

Diploma Web Application Development: Introduction

ICT50220 Diploma of Information Technology(Front-End Web Development)

Code	Title
ICTWEB517	Create web-based programs
ICTWEB546	Validate application design against specifications

Web Application Development & HTTP

Start by pulling down the latest class code from GitHub

Activity 1 – Set up Postman

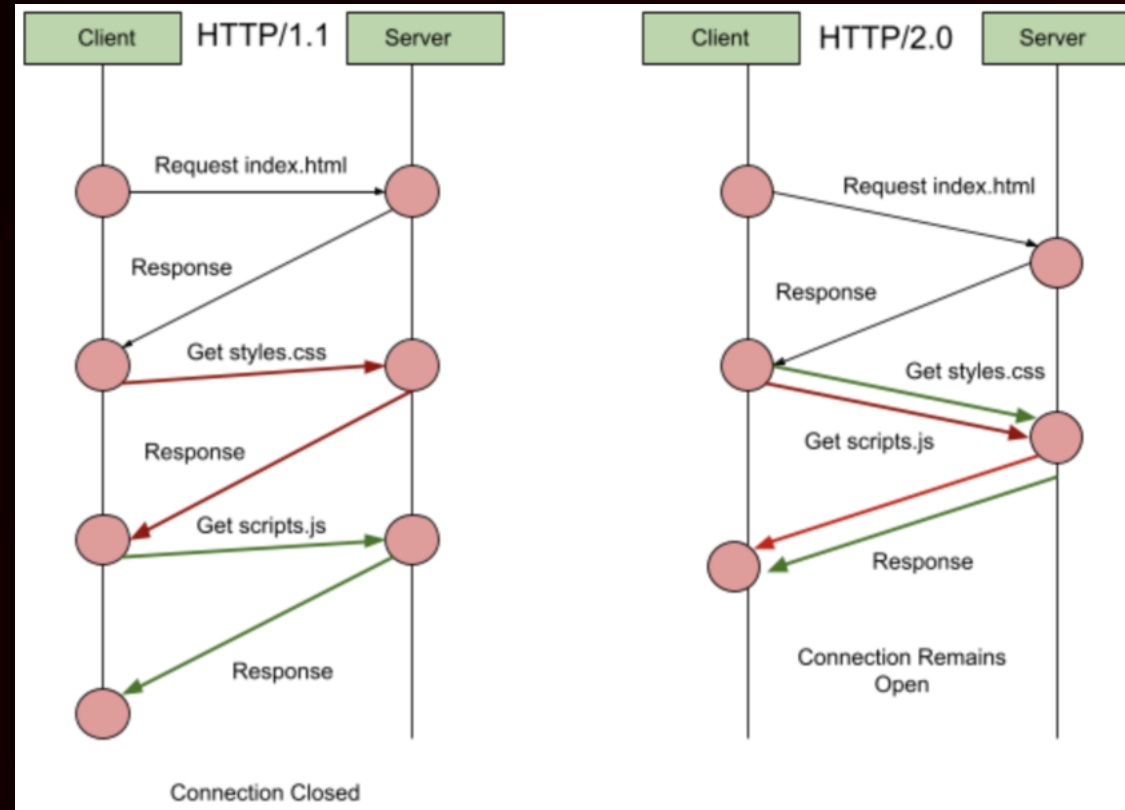
The NASA API

1. Navigate to the NASA API Portal:
2. Go to the [NASA API Portal](#) in a web browser.
3. Register for an API Key:
4. Click on the "Get Started" or "Sign Up" button.
5. Fill in the required details (name, email, etc.).
6. Agree to the terms and conditions.
7. Submit the registration form.
8. Check your email for the API key and confirmation.

What is HTTP

- HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the web.
- HTTP is a stateless protocol, meaning each request is independent.

HTTP 1 vs 2



<https://code.tutsplus.com/http-the-protocol-every-web-developer-must-know-part-1--net-31177t>

URI/URL

- Uniform Resource Identifier: A generic term for all types of names and addresses.
- Uniform Resource Locator: A specific type of URI that includes location information.
- Components of a URL: scheme, host, path, query & fragment.

