



NYP LIT

Chatbot with Javascript Workshop

Creating a fully functional chatbot using Javascript

9/10/23



all copyrights reserved for LIT CLUB





Agenda

Getting started with DOM structures

Brief introduction to Javascript

Introduction to APIs, Nodejs and React

Quiz Time



Pre-Workshop Requisites

- PyCharm or Visual Studio Code installed
- Postman installed with account
- RapidAPI account
- Knowledge of basic CSS and HTML



[Back to Agenda Page](#)



Getting started with DOM structures!



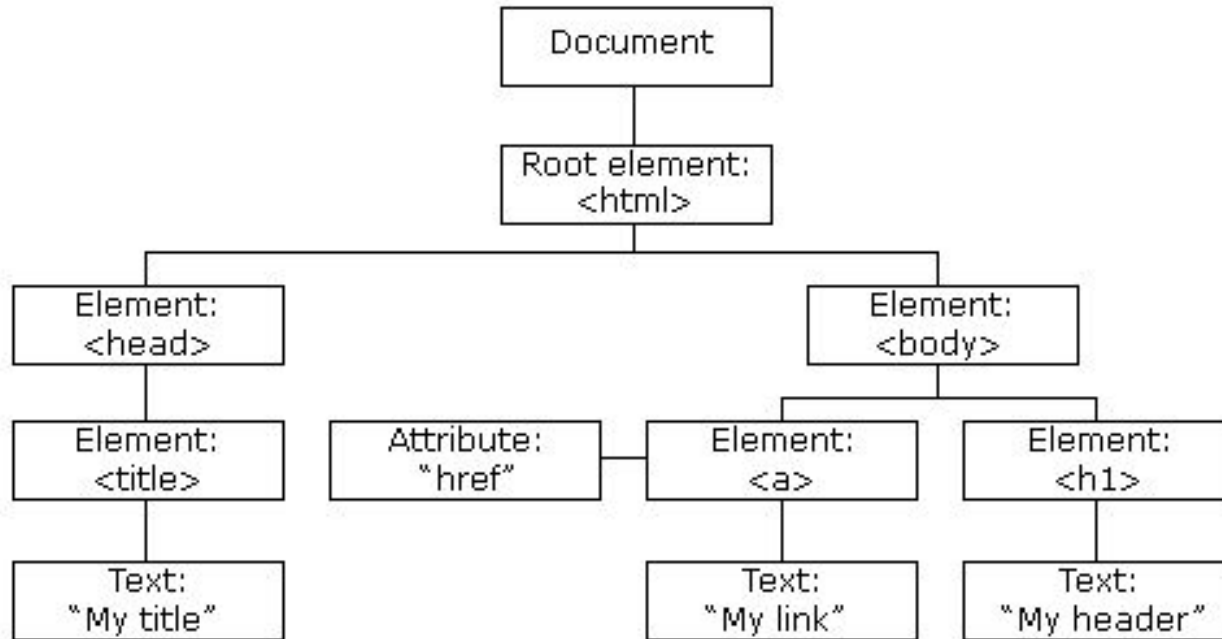


DOM Structures

The **Document Object Model (DOM)** is a programming interface for web documents.

- **Hierarchy:** The DOM forms a hierarchical structure:
- **Root Node:** Represents the entire document (<html>).
- **Elements:** Represent HTML tags (e.g., <div>, <p>).
- **Nodes:** Include elements, attributes, text, comments, etc.
- **Relationships:**
 - *Parent-Child:* Elements have parent and child relationships.
 - *Siblings:* Elements at the same level are siblings.
 - *Traversal:* Methods like parentNode, childNodes, etc., navigate the tree.

An example of a DOM tree





Brief Introduction to Javascript!

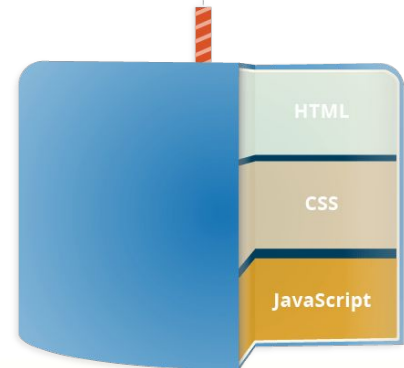
[Back to Agenda Page](#)





What is Javascript?

