Name: Sciddhanto Sinha

Roll no: 2213111

Batch: A

Experiment No: 6

Write a Javascript to create shopping applications which adds the books in cart, updates the existing books, delete the book and display the same. Create proper UI for the same.

Objective: To learn about concept and implementation of JAVASCRIPT

Theory:

What is Javascript. Why it is called Client side Scripting language Client side: JavaScript is a client-side language, which means it gets executed at the client side (i.e, user side). On contrary, PHP is a server-side scripting language, as it gets executed at server. Whenever you browse the web, all the HTML, CSS & JS files are fetched from the server & then executed/interpreted at your side by your browser. Scripting language: Since it is interpreted rather than compiled & are úsed to execute tasks one-by-one. More professionally, A scripting or script language is a programming language that supports scripts: programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted

Explain Javascript functions used in above programs like with syntax and example

getElementByID()

The getElementById() method returns an element with a specified value. The getElementById() method returns null if the element does not exist.
document.getElementById("demo").style.color
= "red";

•innerHTML()

The innerHTML property sets or returns the HTML content (inner HTML) of an element. let html
= document.getElementById("myList").innerHTML;

value()

The Object.values() method returns an array of a given object's own enumerable property values, in the same order as that provided by a for...in loop. (The only difference is that a for...in loop enumerates properties in the prototype chain as well.) console.log(Object.values(object1))

•parseInt()

The parseInt() function parses a string argument and returns an integer of the specified radix (the base in mathematical numeral systems). function roughScale(x, base) { const parsed = parseInt(x, base); if (isNaN(parsed)) { return 0; } return parsed * 100;
}

DOM

- It is an application programming interface.
- Dom represents a document as a hierarchical tree of nodes allowing dev to add, remove, and modify individual part of the page.
- Document node represents every document as a root. In this example only child of document node is html which is called as document element.
- Every piece of markup can be represented by a node in the tree: HTML elements are represented by element nodes, attributes are represented by attribute nodes, the document type is represented by a document type node, and comments are represented by comment nodes.
- In total, there are 12 node types, all of which inherit from a base type.
- DOM Level 1 describes an interface called Node that is to be implemented by all node types in the DOM.
- Every node has a nodeType property that indicates the type of node that it is. Node types are represented by one of the following 12 numeric constants on the Node type:

a. Node.ELEMENT_NODE (1)
b. Node.ATTRIBUTE_NODE (2)
c. Node.TEXT_NODE (3)
d. Node.CDATA_SECTION_NODE (4)
e. Node.ENTITY_REFERENCE_NODE (5)
f. Node.ENTITY_NODE (6)
g. Node.PROCESSING_INSTRUCTION_NODE (7)
h. Node.COMMENT_NODE (8)
i. Node.DOCUMENT_NODE (9)
j. Node.DOCUMENT_TYPE_NODE (10)
k. Node.DOCUMENT_FRAGMENT_NODE (11)
l. Node.NOTATION_NODE (12)

```
if (someNode.nodeType == Node.ELEMENT_NODE)
{
   alert(“Node is an element.”); //won’t work in IE < 9
}
```

This example compares the someNode.nodeType to the Node.ELEMENT_NODE constant. If they’re equal, it means someNode is actually an element. Unfortunately, since Internet Explorer 8 and earlier doesn’t expose the Node type constructor, this code will cause an error. For cross-browser compatibility, it’s best to compare the nodeType property against a numeric value, as in the following:

```
if (someNode.nodeType == 1)
    {
       alert(“Node is an element.”); //works in all browsers
```

