Check the JSON Response Object Labs

Perform these labs on your own computer using VS Code or any editor and command prompt to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Return JSON With a Body

Open the **Index.js** in the Web API Server project and add a new route.

```
// Return status code with a body
router.get('/responseWithBody', function (req, res,
next) {
  res.status(200).json({
    "status": 200,
    "statusText": "OK",
    "message": "Success.",
    "data": { "message": "Success" }
  });
});
```

Open the **index.html** file and add a new below the last .

```
<div>
  <button onclick="responseWithBody();">
    Return a Body
  </button>
</div>
```

Add a new function to call the new route you just added.

```
function responseWithBody() {
  // If no body is returned, the body property is empty
  fetch(`${SERVER_URI}/responseWithBody`)
    .then(response => {
      console.log(response);
      return response.json();
    })
    .then(response => {
      console.log(response);
      console.log(response.data);
    })
    .catch(error => console.error(error));
}
```

Try It Out

Run the application and click on the **Return a Body** button.

Lab 2: Return JSON Without a Body

Open the **Index.js** in the Web API Server project and add a new route.

```
// Return status code without a body
router.get('/responseWithoutBody', function (req, res,
next) {
  res.status(200).send();
});
```

Open the **index.html** file and add a new button within the last you just added.

```
<div>
  <button onclick="responseWithBody();">
    Return a Body
  </button>
  <button onclick="responseWithoutBody();">
    No Body Returned
  </button>
</div>
```

Add a new function to call the new route you just added.

```
function responseWithoutBody() {
  // If no body is returned, the response object is a
JSON object
  fetch(`${SERVER_URI}/responseWithoutBody`)
    .then(response => response)
    .then(response => {
      console.log(response);
      console.log(response.status);
    })
    .catch(error => console.error(error));
}
```

Try It Out

Run the application and click on the **No Body Returned** button.

Lab 3: Return Text Only

Open the **Index.js** in the Web API Server project and add a new route.

```
// Return status code with text only
router.get('/responseWithText', function (req, res,
next) {
  res.status(200).send("Status ID: 200");
});
```

Open the **index.html** file and add a new button within the last you just added.

```
<div>
  <button onclick="responseWithBody();">
    Return a Body
  </button>
  <button onclick="responseWithoutBody();">
    No Body Returned
  </button>
  <button onclick="responseWithText();">
    Return Text Only
  </button>
</div>
```

Add a new function to call the new route you just added.

```
function responseWithText() {
  // Return text only
  fetch(`${SERVER_URI}/responseWithText`)
    .then(response => {
      console.log(response);
      return response.text();
    })
    .then(response => {
      console.log(response);
    })
    .catch(error => console.error(error));
}
```

Try It Out

Run the application and click on the **Return Text Only** button.

Lab 4: Create a Generic Approach to Process Response Object

Open the **index.html** file and add a new button within the last you just added.

```
<div>
  <button onclick="responseWithBody();">
    Return a Body
  </button>
  <button onclick="responseWithoutBody();">
    No Body Returned
  </button>
  <button onclick="responseWithText();">
    Return Text Only
  </button>

  <button onclick="responseGeneric();">
    Process Response Object
  </button>
</div>
```

Add a new function to generically process the response object.

```
async function processResponseObject(response) {
  let text = "";
  let data = {};
  // Create custom response object
  let ret = {
    "ok": response.ok,
    "status": response.status,
    "statusText": response.statusText,
    "url": response.url
  };

  try {
    // Get text first
    text = await response.text();
    // Attempt to convert to JSON
    data = JSON.parse(text);
    // Return JSON data
    ret.data = data.data;
  }
  catch {
    // Return the text
    ret.data = text;
  }

  return ret;
}
```

Add the responseGeneric() function that can be called from the button you created and calls the processResponseObject() function.

```
function responseGeneric() {
  let uri = `${SERVER_URI}/responseWithBody`;
  //let uri = `${SERVER_URI}/responseWithoutBody`;
  //let uri = `${SERVER_URI}/responseWithText`;

  fetch(uri)
    .then(response => processResponseObject(response))
    .then(response => console.log(response))
    .catch(error => console.error(error));
}
```

Try It Out

Save the changes, reload the web page, and click on the **Process Response Object** button. You should see the same result as you saw when you clicked on the **Return a Body** button.

Comment out the **uri** that calls the **responseWithBody** endpoint.

Uncomment the **uri** that calls the **responseWithoutBody** endpoint.

Save your change and reload the web page.

Click on the **Process Response Object** button. You should see the same result as you saw when you clicked on the **No Body Returned** button.

Comment out the **uri** that calls the **responseWithoutBody** endpoint.

Uncomment the **uri** that calls the **responseWithText** endpoint.

Save your change and reload the web page.

Click on the **Process Response Object** button. You should see the same result as you saw when you clicked on the **Return Text Only** button.