Phase-4 Second Project WriteUp

**Team Budget Planner.**

DESCRIPTION

**Project objective:**

As a developer, write a program using JavaScript to decide the budget of a specific team.

**Background of the problem statement:**

As a developer, you are assigned to a project. You need to develop a website where program managers of a specific team will add details of professional deals they want to have with vendors. The finance team will check expenses of those teams and will decide their annual budget.

**You must use the following:**

● Visual Studio Code: An IDE to code for the application  
● JavaScript: A programming language  
● Git: To connect and push files from the local system to GitHub  
● GitHub: To store the application code and track its versions  
● JavaScript concepts: Functions, prototypes, primitives, objects, IIFEs, promises, async, and webpack

**Following requirements should be met:**

● Versions of the code should be tracked on GitHub repositories   
● Team Budget Planner should work properly

**index.html**

* Open Visual Studio Code
* *[Right click]* on the **src** folder of the project
* Select *New File* -> Enter the filename as **index.html**
* Write the code shown below resolving the warning and errors due compatibility-related issues

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Budget App</title>

    <link rel="stylesheet" href="style.css"/>

</head>

<body>

    <div class="container">

        <div class="header">

          <h1>Budget App</h1>

        </div>

        <div class="budget-section">

          <div class="budget col col-md col-sm">

            <h2>budget</h2>

            <img src="image/money-bag.svg" width="40" alt="" />

            <p class="amount">$ <span id="budgetAmount">0</span></p>

          </div>

          <div class="expenses col col-md col-sm">

            <h2>expenses</h2>

            <img src="image/accounting.svg" width="40" alt="" />

            <p class="exp-amount">$ <span id="expensesAmount">0</span></p>

          </div>

          <div class="balance col col-md col-sm">

            <h2>balance</h2>

            <img src="image/law.svg" width="40" alt="" />

            <p class="amount bala">$ <span id="balanceAmount">0</span></p>

          </div>

        </div>

        <div id="displayExpenses">

          <div class="budget-list">

            <div class="col">

              <h4>expense title</h4>

            </div>

            <div class="col">

              <h4>expense value</h4>

            </div>

            <div class="col"></div>

          </div>

          <div id="expValue"></div>

        </div>

      </div>

      <div class="toggle">

        <button id="myBtn" type="button">+</button>

      </div>

      <!-- The Modal -->

      <div id="myModal" class="modal">

        <!-- Modal content -->

        <div class="modal-content">

          <div class="modal-header">

            <span class="close">&times;</span>

            <h2>Budget Form</h2>

          </div>

          <div class="modal-body">

            <div class="budget-form" id="budgetform">

              <form id="addForm">

                <label for=""> Make a budget</label> <br />

                <input type="number" id="number" /> <br />

                <button type="submit">Add Budget</button>

              </form>

            </div>

            <div class="expense-form" id="expense-form">

              <form action="" id="expForm">

                <div class="">

                  <label for="">please enter your expense</label> <br />

                  <input type="text" id="expName" />

                </div>

                <div class="">

                  <label for="">please enter expense amount</label> <br />

                  <input type="number" id="expNumber" />

                </div>

                <button type="submit" id="submitExpen">Add expense</button>

            </form>

            <button onclick="callBudget()" id="bug">

              Back to add budget ->

            </button>

          </div>

          <div class="edit-form" id="editForm">

            <form action="" id="saveEdit">

              <div class="">

                <label for="">Edit your expense</label> <br />

                <input type="text" id="editExpName" />

              </div>

              <div class="">

                <label for="">Edit expense amount</label> <br />

                <input type="number" id="editExpNumber" />

              </div>

              <button type="submit">Save changes</button>

            </form>

          </div>

        </div>

      </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**style.css**

* *[Right click]* on the **src** folder of the project
* Select *New File* -> Enter the filename as **style.css**
* Execute the code shown below resolving the warning and errors due compatibility-related issues

@font-face {

    font-family: "General";

    src: url("./font/Nunito-ExtraLight.ttf");

  }

  @font-face {

    font-family: "heading";

    src: url("./font/Nunito-Bold.ttf");

  }

  body {

    margin: 0;

    padding: 0;

    width: 100%;

    font-family: "General";

    height: 100vh;

    background-color: #f5f5f5;

