TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS (502070)

LAB1 – HTML/JS/AJAX/LOCAL/SESSION STORAGE

TDTU

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS
Lab1 – HTML/JavaScript

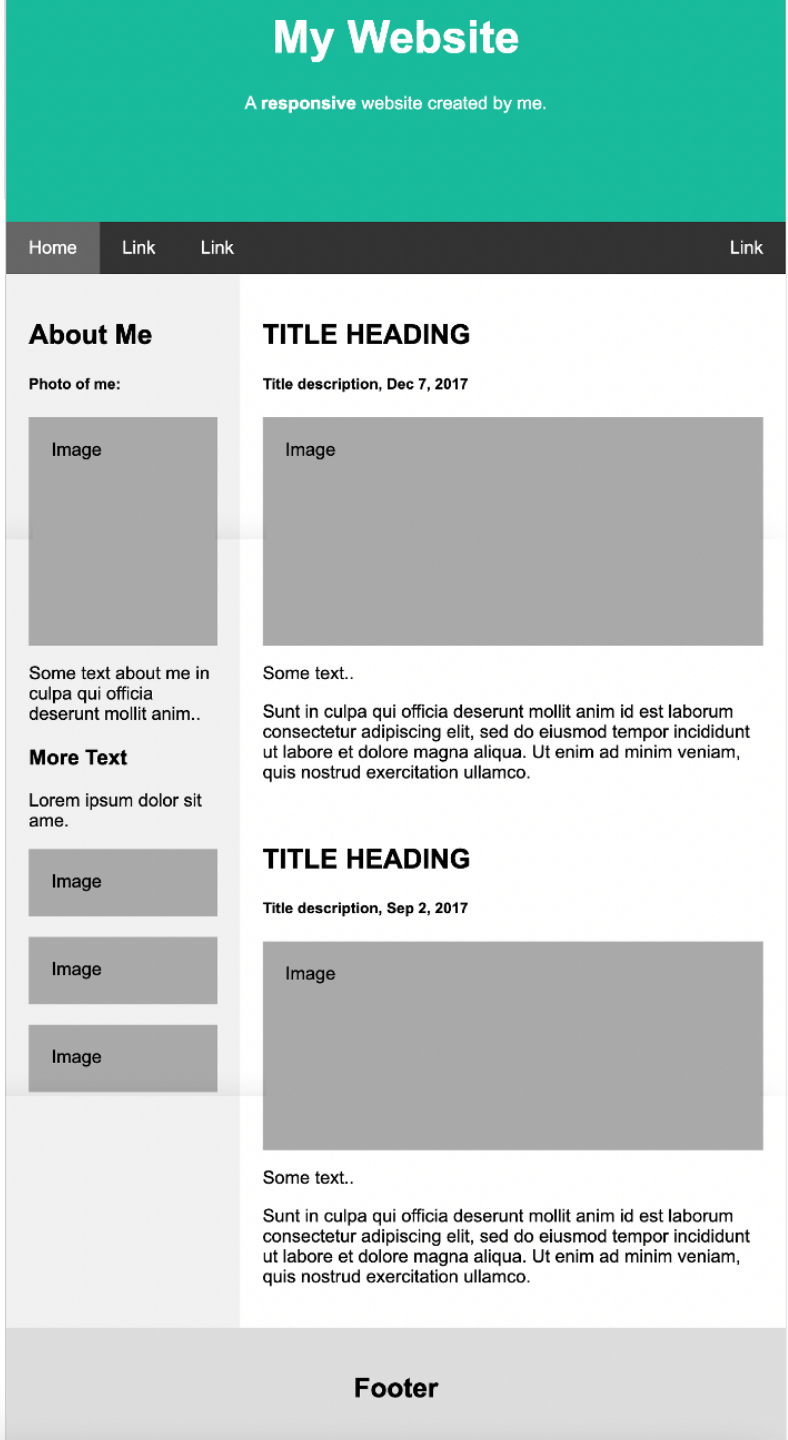
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page Title</title>
<meta charset="UTF-8">
</head>
<body>

<div class="header">
  <h1>My Website</h1>
  <p>A <b>responsive</b> website created by me.</p>
</div>

