HW 6 - Jade Delight, Joao Quintanilha

💡 **If you had an extra two weeks for this project, what feature(s) would you add**

- I would add figures in the table of products, so the user can better understand details of the options
- I would improve the styling of the additional window made, as for now it has a very poor HTML styling
- I would try to align the table options to the middle of the box-element
- The pages are not responsive. They work well in big screens, but for phones they would need some refinement

💡 **The thing I like the least about Javascript is…**

- JQuery. And JSON. And AJAX. I am still catching up with all the technologies and resources behind JavaScript and I confess that trying to understand all of them at the same time has been extremely excruciating for me. Nevertheless, It is always rewarding when I get to see my code working in a web page.
- I also don't like how hard it gets to test things with JavaScript. Since I am used and dependent on compilers for C and C++, I feel like I am re-learning how to ride a bike now that I have to test things on JS.

💡 **URL of the working page**

Jade Delight

https://fifth-island.github.io/Jade_Delight/

→ jade_delight.html

```
<!DOCTYPE html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Jade Delight</title>

<!-- I added the following lines to include the jQuery library, my JavaScript code file an
d my CSS stylesheet -->
<script   src="https://code.jquery.com/jquery-3.1.1.min.js"   integrity="sha256-hVVnYaiADR
TO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="   crossorigin="anonymous"></script>
<script type="text/javascript" src="code.js"></script>
<link rel="stylesheet" href="rules.css">

</head>

<body>
<script>

function MenuItem(name, cost)
{
  this.name = name;
  this.cost=cost;
}

menuItems = new Array(
  new MenuItem("Chicken Chop Suey", 5.5),
  new MenuItem("Sweet and Sour Pork", 7.25),
  new MenuItem("Shrimp Lo Mein", 6.80),
  new MenuItem("Moo Shi Chicken", 9.50),
  new MenuItem("Fried Rice", 3.25)
);

function makeSelect(name, minRange, maxRange)
{
  var t= "";
  t = "<select name='" + name + "' size='1'>";
  for (j=minRange; j<=maxRange; j++)
     t += "<option>" + j + "</option>";
  t+= "</select>";
  return t;
}

function td(content, className="")
{
  return "<td class = '" + className + "'>" + content + "</td>";
}

</script>

<h1>Jade Delight</h1>
```

```
<form>

<p class="userInfo"><label>First Name:</label> <input type="text"  name='fname' /></p>
<p class="userInfo"><label>Last Name*:</label>  <input type="text"  name='lname' /></p>
<p class="userInfo address"><label>Street*:</label> <input type="text" name='street' /></p
>
<p class="userInfo address"><label>City*:</label> <input type="text" name='city' /></p>
<p class="userInfo"><label>Phone*:</label> <input type="text"  name='phone' /></p>
<p>
  <input type="radio"  name="p_or_d" value = "pickup" checked="checked"/>Pickup
  <input type="radio"  name='p_or_d' value = 'delivery'/>
  Delivery
</p>
<table border="0" cellpadding="3">
  <tr>
    <th>Select Item</th>
    <th>Item Name</th>
    <th>Cost Each</th>
    <th>Total Cost</th>
  </tr>
<script>

  var s = "";
  for (i=0; i< menuItems.length; i++)
  {
    s += "<tr>";
    s += td(makeSelect("quan" + i, 0, 10),"selectQuantity");
    s += td(menuItems[i].name, "itemName");
    s += td("$" +menuItems[i].cost.toFixed(2), "cost");
    s += td("$<input type='text' name='cost'/>", "totalCost");
    s+= "</tr>";
  }
  document.writeln(s);

</script>
</table>
<p class="subtotal totalSection"><label>Subtotal</label>:
   $ <input type="text"  name='subtotal' id="subtotal" />
</p>
<p class="tax totalSection"><label>Mass tax 6.25%:</label>
  $ <input type="text"  name='tax' id="tax" />
</p>
<p class="total totalSection"><label>Total:</label> $ <input type="text"  name='total' id
="total" />
</p>

<input type = "button" value = "Submit Order" onclick="confirm_order();" />

</form>
</body>
</html>
```