  }

  .container {

    min-height: 80vh;

  }

  .header {

    height: 60px;

    width: 100%;

  }

  h1 {

    margin: 0;

    text-align: center;

    font-family: "heading";

    padding-top: 10px;

  }

  .budget-section {

    display: flex;

    justify-content: space-around;

    align-items: flex-start;

    width: 100%;

    min-height: 160px;

    font-weight: 900;

  }

  .budget-list {

    display: flex;

    justify-content: space-around;

    align-items: flex-start;

    width: 100%;

    height: 70px;

    font-size: 20px;

  }

  .exp-amount {

    color: #b80c09;

  }

  .amount {

    color: #317b22;

  }

  .col {

    width: 40%;

    text-align: center;

    text-transform: capitalize;

  }

  .col p {

    font-size: 40px;

    padding-top: 5px;

    margin: 0;

  }

  .toggle {

    position: fixed;

    right: 0;

    bottom: 18px;

  }

  .toggle button {

    border: none;

    border-radius: 600px;

    background: #04458f;

    padding: 14px 20px;

    color: #ffffff;

    font-size: 30px;

    margin: 10px 50px 0 0;

    cursor: pointer;

  }

  /\* modal form \*/

  .budget-form {

    width: 100%;

    min-height: 200px;

}

form {

  margin: 10px 0 20px 10px;

  min-height: 100px;

}

input {

  width: 60%;

  padding: 8px 0;

  border-radius: 6px;

  border: 1px solid gray;

  margin: 20px 0;

  padding-left: 5px;

  color: #495057;

}

.budget-list .exp p {

  color: #b80c09;

  font-size: 18px;

}

#displayExpenses {

  width: 100%;

  display: none;

}

.expValue {

  display: flex;

  justify-content: space-around;

  height: 40px;

  font-weight: 900;

  text-align: center;

}

#expTitleName {

  width: 40%;

}

#expValueAmount {

  width: 40%;

}

#edite\_delete {

  width: 40%;

}

#edite\_delete img {

  cursor: pointer;

}

#edite\_delete button {

  margin-left: 1px;

  background: none;

  border: none;

}

.budget-form button {

  background: rgb(1, 83, 1);

  border: none;

  border-radius: 6px;

  color: #ffffff;

  padding: 10px 40px;

  cursor: pointer;

  text-transform: capitalize;

  font-size: 16px;

  font-weight: 400;

}

#bug {

  font-size: 14px;

  background: none;

  color: #04458f;

  margin: 0px 0px 15px 10px;

  text-align: justify;

  padding: 0;

}

#expense-form {

  width: 100%;

  display: none;

}

#editForm {

  display: none;

  width: 100%;

}

#editForm button {

  background: #b80c09;

  border: none;

  border-radius: 6px;

  color: #ffffff;

  padding: 10px 30px;

  cursor: pointer;

  text-transform: capitalize;

  font-size: 16px;

  font-weight: 400;

}

.expense-form button {

  background: #b80c09;

  border: none;

  border-radius: 6px;

  color: #ffffff;

  padding: 10px 30px;

  cursor: pointer;

  text-transform: capitalize;

  font-size: 16px;

  font-weight: 400;

}

/\* The Modal (background) \*/

.modal {

  display: block;

  position: fixed;

  z-index: 1;

  left: 0;

  top: 0;

  width: 100%;

  height: 100%;

  overflow: auto;

  background-color: rgba(0, 0, 0, 0.4);

  animation-name: fadeIn;

  animation-duration: 0.4s;

  text-transform: capitalize;

  font-weight: 600;

}

/\* Modal Content \*/

.modal-content {

  position: fixed;

  top: 20%;

  left: 25%;

  background-color: #fefefe;

  width: 50%;

  min-height: 40px;

  animation-name: slideIn;

  animation-duration: 0.4s;

}

/\* The Close Button \*/

.close {

  color: white;

  float: right;

  font-size: 28px;

  font-weight: bold;

  cursor: pointer;

  color: #000;

}

.modal-header {

  padding: 2px 16px;

  border-bottom: 2px solid gray;

}

.modal-body {

  padding: 2px 16px;

}

/\* Add Animation \*/

@keyframes slideIn {

  from {

    top: -300px;

    opacity: 0;