- A scripting or programming language that allows you to implement complex features on web pages
- Is capable of implementing things like displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc.
- It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS).





What can Javascript do?

- The core client-side JavaScript language consists of some common programming features that allow us to do things like:
 - Store useful values inside variables.
 - Operations on pieces of text (known as "strings" in programming).
 - Running code in response to certain events occurring on a web page.

HTML



CSS



JS





Javascript running order

When the browser encounters a block of JavaScript, it generally runs it in order, from top to bottom. This means that you need to be careful what order you put things in. For example:

```
const para = document.querySelector("p");

para.addEventListener("click", updateName);

function updateName() {

    const name = prompt("Enter a new name");

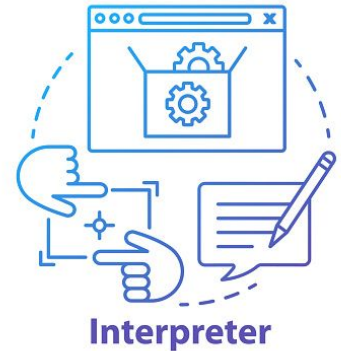
    para.textContent = `Player 1: ${name}`;

}
```



Interpreted code

- In interpreted languages, the code is run from top to bottom and the result of running the code is immediately returned
- The code doesn't need to be in a different form before the browser runs it
- Received in its programmer-friendly text form and processed directly



Interpreted vs Compiled code

- Compiled languages on the other hand are transformed (compiled) into another form before they are run by the computer.
- For example, C/C++ are compiled into machine code that is then run by the computer. The program is executed from a binary format, which was generated from source code.





So...which one is Javascript?

- JavaScript is a lightweight interpreted programming language.
- Web browser receives the JavaScript code in its original text form and runs the script
- From a technical standpoint, most modern JavaScript interpreters actually use a technique called **just-in-time compiling** to improve performance;
- However, JavaScript is still considered an interpreted language, since the compilation is handled at run time, rather than ahead of time.



Server-side versus Client-side code

- Client-side code is code that is run on the user's computer — when a web page is viewed, the page's client-side code is downloaded, then run and displayed by the browser.
- Server-side code on the other hand is run on the server, then its results are downloaded and displayed in the browser.
- Examples of popular server-side web languages include PHP, Python, Ruby, ASP.NET, and even JavaScript



How do you add JavaScript to your page?

- JavaScript is applied to your HTML page just like CSS
- Whereas CSS uses `<link>` elements to apply external style sheets and `<style>` elements to apply internal stylesheets to HTML, JavaScript only needs one friend in the world of HTML — the `<script>` element.
- Example:

```
<script>
```

```
// JavaScript goes here
```

```
</script>
```

test.html

```
document.addEventListener("DOMContentLoaded", () => {  
  function createParagraph() {  
    const para = document.createElement("p");  
    para.textContent = "You clicked the button!";  
    document.body.appendChild(para);  
  }  
  const buttons = document.querySelectorAll("button");  
  for (const button of buttons) {  
    button.addEventListener("click", createParagraph);  
  }  
});
```

test.js



Let's find out how javascript syntax looks like!

- Please have the Javascript cheatsheet provided by us side-by-side to understand this following section better!



Statements in Javascript

- Composed of Values, Operators, Expressions, Keywords, and Comments
- The statements are executed, one by one, in the same order as they are written.

```
let x, y, z;      // Statement 1
```

```
x = 5;           // Statement 2
```

```
y = 6;           // Statement 3
```

```
z = x + y;       // Statement 4
```



Semi-colons in Javascript

- Semicolons separate JavaScript statements.
- Add a semicolon at the end of each executable statement:

```
let a, b, c; // Declare 3 variables  
a = 5;      // Assign the value 5 to a  
b = 6;      // Assign the value 6 to b  
c = a + b;   // Assign the sum of a and b to c
```

- When separated by semicolons, multiple statements on one line are allowed:

```
a = 5; b = 6; c = a + b;
```



