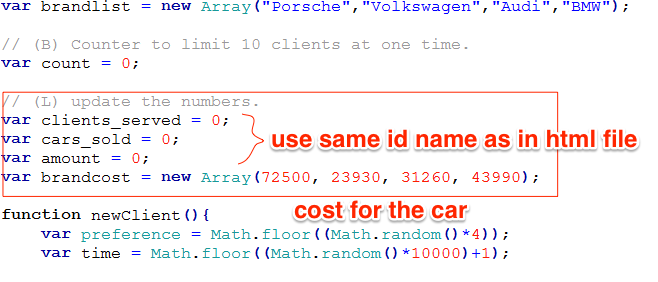
(If you have time later, you may improve the look and feel.)



## (L) Update the numbers

Step In dealer.js:

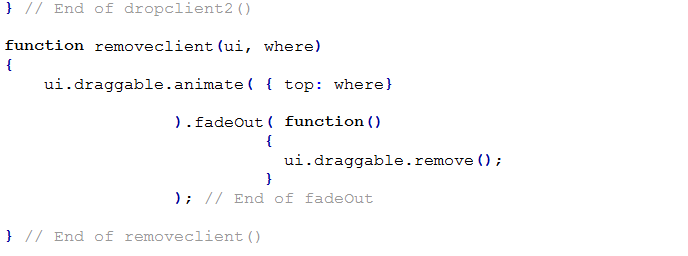
Create and initialize variables.

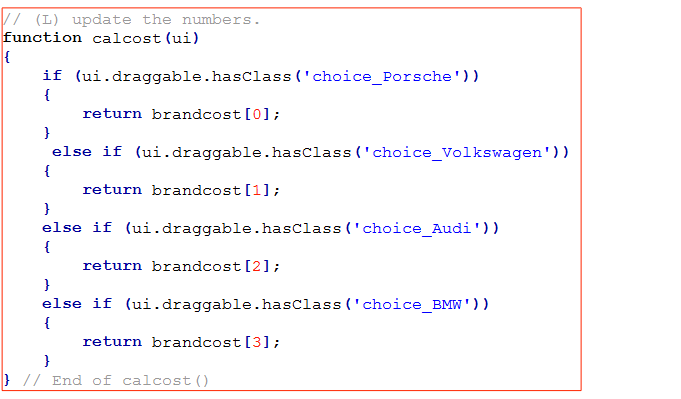


:

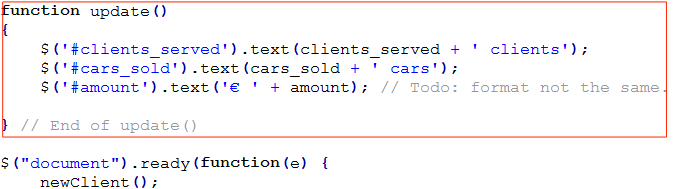
:

Create a function to return the cost of car being bought, depending on the client choice class.





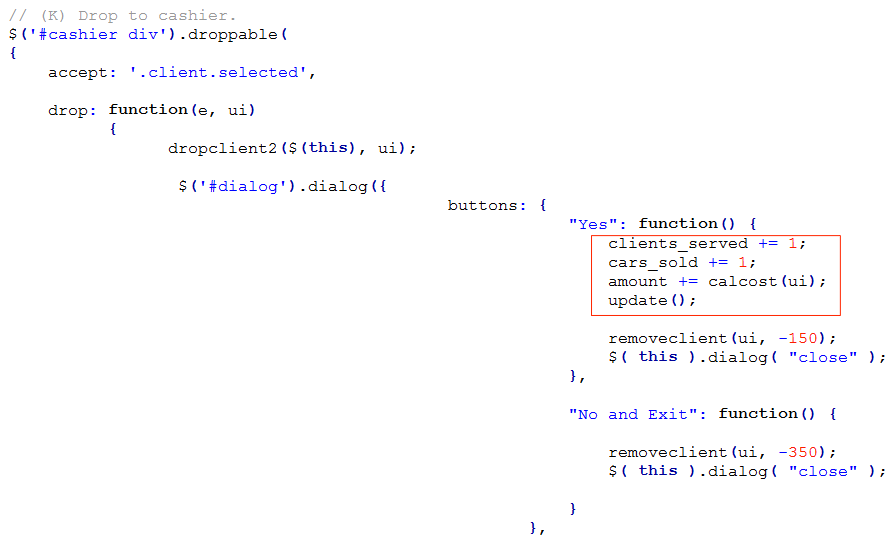
Create a new function to update the figures on the HTML page. Note that there is still one outstanding issue (the format of the output for the amount earned).



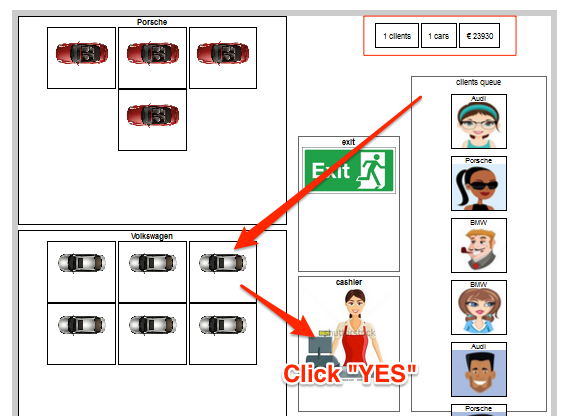
:

:

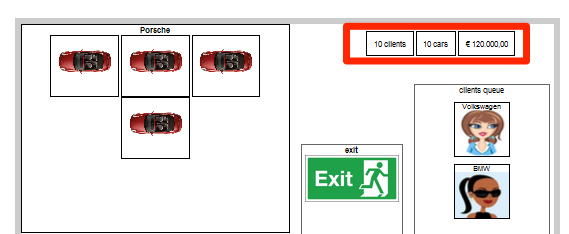
Add 1 to clients\_served and cars\_sold, add the cost of car to amount and call update( ) to update the figures in the HTML page.



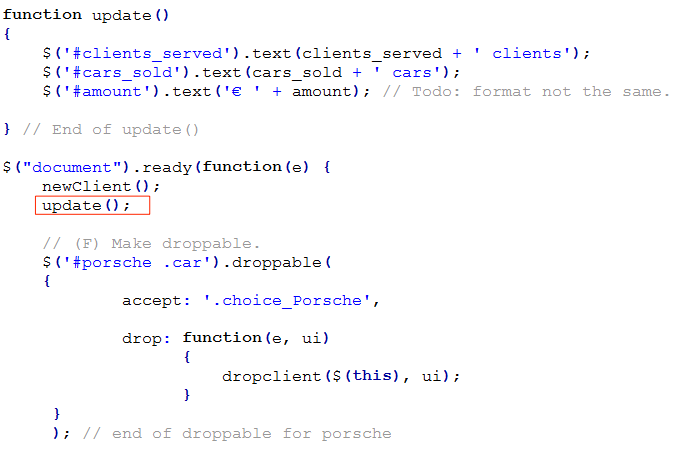
Step Test the code.



Step Observe when the app just started, the numbers are not zero’s.



Step Add code to call update( ) when the code start in dealer.js:



Step Test the code.



That’s all.

Continue to test and debug OR improve the looks and feel.

Happy coding and continue to learn coding.