  }

  to {

    top: 0;

    opacity: 1;

  }

}

@keyframes fadeIn {

  from {

    opacity: 0;

  }

  to {

    opacity: 1;

  }

}

**script.js**

* *[Right click]* on the **src** folder of the project
* Select *New File* -> Enter the filename as **script.js**
* Execute the code shown below resolving the warning and errors due compatibility-related issues

const modal = document.getElementById("myModal");

const btn = document.getElementById("myBtn");

const span = document.getElementsByClassName("close")[0];

btn.onclick = function () {

  expName.value = "";

  expNumber.value = "";

  expenseForm.style.display = "block";

  editForm.style.display = "none";

  modal.style.display = "block";

};

span.onclick = function () {

  modal.style.display = "none";

};

window.onclick = function (event) {

  if (event.target == modal) {

    modal.style.display = "none";

  }

};

const amountInput = document.getElementById("number");

const addForm = document.getElementById("addForm");

const budgetAmount = document.getElementById("budgetAmount");

const balanceAmount = document.getElementById("balanceAmount");

const editForm = document.getElementById("editForm");

const saveEdit = document.getElementById("saveEdit");

const editExpValue = document.getElementById("editExpValue");

const editExpNumber = document.getElementById("editExpNumber");

const expForm = document.getElementById("expForm");

const expensesAmount = document.getElementById("expensesAmount");

const expValue = document.getElementById("expValue");

const displayExpenses = document.getElementById("displayExpenses");

const expenseForm = document.getElementById("expense-form");

const budgetform = document.getElementById("budgetform");

let expName = document.getElementById("expName");

let expNumber = document.getElementById("expNumber");

let id = 0;

let details = [];

function getBudgetAmount(amount) {

  if (!amount) {

    amountInput.style.border = "1px solid #b80c09";

    amountInput.placeholder = "input can not be empty";

    amountInput.style.color = "#b80c09";

    setTimeout(() => {

      amountInput.style.color = "#495057";

      amountInput.style.border = "1px solid gray";

    }, 3000);

  } else {

    budgetAmount.innerText = amount;

    balanceAmount.innerText = amount;

    expenseForm.style.display = "block";

    budgetform.style.display = "none";

    editForm.style.display = "none";

    amountInput.value = "";

  }

}

function addExpenses(name, number) {

  if (!name.length || !number.length) {

    expName.style.border = "1px solid #b80c09";

    expName.placeholder = "input can not be empty";

    expName.style.color = "#b80c09";

    expNumber.style.border = "1px solid #b80c09";

    expNumber.placeholder = "input can not be empty";

    expNumber.style.color = "#b80c09";

    setTimeout(() => {

      expName.style.color = "#495057";

      expName.style.border = "1px solid gray";

      expName.placeholder = "input can not be empty";

      expNumber.placeholder = "input can not be empty";

      expNumber.style.border = "1px solid gray";

      expNumber.style.color = "#495057";

    }, 3000);

  } else {

    const userExp = {

      id: id,

      name: name,

      number: parseInt(number),

    };

    details.push(userExp);

    displayExp(details);

    id++;

    expName.value = "";

    expNumber.value = "";

  }

}

function displayExp(details) {

  expValue.innerHTML = null;

  for (i = 0; i < details.length; i++) {

    expValue.innerHTML += `

    <div class="expValue" id="${details[i].id}">

      <div id="expTitleName" class="exp"><p>${details[i].name}</p></div>

      <div id="expValueAmount" class="exp"><p> <span>$ </span> ${details[i].number}</p></div>

      <div id="edite\_delete">

        <p>

          <button id="${details[i].id}" onclick="editExpDetails(${details[i].id})"> <img src="image/edit.svg" width="15" alt=""  /></button>

          <button id="${details[i].id}" onclick="delExpenseDetails(${details[i].id})"><img src="image/trash.svg" width="15" alt="" /></button>

        </p>

      </div>

    </div>

  `;

  }

  calcExpenses();

  displayExpenses.style.display = "block";

}

function calcExpenses() {

  let totalExp = 0;

  for (i = 0; i < details.length; i++) {

    totalExp = details[i].number + totalExp;