}

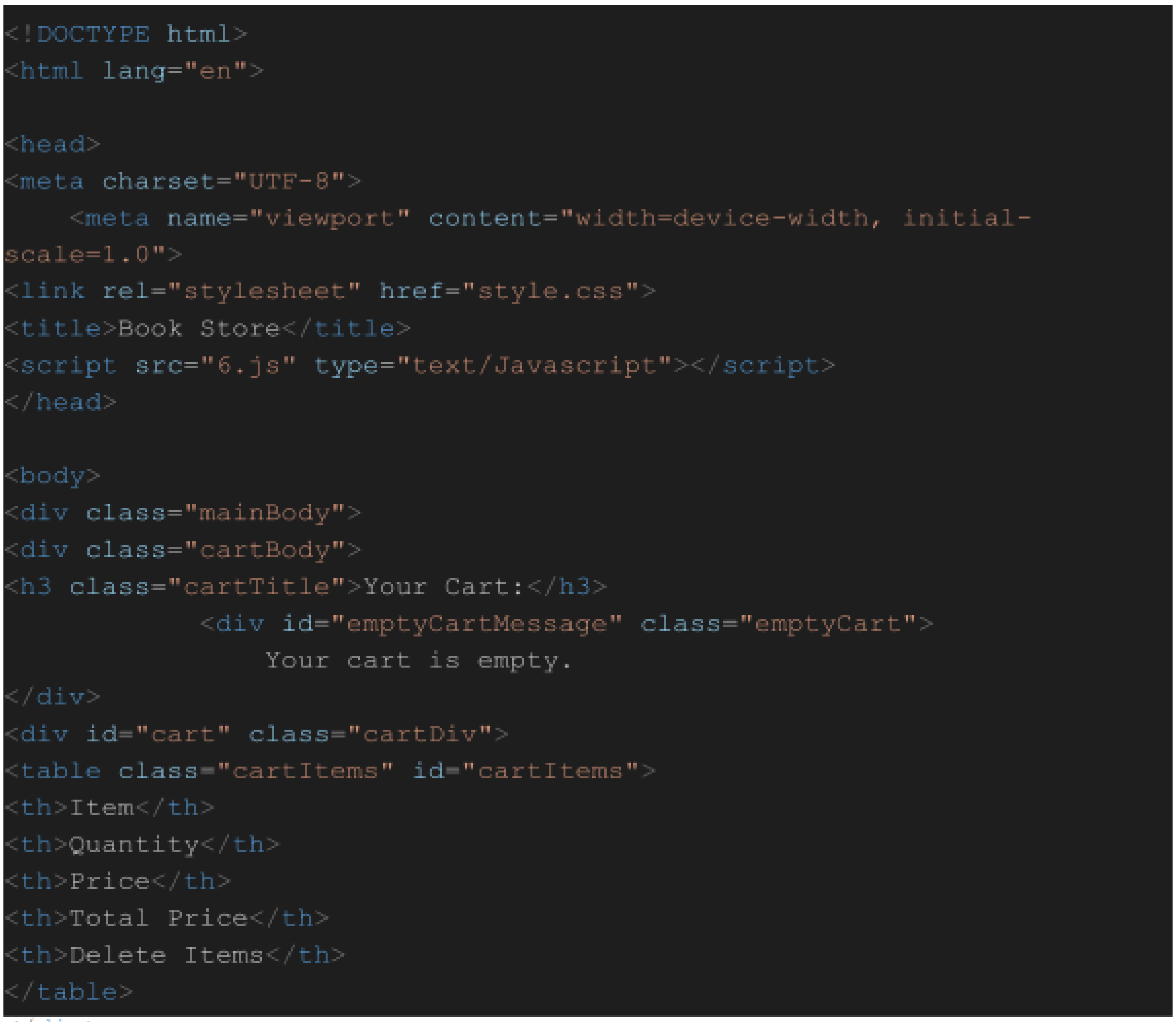
Properties of note type

- Two properties, nodeName and nodeValue, give specifi c information about the node. The values of these properties are completely dependent on the node type.
- nodeName is always equal to the element's tag name, and nodeValue is always null.
- Each node has a childNodes property containing a NodeList. A NodeList is an array- like object used to store an ordered list of nodes that are accessible by position. Keep in mind that a NodeList is not an instance of Array even though its values can be accessed using bracket notation and the length property is present.
- Node Relationship child parent

Conclusion:
Thus, we have successfully implemented the program.