Making a GET HTTP Request to API

- Open/Download Postman
- Inspect the NASA API Docs

<https://api.nasa.gov/#browseAPI>

- Add your NASA api key to the following URL query

https://api.nasa.gov/planetary/apod?api_key=API_KEY

- Paste the URL into a **GET** request in Postman

Calling the API

The screenshot shows a REST client interface with the following details:

- Environment:** GET API Key, GET api_call_basic_auth, No Environment
- URL:** `https://api.nasa.gov/planetary/apod?api_key=[REDACTED]`
- Method:** GET
- Status:** 200 OK, Time: 372 ms, Size: 1.76 KB
- Response Body (JSON):**

```
1 {
2   "copyright": "\nDonato Lioce; \nText: Natalia Lewandowska \n(SUNY Oswego)\n",
3   "date": "2024-08-07",
4   "explanation": "To some, they look like battlements, here protecting us against the center of the Milky Way. The Three Merlons,
                    also called the Three Peaks of Lavaredo, stand tall today because they are made of dense dolomite rock which has better
                    resisted erosion than surrounding softer rock. They formed about 250 million years ago and so are comparable in age with one
                    of the great extinctions of life on Earth. A leading hypothesis is that this great extinction was triggered by an asteroid
                    about 10-km across, larger in size than Mount Everest, impacting the Earth. Humans have gazed up at the stars in the Milky Way
                    and beyond for centuries, making these battlefield-like formations, based in the Sexten Dolomites, a popular place for current
                    and ancient astronomers.",
5   "hdurl": "https://apod.nasa.gov/apod/image/2408/DolomitesSky_Lioce_4681.jpg",
6   "media_type": "image",
7   "service_version": "v1",
8   "title": "Milky Way Behind Three Merlons",
9   "url": "https://apod.nasa.gov/apod/image/2408/DolomitesSky_Lioce_960.jpg"
10 }
```

Constructing the URL

Using the API Docs add the following parameters:

- start_date
- end_date
- count
- date

Consider the 2-dimensional nature of the data you are receiving in the response.

Adding parameters

HTTP nasa_api / api_call_with_parameters

GET ▼ https://api.nasa.gov/planetary/apod?start_date=2022-10-01&end_date=2023-10-01

Params ● Authorization ● Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key		Value	Description
<input checked="" type="checkbox"/>	api_key	ⓘ	GurS0wJyr12na3jh...	
<input checked="" type="checkbox"/>	start_date		2022-10-01	
<input checked="" type="checkbox"/>	end_date		2023-10-01	
<input type="checkbox"/>	date		2024-08-01	
<input type="checkbox"/>	count		5	

Headers

- Headers provide metadata about the **request** or **response**.
- This can include:
 - server settings,
 - information on usage
 - cached/stored data to make content delivery faster
 - custom headers to transmit application specific information

Common HTTP Headers

- Content-Type : type of the resource or the data being sent to the
 - *Content-Type: application*
- Authorization: contains the credentials to authenticate a user (e.g. API Key)
 - *Authorization: Bearer <token>*
- Accept: Lets the server know what types of data the client can process.
 - *Accept: application/json*
- Cache-Control: caching information in requests and response for prompt delivery
 - *Cache-Control: no-cache*
- Referer: Contains the address of the previous web page
 - *Referer: https://example.com*



















Request & Responses

Let's look at two requests in Postman:

1. <https://api.nasa.gov/planetary/apod>
2. <https://blackboard.northmetrotafe.wa.edu.au/>