  }

  expensesAmount.innerText = totalExp;

  updateBalance();

}

function updateBalance() {

  balanceAmount.innerText =

    parseInt(budgetAmount.innerText) - parseInt(expensesAmount.innerText);

}

function delExpenseDetails(id) {

  let index = details.findIndex((item) => item.id === id);

  details.splice(index, 1);

  displayExp(details);

}

function editExpDetails(id) {

  expenseForm.style.display = "none";

  budgetform.style.display = "none";

  editForm.style.display = "block";

  details.findIndex((item) => {

    if (item.id === id) {

        editExpName.value = item.name;

      editExpNumber.value = item.number;

      saveEdit.children[2].id = item.id;

      modal.style.display = "block";

    }

  });

}

function getExpValue(editExpName, editExpNumber, id) {

    edited = details.findIndex((obj) => obj.id == id);

    details[edited].name = editExpName;

    details[edited].number = parseInt(editExpNumber);

    displayExp(details);

  }

  function callBudget() {

    budgetform.style.display = "block";

    expenseForm.style.display = "none";

  }

  saveEdit.addEventListener("submit", (e) => {

    e.preventDefault();

    getExpValue(editExpName.value, editExpNumber.value, saveEdit.children[2].id);

  });

  expForm.addEventListener("submit", (e) => {

    e.preventDefault();

    addExpenses(expName.value, expNumber.value);

  });

  addForm.addEventListener("submit", (e) => {

    e.preventDefault();

    getBudgetAmount(amountInput.value);

  });

What is Budget?

A budget is a financial plan for a defined period, often one year.

It may also include planned sales volumes and revenues, resource quantities, costs and expenses, assets, liabilities, and cash flows.

It may include a budget surplus, providing money for use at a future time, or a deficit in which expenses exceed income.

We have understood what budget is, now thinking of what a Budget App is?

We will say a budget app is an app that takes a budget amount and helps us to plan or breakdown our expenses from the budget amount.

So, we are going to build a Budget App that a user’s to add a budget amount, list down his expenses from the budget amount, get total balance after making list of expenses, edit an expense, and delete an expense.

Create a index.html file and write the code below:

    <div class="container">

        <div class="header">

          <h1>Budget App</h1>

        </div>

        <div class="budget-section">

          <div class="budget col col-md col-sm">

            <h2>budget</h2>

            <img src="image/money-bag.svg" width="40" alt="" />

            <p class="amount">$ <span id="budgetAmount">0</span></p>

          </div>

          <div class="expenses col col-md col-sm">

            <h2>expenses</h2>

            <img src="image/accounting.svg" width="40" alt="" />

            <p class="exp-amount">$ <span id="expensesAmount">0</span></p>

          </div>

          <div class="balance col col-md col-sm">

            <h2>balance</h2>

            <img src="image/law.svg" width="40" alt="" />

            <p class="amount bala">$ <span id="balanceAmount">0</span></p>

          </div>

        </div>

In the above code we create three div.

One is to display the budget amount, the other one is to display the balance amount after making a list of all expenses and the last is to display the amount of the total expenses.

let create a form that takes the budget amount.

            <div class="budget-form" id="budgetform">

              <form id="addForm">

                <label for=""> Make a budget</label> <br />

                <input type="number" id="number" /> <br />

                <button type="submit">Add Budget</button>

              </form>

            </div>

Add a budget amount

We are going to create a function that takes the budget amount and displays it after the user adds an amount.

In this project we will creating different functions that execute different tasks.

Before we proceed lets us know what is a function?

JavaScript a function allows you to define a block of code, give it a name and then execute it as many times as you want. A function can be defined using function keyword and can be executed using () operator. A function can include one or more parameters.