Code:

HTML



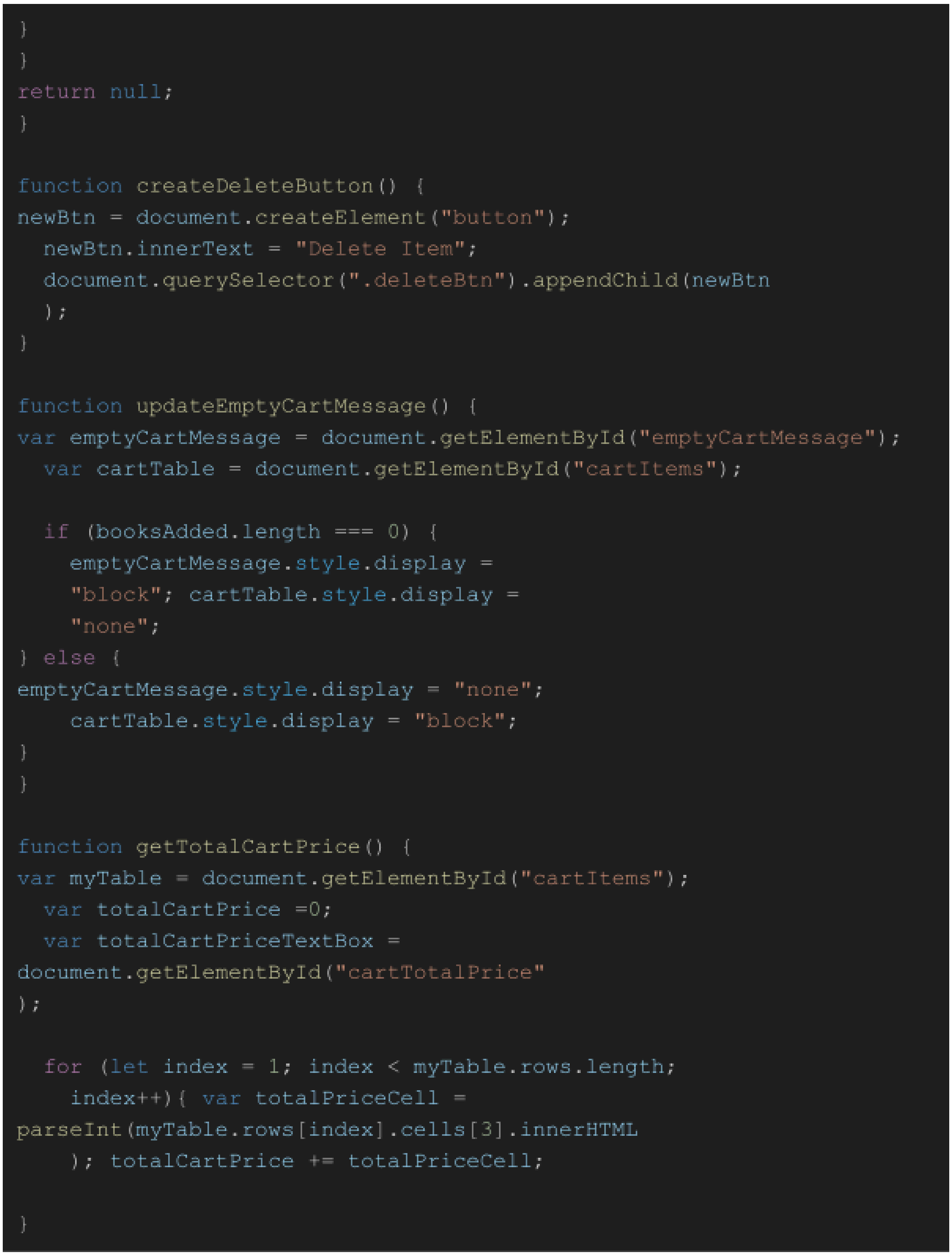
```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
<link rel="stylesheet" href="style.css">
<title>Book Store</title>
<script src="6.js" type="text/Javascript"></script>
</head>

<body>
<div class="mainBody">
<div class="cartBody">
<h3 class="cartTitle">Your Cart:</h3>
            <div id="emptyCartMessage" class="emptyCart">
                Your cart is empty.
</div>
<div id="cart" class="cartDiv">
<table class="cartItems" id="cartItems">
<th>Item</th>
<th>Quantity</th>
<th>Price</th>
<th>Total Price</th>
<th>Delete Items</th>
</table>
</div>
</div>
```

```html
        <div class="cartTotal">
            <input type="text" id="cartTotalPrice" readonly>
            <input type="button" name="" id="checoutButton"
value="Checkout">
        </div>

        <h3 class="cartTitle">Available Books</h3>
        <div id="menu" class="menuDiv">
            <table class="menuTable">
                <th>Book Title</th>
                <th>Author</th>
                <th>Genre</th>
                <th>Price</th>
                <th></th>
                <tr>
                    <td>Atomic Habits</td>
                    <td>James Clear</td>
                    <td>Genre</td>
                    <td>Rs.357</td>
                    <td><input type="button" value="Add to Cart"
onclick="addToCart(0)"></td>
                </tr>
                <tr>
                    <td>Do It Today</td>
                    <td>Darius Foroux</td>
                    <td>Genre</td>
                    <td>Rs.127</td>
                    <td><input type="button" value="Add to Cart"
onclick="addToCart(1)"></td>
                </tr>
                <tr>
                    <td>Think Straight</td>
                    <td>Darius Foroux</td>
                    <td>Genre</td>
                    <td>Rs.194</td>
                    <td><input type="button" value="Add to Cart"
onclick="addToCart(2)"></td>
                </tr>
                <tr>
                    <td>The Psychology Of Money</td>
                    <td>Morgan Housel</td>
                    <td>Genre</td>
                    <td>Rs.250</td>
                    <td><input type="button" value="Add to Cart"
onclick="addToCart(3)"></td>
                </tr>
            </table>
        </div>
    </div>
```

```
</body>

</html>
```