Review the headers using Postman **i** icon

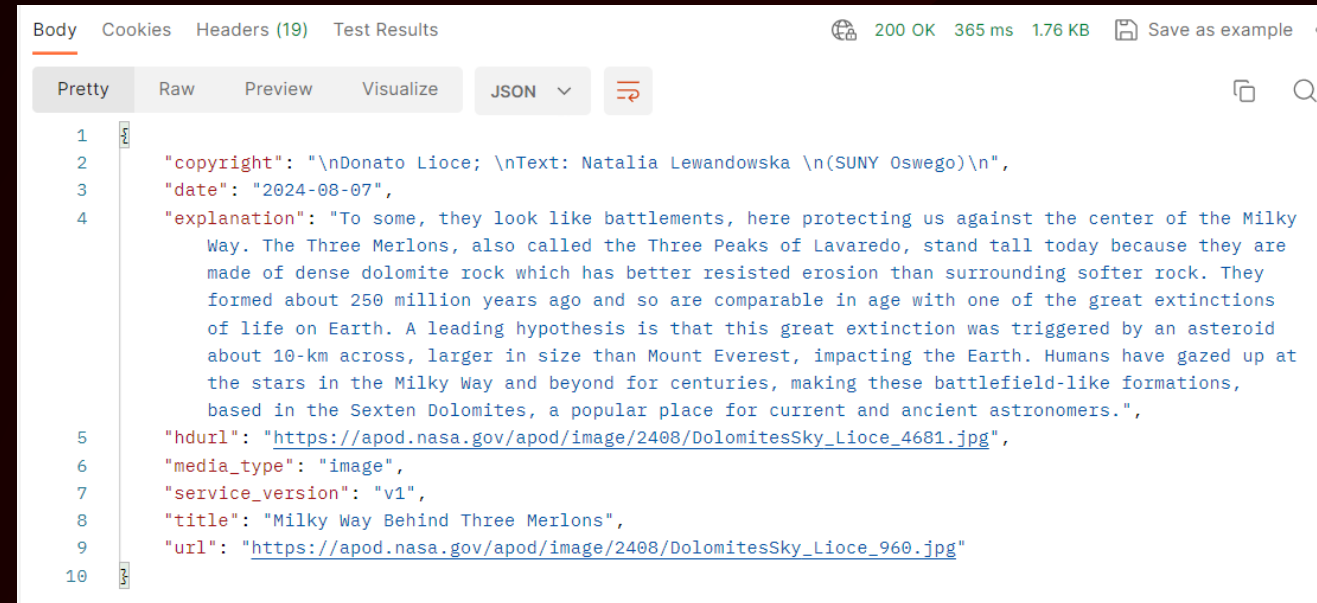
Can you see the cookies saved in your browser's local storage?

Body	Cookies (4)	Headers (29)	Test Results														
		<table><tr><th>Key</th><th></th></tr><tr><td>Access-Control-Allow-Origin</td><td></td></tr><tr><td>Cache-Control</td><td></td></tr><tr><td>Cache-Control</td><td></td></tr><tr><td>Cache-Control</td><td></td></tr><tr><td>Cache-Control</td><td></td></tr><tr><td>Cache-control</td><td></td></tr></table>	Key		Access-Control-Allow-Origin		Cache-Control		Cache-Control		Cache-Control		Cache-Control		Cache-control		
Key																	
Access-Control-Allow-Origin																	
Cache-Control																	
Cache-Control																	
Cache-Control																	
Cache-Control																	
Cache-control																	

Status Codes

Responses provide a:

- status line with the code
- headers as we discussed
- body containing your data/message



The screenshot shows the 'Body' tab of a web browser's developer tools. The response is a JSON object with the following fields:

```
1 {  
2   "copyright": "\nDonato Lioce; \nText: Natalia Lewandowska \n(SUNY Oswego)\n",  
3   "date": "2024-08-07",  
4   "explanation": "To some, they look like battlements, here protecting us against the center of the Milky  
Way. The Three Merlons, also called the Three Peaks of Lavaredo, stand tall today because they are  
made of dense dolomite rock which has better resisted erosion than surrounding softer rock. They  
formed about 250 million years ago and so are comparable in age with one of the great extinctions  
of life on Earth. A leading hypothesis is that this great extinction was triggered by an asteroid  
about 10-km across, larger in size than Mount Everest, impacting the Earth. Humans have gazed up at  
the stars in the Milky Way and beyond for centuries, making these battlefield-like formations,  
based in the Sexten Dolomites, a popular place for current and ancient astronomers.",  
5   "hdurl": "https://apod.nasa.gov/apod/image/2408/DolomitesSky_Lioce_4681.jpg",  
6   "media_type": "image",  
7   "service_version": "v1",  
8   "title": "Milky Way Behind Three Merlons",  
9   "url": "https://apod.nasa.gov/apod/image/2408/DolomitesSky_Lioce_960.jpg"  
10 }
```

Status Codes Ranges

Status codes indicate the result of the request.