Knowing what a function is in JavaScript and what it does let move further to creating our first function.

const amountInput = document.getElementById("number");

const addForm = document.getElementById("addForm");

const budgetAmount = document.getElementById("budgetAmount");

const balanceAmount = document.getElementById("balanceAmount");

function getBudgetAmount(amount) {

if (!amount) {

amountInput.style.border = "1px solid #b80c09";

amountInput.placeholder = "input can not be empty";

amountInput.style.color = "#b80c09";

setTimeout(() => {

amountInput.style.color = "#495057";

amountInput.style.border = "1px solid gray";

}, 3000);

} else {

budgetAmount.innerText = amount;

balanceAmount.innerText = amount;

expenseForm.style.display = "block";

budgetform.style.display = "none";

editForm.style.display = "none";

amountInput.value = "";

}

}

addForm.addEventListener("submit", (e) => {

e.preventDefault();

getBudgetAmount(amountInput.value);

});

In the above code, we started by declaring a global variable amountInput and addForm. amountInput holds the budget amount that was added and addForm allows us to listen to the submit button inside our form when a user clicks the submit button after adding a budget amount and it also calls the getBudgetAmount() function.

The getBudgetAmount() take in an argument.

Note: Argument are values passed to a function when it is invoked.

we invoked or called the getBudgetAmount() function when we click on the submit button.

Inside the getBudgetAmount() function, we created a condition that checks if the user passed in an empty value or not.

If the condition is passed it takes the value and displays it.

We can see that we now have a budget amount and also a balanced amount. You might wonder why a balanced amount when we have not yet made any expenses. At this point, our budget amount is still our balance amount.

The expenses are 0 because we have not yet made any expenses yet.

Make an expense list

We are going to create another form that takes the expense name and expense amount.

<div class="expense-form" id="expense-form">

<form action="" id="expForm">

<div class="">

<label for="">please enter your expense</label> <br />

<input type="text" id="expName" />

</div>

<div class="">

<label for="">please enter expense amount</label> <br />

<input type="number" id="expNumber" />

</div>

<button type="submit" id="submitExpen">Add expense</button>

</form>

</div>

Let’s create a function that display list of expenses.

const expForm = document.getElementById("expForm");

let expName = document.getElementById("expName");

let expNumber = document.getElementById("expNumber");

let id = 0;

let details = [];

function addExpenses(name, number) {

if (!name.length || !number.length) {

expName.style.border = "1px solid #b80c09";

expName.placeholder = "input can not be empty";

expName.style.color = "#b80c09";

expNumber.style.border = "1px solid #b80c09";

expNumber.placeholder = "input can not be empty";

expNumber.style.color = "#b80c09";

setTimeout(() => {

expName.style.color = "#495057";

expName.style.border = "1px solid gray";

expName.placeholder = "input can not be empty";

expNumber.placeholder = "input can not be empty";

expNumber.style.border = "1px solid gray";

expNumber.style.color = "#495057";

}, 3000);

} else {

const userExp = {

id: id,

name: name,

number: parseInt(number),

};

details.push(userExp);

displayExp(details);

id++;

expName.value = "";

expNumber.value = "";

}

}

expForm.addEventListener("submit", (e) => {

e.preventDefault();

addExpenses(expName.value, expNumber.value);

});

In the above code, we started by declaring different variables that we will sign values to and also get values from.

The expNumber and expName variable hold the value of the expense amount and expense name input inside the expense from.

We declared an empty array that will push the list of expenses created by the user and also, we declared an id and set the value to 0.

Lastly expForm variable allows us to listen to the submit button inside our form when a user clicks the submit button after adding an expense list and it also calls the addExpenses() function.

The addExpenses() function takes two arguments, the expense name and the expense amount.

Inside the addExpense() function, we created a condition that checks if the user passed in an empty value or not.

If the condition is not passed it prompts an error to the user but if it passes it creates an object and signs the id we created to the id in the object, the expense name to the name in the object, and the expense amount to amount in the object. Note: we passed the amount as a number and the parseInt() method we used is a javascript method that converts a number that is passed as a string to a number. The Html 5 input type number sends a number as a string and we need to convert it to a number before we used it.

After that we push the object to the empty array we created and increase the Id we declared to 1 using Id++ and we invoke the displayExp() function that take the array as a parameter.