JAVASCRIPT:

```
var bookQuantity = 0;

var booksName = [
"Atomic Habits by James Clear", "Do
  It Today by Darius Foroux", "Think
  Straight by Darius Foroux",
"The Psychology Of Money by Morgan Housel",
];
var bookPrices = [357, 127, 194, 250];
var booksAdded = [];
var totalCartPrice = 0;
window.onload = function () {
updateEmptyCartMessage();
  getTotalCartPrice();
};
function addToCart(bookNo) {
var myTable = document.getElementById("cartItems");

var bookName = booksName[bookNo]; var
  bookPrice = bookPrices[bookNo];
var existingRowIndex = checkPresent(bookName);

  if (existingRowIndex !== null) {

var rowToUpdate = myTable.rows[existingRowIndex + 1]; var
    quantityCell =
rowToUpdate.cells[1];
var currentQuantity = parseInt(quantityCell.innerHTML);
    quantityCell.innerHTML = currentQuantity + 1;
var totalCell = rowToUpdate.cells[3]; totalCell.innerHTML =
    bookPrice * (currentQuantity + 1); getTotalCartPrice();
} else {
var newRow = myTable.insertRow(myTable.rows.length);
    booksAdded.push(bookName);
    var item = newRow.insertCell(0);
    var quantity = newRow.insertCell(1);
    var price = newRow.insertCell(2);
    var total = newRow.insertCell(3);
    var deleteCell = newRow.insertCell(4);

    item.innerHTML = bookName;
    price.innerHTML = bookPrice;
    quantity.innerHTML = 1;
    total.innerHTML = bookPrice;

    var deleteButton = document.createElement("button");
    deleteButton.innerText = "Remove item";
    deleteButton.onclick = function () {
      deleteCartItem(newRow);
    };
    deleteCell.appendChild(deleteButton);
    getTotalCartPrice();
    updateEmptyCartMessage();
  }
}

function deleteCartItem(row) {
  var rowIndex = row.rowIndex;
  var quantityCell = row.cells[1];
  var currentQuantity = parseInt(quantityCell.innerHTML);

  if (currentQuantity > 1) {
    quantityCell.innerHTML = currentQuantity - 1;
    var totalCell = row.cells[3];
    var bookPrice = parseFloat(row.cells[2].innerHTML);
    totalCell.innerHTML = bookPrice * (currentQuantity - 1);
  } else {
    row.remove();
    var bookName = row.cells[0].innerHTML;
    var indexToRemove = booksAdded.indexOf(bookName);
    if (indexToRemove !== -1) {
      booksAdded.splice(indexToRemove, 1);
    }
  }
  getTotalCartPrice();
  updateEmptyCartMessage();
}

function checkPresent(name) {
  for (let index = 0; index < booksAdded.length; index++) {
    if (name == booksAdded[index]) {
      return index;
      break;
```