→ code.js

```
// global variable that defines the how the food will be retrieved
var transport = "pickup";

const MASS_TAX = 0.0625;


/*        function ready
 *        purpose: update order page automatically as the user selects
 *                 its positions
 *      parameter: none
 *         return: nothing
 *           note: use of ready() method to make the function available
 *                 after loading the page
*/
$(document).ready(function() {

  initialization(); // set initial elements of the form
  transportation(); // evaluate if the city and street will be included

  cost_update(); // update the table with the proper calculations

});


/*        function transportation
 *        purpose: evaluate which method of deliverying the food
 *                 will be applied
 *      parameter: none
 *         return: nothing
 *           note: - use of trigger "click" to activate the function
 *                 - global variable transport is updated according to
 *                 the specified method
*/
function transportation() {

// evaluate if the radio button is set for pickup or delivery
$('input:radio[value="pickup"]').click(function() {
 $('.address').css("display","none");
 transport = "pickup";
});

$('input:radio[value="delivery"]').click(function()
{
 $('.address').css("display","block");
 transport = "delivery";

});
}
```

```
/*        function initialization
 *        purpose: initalize the windows eith default empty data
 *      parameter: none
 *         return: nothing
 *           note: simple function -> it could be better developed
*/

function initialization()
{
 // default: address options is not displayed
 $('.address').css("display","none");
 transport = "pickup";

}


/*        function cost_update
 *        purpose: update total values and subtotal, tax, and total
 *                 as the client select the number of products to order
 *      parameter: none
 *         return: nothing
 *           note: - numbers are rounded to 2 decimal cases with toFixed
 *                 - 0.0625 is the value for tax -> CONSTANT ELEM = MASS_TAX
*/
function cost_update() {
 $("select").on("input", function() { // occurr whenever a change is noticed
  var subtotal = 0;
  $("select option:selected").each(function(idx) {

   // walk through the products updating them accordingly
   var qty = $(this).val();
   var cost_per_product = qty * menuItems[idx].cost;
   $("input[name='cost']:eq(" + idx + ")").attr("value", (cost_per_product).toFixed(2));
   subtotal = subtotal + cost_per_product; // subtotal is updated every time a change happ
ens
  });

  taxes = subtotal * MASS_TAX; // obtain state tax calculation

  // summarized calculations are also updated
  $("input[name='subtotal']").attr("value", (subtotal).toFixed(2));
  $("input[name='tax']").attr("value", (taxes).toFixed(2));
  $("input[name='total']").attr("value", (subtotal + taxes).toFixed(2));

 });
}

/*        function confirm_order
 *        purpose: submit all the input values generating a ticket receipt
 *      parameter: none
 *         return: nothing
 *           note: - delegate validation of the web page with the
 *                 function validation
 *                 - delegate time calculation for print_time() function
```

```
 *                 and for the usage of the Date class
 *                 - helper function print_item(source) retrieve the whole
 *                 row from the HTML and generate a line informing
 *                 the quantity of products and the related prices
 *                 - event function that is called every time the button
 *                 submit is pressed on the html text
 *                 - print HTML text into a new window by using window.open
*/
function confirm_order() {
 // check for all the requirements of the submission first
 // it warns the client wether something went wrong
 var boolean_valid = validation();

 // only proceed for the receipt generation if the validation is succesful
 if (boolean_valid) {

  // pop up message confirming the success of the transaction
  alert("Thank you for buying with us");

  // initalize a window object
  var summary = window.open();

  // create a string array to generate all the lines for the new window html page
  lines = "<head>";
  lines += "<link rel='stylesheet' href='rules.css'></link>" // insert the general CSS rul
es to the file
  lines += "</head>";
  lines += "<body style='background-color:#f4a460'>"; // updated according to the general
 CSS rules
  lines += "<h1>Thank You!</h1><br />";

  lines += "<div class='receipt'>";

  lines += "<p>Here is your receipt:</p>";

  lines += print_item($("table").find("tr:eq(1)"));
  lines += print_item($("table").find("tr:eq(2)"));
  lines += print_item($("table").find("tr:eq(3)"));
  lines += print_item($("table").find("tr:eq(4)"));
  lines += print_item($("table").find("tr:eq(5)"));

  lines += "<br><br>";


  lines += "<strong>Subtotal:</strong> $" + $('input:text[name="subtotal"]').val() + "<br
>";
  lines += "<strong>Tax:</strong> $" + $('input:text[name="tax"]').val() + "<br>";
  lines += "<strong>Total:</strong> $" + $('input:text[name="total"]').val() + "<br>";

  lines += "<br><br>" + print_time() + "<br>";

  lines += "</div>";

  lines += "</body>";
```

```
  // print whole file with new lines
  summary.document.writeln(lines);
 }

}

/*        function print_item
 *        purpose: get location of a whole row from the table and convert
 *                 its data into a string
 *      parameter: line row from table of products
 *         return: string informing the type of product, quantity and
 *                 total price
 *           note: - helper function for confirm_order() event function
*/
function print_item(line) {

 qty = line.find("td:eq(0)>select").val();
 if (qty == "0" || qty == "0.00") { // the string is only formed if products > 0
  return "";

 } else {

  var total_cost = line.find("td:eq(3)>input").val();
  var detail = line.find("td:eq(1)").text();
  return (detail + ": " + qty + " by the price of " + total_cost + "<br>");

 }
}

/*        function print_time
 *        purpose: define how much time will take according to the delivery
 *                 method previously defined
 *      parameter: nothing
 *         return: string informing the time in hours and minutes
 *           note: - usage of Date class to figure out current time
 *                 - usage of global variable to identify the defined method
*/
function print_time() {

 var line;
 // if statements perform the same task
 // only difference is that for delivery it takes the double of time
 if (transport == "pickup") {
   var mydate = new Date();
   line = new Date(mydate.getTime() + 20*60000);
   h = (line.getHours()<10?'0':'') + line.getHours(),
   m = (line.getMinutes()<10?'0':'') + line.getMinutes();
   line = "Pickup time: " + h + ':' + m;
 } else {
  var mydate = new Date();
   line = new Date(mydate.getTime() + 40*60000);
   h = (line.getHours()<10?'0':'') + line.getHours(),
   m = (line.getMinutes()<10?'0':'') + line.getMinutes();
```

```
    line = "Pickup time: " + h + ':' + m;
 }
 return line;

}

/*        function validation
 *        purpose: confirm if last name, phone, street and city are
 *                 properly included in the submission
 *      parameter: nothing
 *         return: boolean value that indicates success or
 *                 failure of validation
 *           note: - it defines if the printing of the new window
 *                 will be performed or if the client needs to adjust errors
 *                 - use of helper functions to check for specific validations
*/
function validation() {
 var validated = true;

 // check if the last name is not empty
 validated = valid_helper($('input:text[name="lname"]'));
 if (!validated) {
  return validated;
 }

 // check phone validity -> extract only the numbers of the string
 // and affer if they match to the expected lengths
 var phone = $('input[name="phone"]').val();
 var phone_address = $('input[name="phone"]');

 // delete all non-numbers of the phone input
 var digits = String(phone.replace(/[^0-9]/g, ""));

 // if the remaining matches with the expected lenght, the phone number is correct
 if (digits.length != 7 && digits.length != 10) {
  alert("Must enter a valid phone number");
  phone_address.focus();
  phone_address.select();
  validated = false;
 }

 // assert if previous lines of codes found an invalid elemenet
 if (!validated) {
  return validated;
 }

 // check if street and city inputs are filled, in case the client wants delivery mehtod
 if (transport == "delivery") {
  validated = valid_helper($('input:text[name="street"]'));
  validated = valid_helper($('input:text[name="city"]'));
 }

 // assert if previous lines of code find an invalid element
 if (!validated) {
```

```
  return validated;
 }

 // check if at least one product has been chosen
 validated = valid_products_helper();

 // return true if has passed the assertions
 return validated;
}

/*        function valid_helper
 *        purpose: check if input box is empty
 *      parameter: input txt
 *         return: boolean value whether the text is empty or not
 *           note: - in case of empty boxes, an alert will instruct the client
*/
function valid_helper(source) {
 // check if the last name is not empty
 var name = source;
 if(name.val() == "") {
  alert("Must enter a value for the text-box");
  name.focus();
  name.select();
  return false;
 } else {
   return true;
 }
}

/*        function valid_products_helper
 *        purpose: check if at least one product has been added to the cart
 *      parameter: nothing
 *         return: boolean value whether the cart is empty or not
 *           note: - in case of empty cart, an alert will instruct the client
*/
function valid_products_helper() {
 var marker = false;
 $("select option:selected").each(function() {
  if ($(this).val() != 0)
    marker = true;    // in case one of the values is not empty
 });

 if (!marker) {
  alert("Must choose at least one product");
  $(this).focus();
 }
 return marker;

}
```