- 1xx - Informational only
- 2xx - Successful response
- 3xx - Redirection to another URI/URL
- 4xx - Client Errors - front end errors
- 5xx - Server Errors - backend errors

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

Request & Methods

There are several different methods however the most common relate to CRUD activities

- **GET:** *Retrieve data.*
- **POST:** *Send data to the server.*
- **PUT:** *Update existing data.*
- **DELETE:** *Remove data.*

This how we manipulate data in databases

CREATE

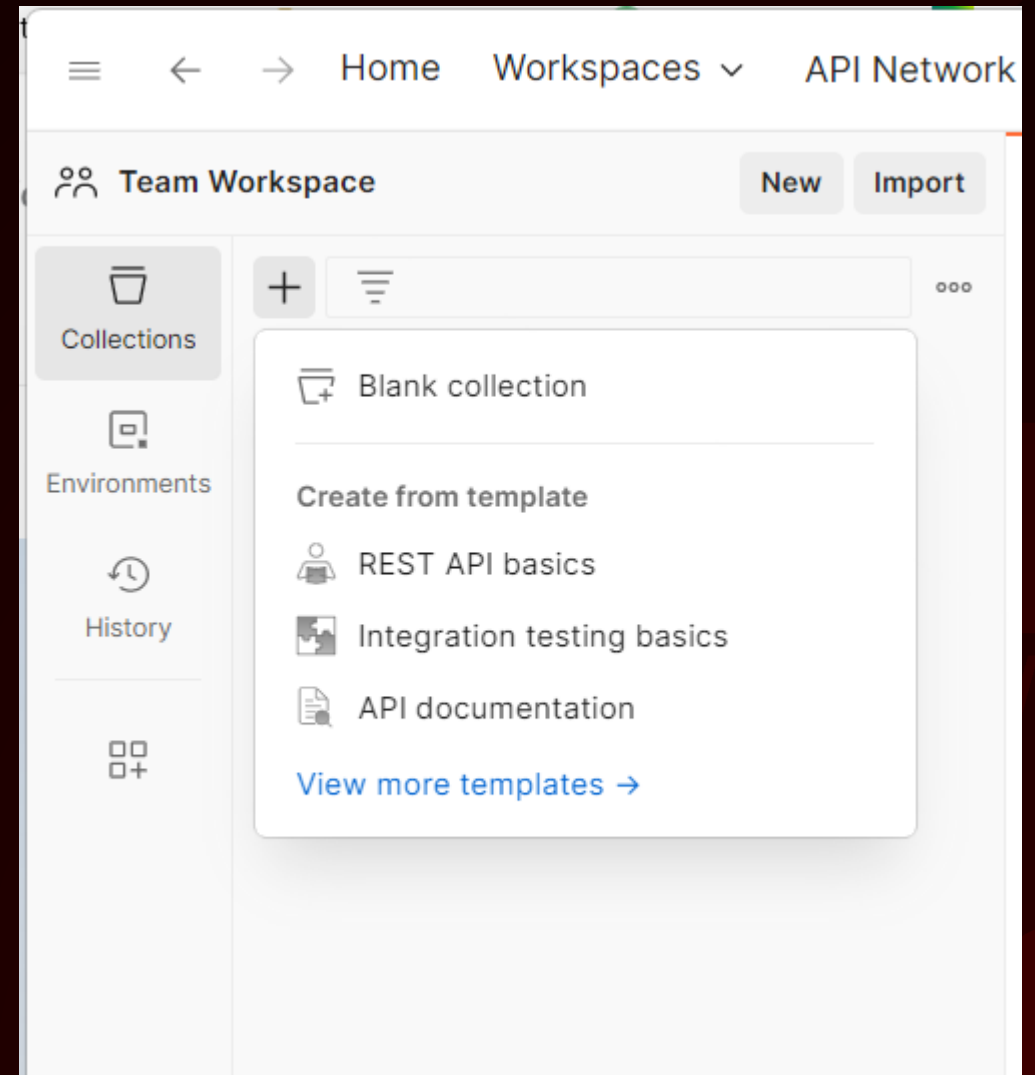
READ

UPDATE

DELETE

Testing HTTP Methods

- Open Postman Collections.
- Find and select the **REST API** basic template
- Open the collection in Postman
- Try the **GET** request
- Place the `user_object.json` in the body of the **POST** request
- Notice how the server returned the successful request information