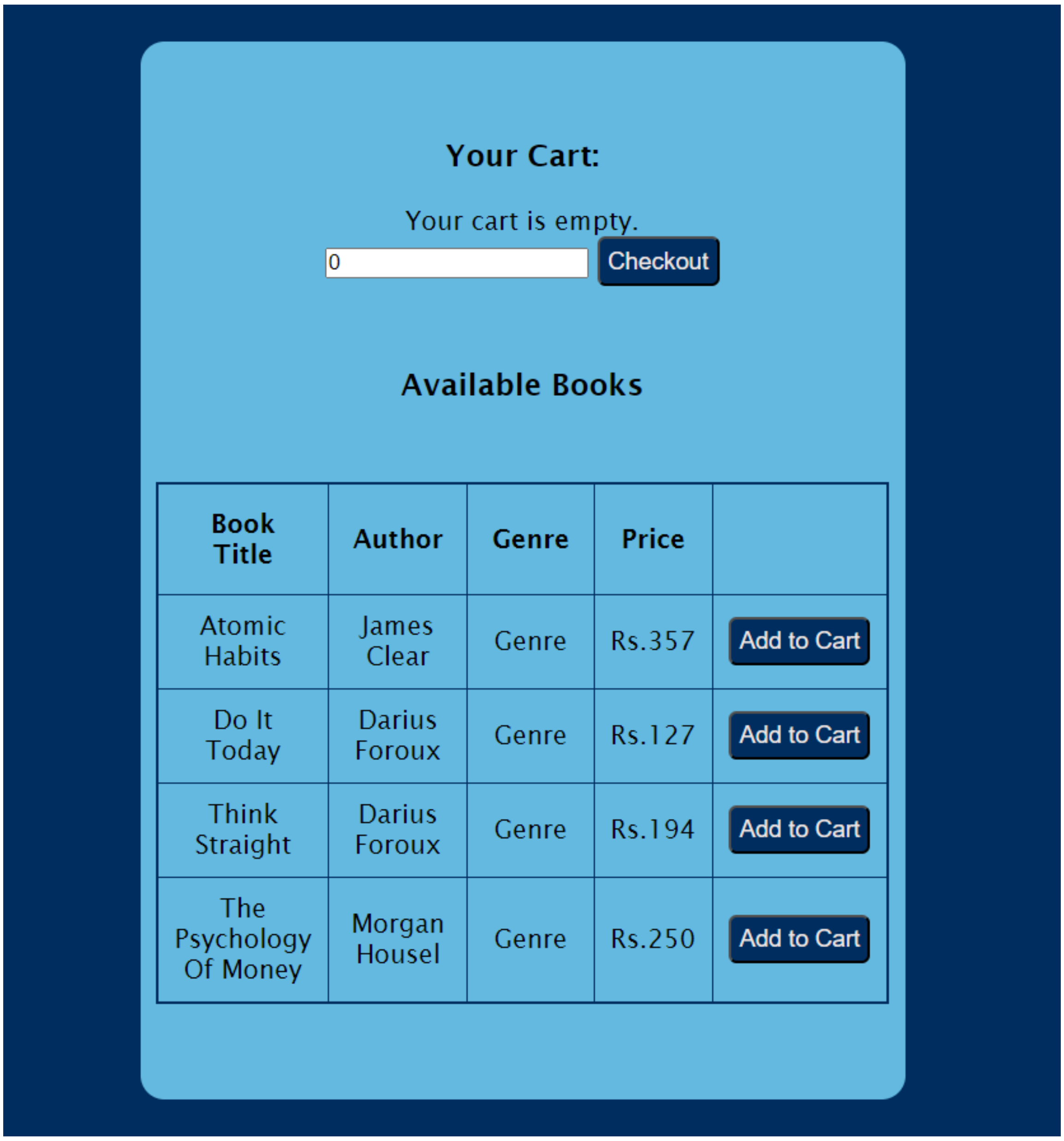


```
}
}
return null;
}

function createDeleteButton() {
newBtn = document.createElement("button");
  newBtn.innerText = "Delete Item";
  document.querySelector(".deleteBtn").appendChild(newBtn
  );
}

function updateEmptyCartMessage() {
var emptyCartMessage = document.getElementById("emptyCartMessage");
  var cartTable = document.getElementById("cartItems");

  if (booksAdded.length === 0) {
    emptyCartMessage.style.display =
    "block"; cartTable.style.display =
    "none";
} else {
emptyCartMessage.style.display = "none";
    cartTable.style.display = "block";
}
}

function getTotalCartPrice() {
var myTable = document.getElementById("cartItems");
  var totalCartPrice =0;
  var totalCartPriceTextBox =
document.getElementById("cartTotalPrice"
);

  for (let index = 1; index < myTable.rows.length;
    index++){ var totalPriceCell =
parseInt(myTable.rows[index].cells[3].innerHTML
    ); totalCartPrice += totalPriceCell;

}
```