Now we are going to create the displayExp() function that displays the expense list after the user adds an expense. Note the function we created above only creates an expense.

function displayExp(details) {

expValue.innerHTML = null;

for (i = 0; i < details.length; i++) {

expValue.innerHTML += `

<div class="expValue" id="${details[i].id}">

<div id="expTitleName" class="exp"><p>${details[i].name}</p></div>

<div id="expValueAmount" class="exp"><p> <span>$ </span> ${details[i].number}</p></div>

<div id="edite\_delete">

<p>

<button id="${details[i].id}" onclick="editExpDetails(${details[i].id})"> <img src="image/edit.svg" width="15" alt="" /></button>

<button id="${details[i].id}" onclick="delExpenseDetails(${details[i].id})"><img src="image/trash.svg" width="15" alt="" /></button>

</p>

</div>

</div>

`;

}

calcExpenses();

displayExpenses.style.display = "block";

}

The displayExp() function we created takes the array as a parameter and inside the function, we loop through the array using for loop and using .innerHTML to append the object to the Html tag we created.

Calculate the total expense

We can now make an expense list and we need to know the total amount after making the list.

We are going to create a function that calculate the expenses and give us the total amount.

function calcExpenses() {

let totalExp = 0;

for (i = 0; i < details.length; i++) {

totalExp = details[i].number + totalExp;

}

expensesAmount.innerText = totalExp;

updateBalance();

}

In the function above we declared a let variable and assigned it a value of 0, then we loop through the array again to get the expense amount. After that, we reassign another value to the variable we created and the value is the expense amount we got from the array and we pulsed it with the variable again.

We invoked the calcExpenses() function inside the dispalyExp() function and we invoke the updateBalance() function.

Get total balance after adding expenses

We are going to create a function that minus the total expense from the budget amount and updates the total balance fields as the user makes a list of expenses.

function updateBalance() {

balanceAmount.innerText =

parseInt(budgetAmount.innerText) - parseInt(expensesAmount.innerText);

}

The function above gets the total expense amount and minus it from the budget amount.

If we load our app now and add a budget amount and make a list of expenses, we will see the balance and total expense field updating as we add an expense.

Edit an expense

Let make our app more user friendly by giving our users access to be able to edit an expense after adding or making list of expenses.

First, we are going to create another form.

<div class="edit-form" id="editForm">

<form action="" id="saveEdit">

<div class="">

<label for="">Edit your expense</label> <br />

<input type="text" id="editExpName" />

</div>

<div class="">

<label for="">Edit expense amount</label> <br />

<input type="number" id="editExpNumber" />

</div>

<button type="submit">Save changes</button>

</form>

</div>

Now let’s create a edit function:

function editExpDetails(id) {

expenseForm.style.display = "none";

budgetform.style.display = "none";

editForm.style.display = "We block";

details.findIndex((item) => {

if (item.id === id) {

editExpName.value = item.name;

editExpNumber.value = item.number;

saveEdit.children[2].id = item.id;

modal.style.display = "block";

}

});

}

Remember that each expense has an id and with that id we can find the index of that object and we can modify the object as we want.

The function above takes the id of each object as a parameter and loop through the array of object and get the index of that object sets the values back to the input field and bind the id of that object to the form id.

Note the id is passed when a user clicks on the edit button.

const editForm = document.getElementById("editForm");

const saveEdit = document.getElementById("saveEdit");

const editExpValue = document.getElementById("editExpValue");

const editExpNumber = document.getElementById("editExpNumber");

function getExpValue(editExpName, editExpNumber, id) {

edited = details.findIndex((obj) => obj.id == id);

details[edited].name = editExpName;

details[edited].number = parseInt(editExpNumber);

displayExp(details);

}

saveEdit.addEventListener("submit", (e) => {

e.preventDefault();

getExpValue(editExpName.value, editExpNumber.value, saveEdit.children[2].id);

});

The function above gets the expense name and expense amount from the edit form and also the id. It takes the id that was assigned to the form and find the index of that id and changes the value of that index to the new values that were typed by a user or still saves the value again if the user decided to leave the value.

Delete an expense

Let also give user access to delete an expense if user decide not to edit but to delete.

function delExpenseDetails(id) {

let index = details.findIndex((item) => item.id === id);

details.splice(index, 1);

displayExp(details);

}

Again, we going to use an id.

The function above takes an id and find the index of that id and delete the object from the list of objects inside the array of object.

If we load our app on the browser now and try to edit or delete an expense, it will delete and it will also edit.

**Executing the code:**

Execute the code by clicking run button in the index.html and the output is shown in a browser.

**Conclusion**

Team Budget Planner is working properly.

**Pushing the code to your GitHub repositories:**

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**