Testing HTTP Status Codes

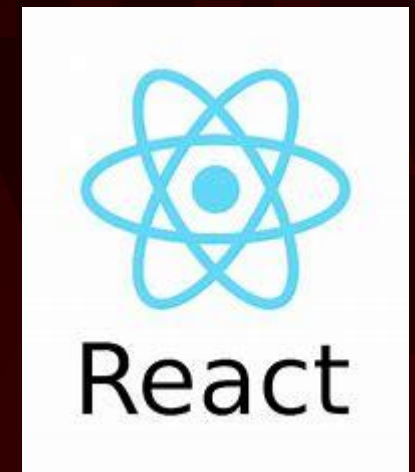
- Add the count parameter when searching dates with the API
- Add an incorrect API key
- Make a call to a route that doesn't exist
- Add an incorrectly form JSON to POST or PUT requests in the REST API Basic collection
- Remove some/all headers from a request

Decoupled Web Applications

- Well-designed APIs include endpoints(routes to the data)
- Secure Authentication methods like OAuth, API keys are used to protect API endpoints.
- Proper handling of HTTP response statuses and errors for the user.
- Caching strategies to reduce server load

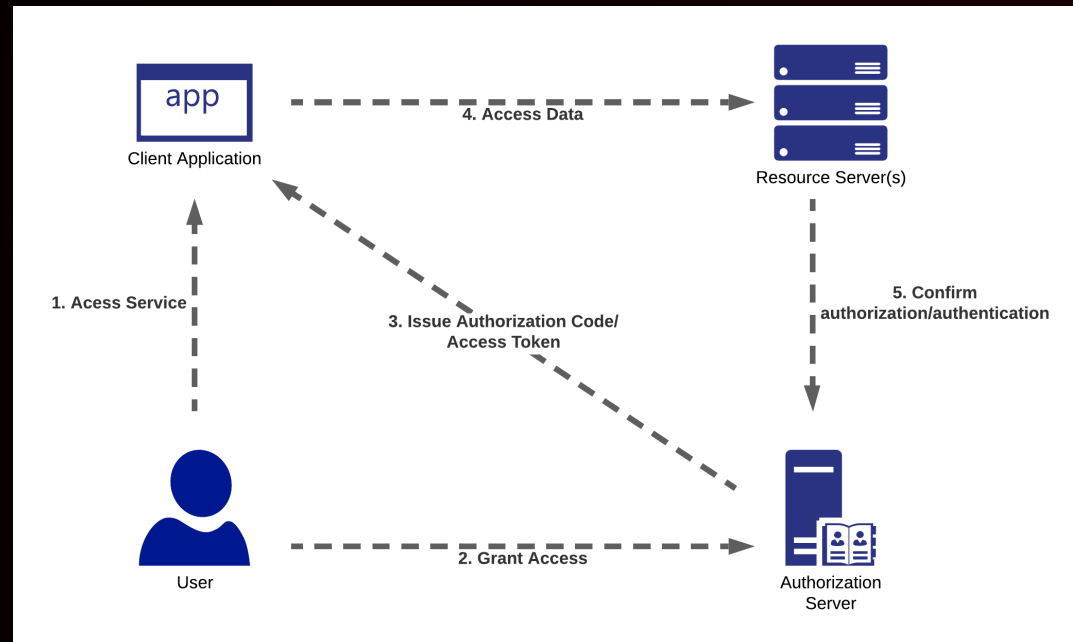
// Our API endpoint

<https://api.nasa.gov/planetary/apod>