CSS:

```css
* {
    margin: 0px;
    padding: 0px;
}

body {
    width: 100%;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    background-color: #003060;

}

.cartItems {
    padding: 25px;
    margin: 55px auto;
    border-collapse: collapse;
}

.cartDiv th {
    text-align: center;
    padding: 5px;
    border: 1px solid black;
}

.cartDiv td {
    text-align: center;
    padding: 25px;
    border: 1px solid #003060;
}

.cartDiv button {

    background: none;
    padding: 5px;
    border-radius: 5px;
    box-shadow: 0 1.5px 0 rgba(0, 0, 0, 20);
    background-color: #003060;
    color: #F4EAE6;
    border-style: none;

}

.cartTitle {
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande',
'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    margin-bottom: 0px;
    margin-top: 50px;
    text-align: center;
    padding-top: 1px;
}

.emptyCart {
    margin-top: 20px;
    text-align: center;
}

input[type="button"] {
    font-size: 15px;
    padding: 5px;
    border-radius: 5px;
    box-shadow: 0 2 0 black;
    background-color: #003060;
    color: #F4EAE6;

}

.menuTable {

    border: 2px solid #003060;
    margin: 50px auto;
    text-align: center;
    border-radius: 10px;
    border-collapse: collapse;
}

.menuTable th {
    padding: 15px;
    border: 1px solid#003060;
}

.menuTable td {
    padding: 10px;
    border: 1px solid #003060;
    border-radius: 2px;
}

.mainBody {
    overflow: auto;
    width: 30%;
    margin: 40px auto;
    padding: 10px;
    border-radius: 15px;
    background-color: #68BBE3;
}
.cartTotal{
    text-align: center;
}
```

OUTPUT :