<div class="navbar">
  <a href="#" class="active">Home</a>
  <a href="#" class="right">Link</a>
</div>

<div class="row">
  <div class="side">
    <h2>About Me</h2>
    <h5>Photo of me:</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text about me in culpa qui officia deserunt mollit anim..</p>
  </div>
</div>

<div class="footer">
  <h2>Footer</h2>
</div>
</body>
</html>
```



Review HTML/JavaScript

JavaScript HTML DOM (w3schools.com)

JavaScript RegExp Reference (w3schools.com)

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

```
<script src="https://www.w3schools.com/js/myScript.js"></script>
```

Review Ajax

AJAX Introduction (w3schools.com)

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Review Ajax

AJAX Introduction (w3schools.com)

BLOB ResponseType

XMLHttpRequest.responseType - Web APIs | MDN (mozilla.org)

Download File as BLOB using AJAX and jQuery (aspsnippets.com)

URL.createObjectURL() - Web APIs | MDN (mozilla.org)

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
    if (xhr.readyState == 2) {
        if (xhr.status == 200) {
            xhr.responseType = 'blob'
        } else {
            xhr.responseType = 'text'
        }
    }
};
return xhr;
```

Lab1 - 2

[jQuery.ajax() | jQuery API Documentation]

```
var request = $.ajax({
  url: "script.php",
  method: "POST",
  data: { id : menuId },
  dataType: "html"
});

request.done(function( msg ) {
  $( "#log" ).html( msg );
});

request.fail(function( jqXHR, textStatus ) {
  alert( "Request failed: " + textStatus );
});
```

[Using the Fetch API - Web APIs | MDN (mozilla.org)]

```
fetch(url)
  .then(function() {
    // handle the response
  })
  .catch(function() {
    // handle the error
  });
```


```
{
  "code": 0,
  "message": "Tải danh sách thành công",
  "data": [
    {
      "id": 1,
      "name": "Đan Trường",
      "age": 35
    },
    {
      "id": 2,
      "name": "Cẩm Ly",
      "age": 32
    },
    {
      "id": 3,
      "name": "Sơn Tùng - MTP",
      "age": 20
    },
    {
      "id": 4,
      "name": "Lý Hải",
      "age": 25
    },
    {
      "id": 5,
      "name": "Lệ Quyên",
      "age": 40
    }
  ]
}
```

Lab1 – 3 - Promise

[JavaScript Promises (w3schools.com)]

[Promise - JavaScript | MDN (mozilla.org)]

Promise Syntax

```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Lab1 – 3 – Async/await

JavaScript Async (w3schools.com)

Making asynchronous programming easier with async and await - Learn web development | MDN (mozilla.org)

```
async function makeResult(items) {
  let newArr = [];
  for(let i = 0; i < items.length; i++) {
    newArr.push('word_' + i);
  }
  return newArr;
}

async function getResult() {
  let result = await makeResult(items); // Blocked on this line
  useThatResult(result); // Will not be executed before makeResult() is done
}
```

Lab1 – 3 – Async/await vs Promise

[JavaScript: Promises or async-await | by Gokul N K | Better Programming)]

Sr.no	Promise	Async/Await
1.	Promise is an object representing intermediate state of operation which is guaranteed to complete its execution at some point in future.	Async/Await is a syntactic sugar for promises, a wrapper making the code execute more synchronously.
2.	Promise has 3 states – resolved, rejected and pending.	It does not have any states. It returns a promise either resolved or rejected.
3.	If the function "fxn1" is to executed after the promise, then promise.then(fxn1) continues execution of the current function after adding the fxn1 call to the callback chain.	If the function "fxn1" is to executed after await, then await X() suspends execution of the current function and then fxn1 is executed.
4.	Error handling is done using .then() and .catch() methods.	Error handling is done using .try() and .catch() methods.
5.	Promise chains can become difficult to understand sometimes.	Using Async/Await makes it easier to read and understand the flow of the program as compared to promise chains.

Lab 1 - 4

HTML Web Storage API (w3schools.com)

The localStorage Object

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Example

```
// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

The sessionStorage Object

The `sessionStorage` object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

Thank you