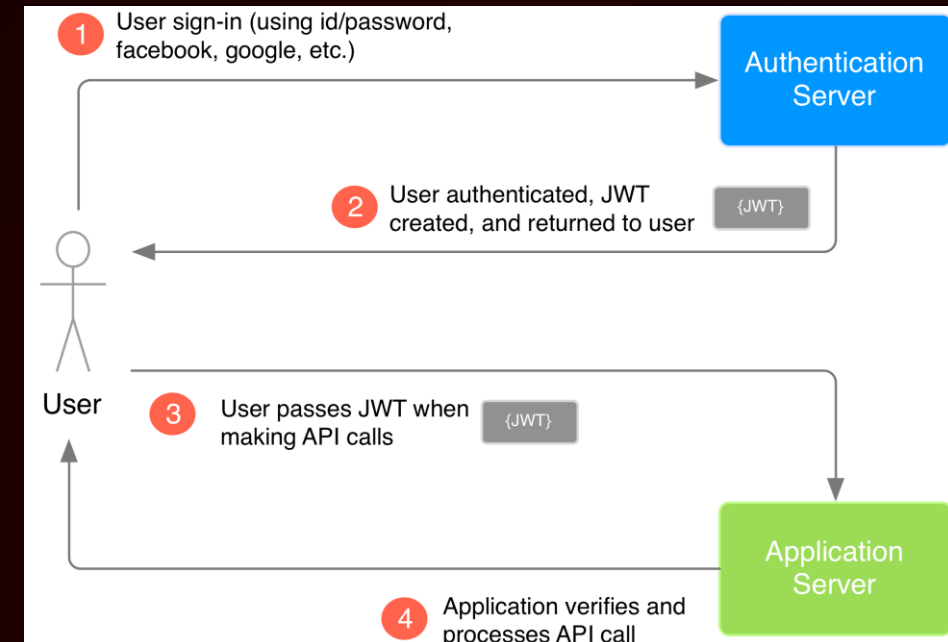


Authentication in Web Applications

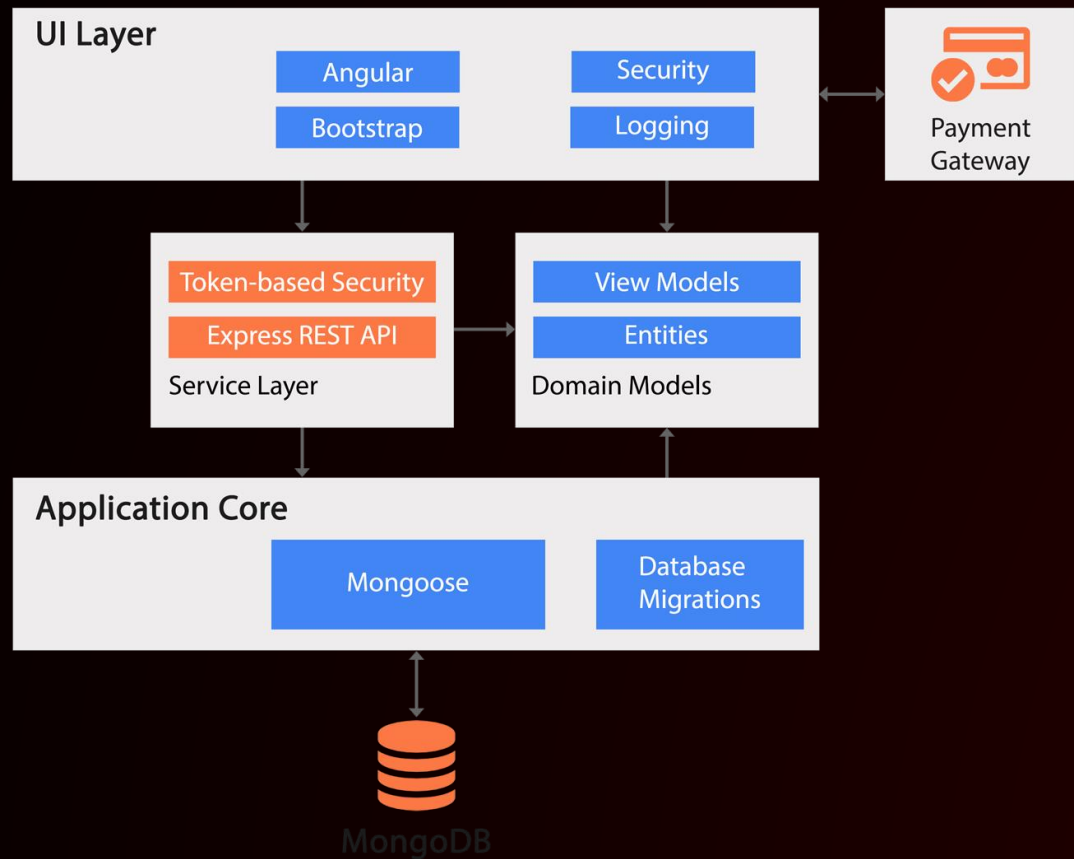
O-Auth 2.0



JWT



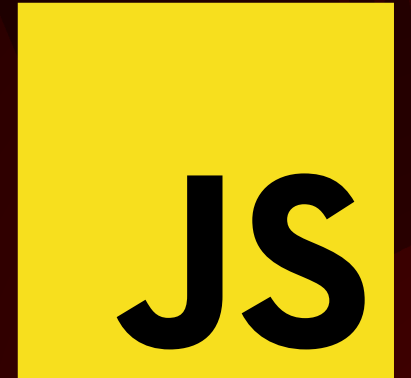
HTTP in Web Applications



Our front end code will need to:

- authenticate using HTTP
- handle errors using HTTP codes/response bodies
- handle all status codes gracefully
- be able to deal with changing complex objects
- effectively cache(save) information to provide better user experience

Challenges



Challenge 1

1. check all the form fields to see which fields have data
2. add them to the apiURL as parameters
3. test the responses in the Network tab of your browser so they all have a status of 200

Hint: use concatenation

Challenge 2