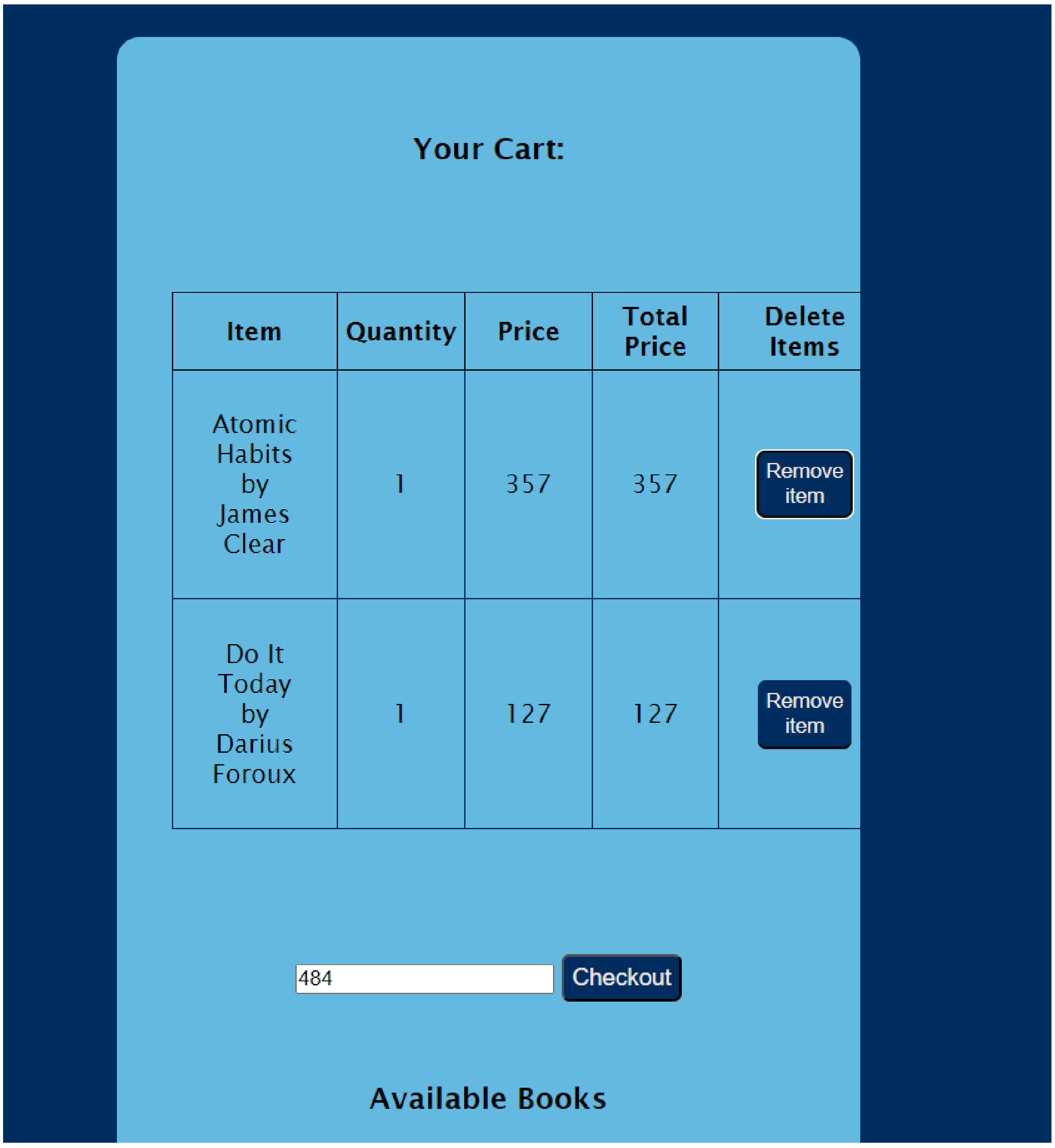

Your Cart:

Your cart is empty.

0 Checkout

Available Books

Book Title	Author	Genre	Price	
Atomic Habits	James Clear	Genre	Rs.357	Add to Cart
Do It Today	Darius Foroux	Genre	Rs.127	Add to Cart
Think Straight	Darius Foroux	Genre	Rs.194	Add to Cart
The Psychology Of Money	Morgan Housel	Genre	Rs.250	Add to Cart



Your Cart:

Item	Quantity	Price	Total Price	Delete Items
Atomic Habits by James Clear	1	357	357	Remove item
Do It Today by Darius Foroux	1	127	127	Remove item

484 Checkout

Available Books