→ CSS stylesheet

```
@import url('https://fonts.googleapis.com/css2?family=Pacifico&family=Roboto+Condensed:ita
l,wght@0,300;0,400;0,700;1,400&display=swap');

/* font-family: 'Pacifico', cursive;
font-family: 'Roboto Condensed', sans-serif; */

h1 {
 font-family: 'Pacifico', cursive;
 text-align: center;
 color: #60B0F4;
 font-size: 80px;
 letter-spacing: 7px;
 -webkit-text-stroke: 1px black;
   /* color: white; */
   text-shadow:
       3px 3px 0 #000,
     -1px -1px 0 #000,
      1px -1px 0 #000,
      -1px 1px 0 #000,
       1px 1px 0 #000;
}

body {
 background-color: #f4a460;
 font-family: 'Roboto Condensed', sans-serif;
}

form {
 padding: 16px;
 width: 50%;
 margin: auto;
 background-color: #b1cde6;
 border: #b1cde6;
 border-radius: 2%;
}

.userInfo input {
 position: absolute; left: 35%;
}

.totalSection input {
 position: absolute; left: 35%;
}

.receipt {
 padding: 16px;
 width: 50%;
 margin: auto;
 background-color: #b1cde6;
```

```
 border: #b1cde6;
 border-radius: 2%;

}
```