1. add the following headers to your API call content type, user agent & cache control
2. set the cache control no cache
3. set the content type application/json
4. Test the responses in the Network tab of your browser so they all have a status of 200

Challenge 3

1. change the anonymous arrow function below to check if the response code is 200(ok)
2. if the response is ok return the response.json() object
3. if not ok throw a new error which includes the status code
4. Test the responses in the Network tab

Challenge 4

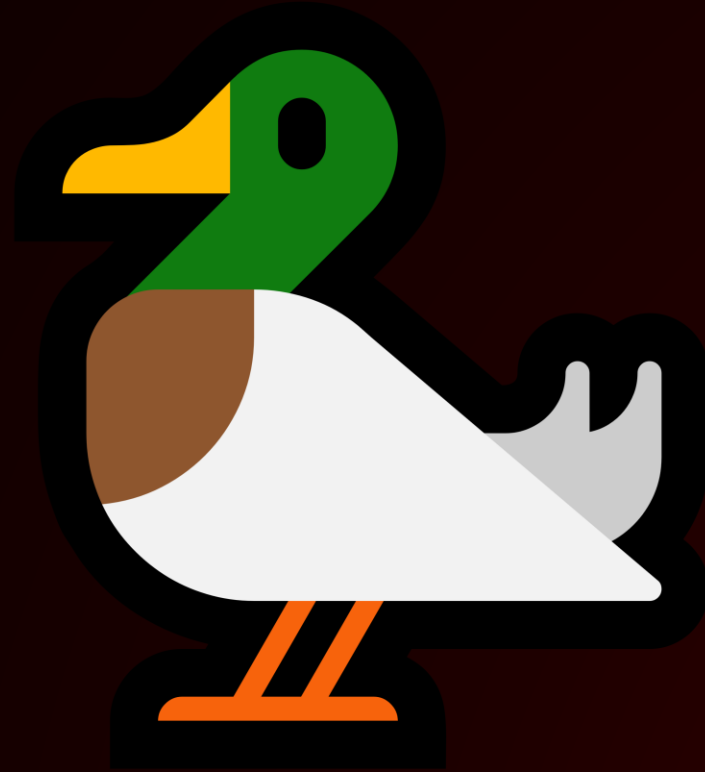
1.The code insert the data into the innerHTML but the code is repeated

2.Move the if/else if/else into a re-usable function ready for modularisation

Work on Assessment 1



JS



TAFE
WA