Code Blocks in Javascript

- JavaScript statements can be grouped together in code blocks, inside curly brackets { . . . }. The purpose of code blocks is to define statements to be executed together. One place you will find statements grouped together in blocks, is in JavaScript functions:

```
function myFunction() {  
  
    document.getElementById("demo1").innerHTML = "Hello  
Dolly!";  
    document.getElementById("demo2").innerHTML = "How are  
you?";  
}
```



Output in Javascript

- JavaScript can "display" data in different ways:
 - Writing into an HTML element, using `innerHTML`.
 - Writing into the HTML output using `document.write()`.
 - Writing into an alert box, using `window.alert()`.
 - Writing into the browser console, using `console.log()`.



Keywords in Javascript

- JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed or a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements



Values in Javascript

- The JavaScript syntax defines two types of values:
 - Fixed values
 - Variable values
- Fixed values are called Literals.
 - Numbers are written with or without decimals
 - Strings are text, written within double or single quotes
- Variable values are called Variables which is used to store data values.
 - Uses the keywords `var`, `let` and `const` to declare variables.
 - An equal sign is used to assign values to variables.

Eg. `let x;`

`x = 6;`



Literals in Javascript

- The two most important syntax rules for fixed values are:
 - Numbers are written with or without decimals
 - Strings are text, written within double or single quotes



Variables in Javascript

- In a programming language, variables are used to store data values.
- JavaScript uses the keywords `var`, `let` and `const` to declare variables.
- An equal sign is used to assign values to variables.
- In this example, `x` is defined as a variable. Then, `x` is assigned (given) the value 6:

```
let x;
```

```
x = 6;
```




Operators in Javascript

- JavaScript uses arithmetic operators (+ - * /) to compute values:
 - (5 + 6) * 10



Expressions in Javascript

- An expression is a combination of values, variables, and operators, which computes to a value. The computation is called an evaluation. For example, `5 * 10` evaluates to 50:
 - `5 * 10`
- Expressions can also contain variable values:
 - `x * 10`
- The values can be of various types, such as numbers and strings. For example, `"John" + " " + "Doe"`, evaluates to "John Doe":
 - `"John" + " " + "Doe"`



Comments in Javascript

- Not all JavaScript statements are "executed". Code after double slashes `//` or between `/*` and `*/` is treated as a comment. Comments are ignored, and will not be executed:

```
let x = 5;    // I will be executed
```

```
// x = 6;    I will NOT be executed
```



Declaring variables in Javascript

- You declare a JavaScript variable with the `var` or the `let` keyword:

```
var carName; or let carName;
```

- After the declaration, the variable has no value (technically it is undefined). To assign a value to the variable, use the equal sign:

```
carName = "Volvo";
```

- You can also assign a value to the variable when you declare it:

```
let carName = "Volvo";
```



Arithmetic operators in Javascript

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus
++	Increment
--	Decrement



Assignment operators in Javascript

Operator	Example
=	x = y
+=	x += y
-=	x -= y
*=	x *= y
/=	x /= y
%=	x %= y
**=	**=



Comparison operators in Javascript

Operator	Description
==	Equal to
===	Equal value and equal type
!=	Not equal
!==	Not equal value or not equal type
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Lesser than or equal to
?	Ternary operator



Logical operators in Javascript

Operator	Description
&&	Logical and
	Logical or
!	Logical not



Data types in Javascript

- JavaScript has 8 Datatypes:
 - String
 - Number
 - BigInt
 - Boolean
 - Undefined
 - Null
 - Symbol
 - Object



Example usage

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```



Functions in Javascript

- A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).

```
// Function to compute the product of p1 and p2
```

```
function myFunction(p1, p2) {  
    return p1 * p2;  
}
```



Return Functions in Javascript

- When JavaScript reaches a return statement, the function will stop executing. If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement. Functions often compute a return value. The return value is "returned" back to the "caller":

```
// Function is called, the return value will end up in x
let x = myFunction(4, 3);
function myFunction(a, b) {
  // Function returns the product of a and b
  return a * b;
}
```



String Methods in Javascript

- The length property returns the length of a string:

```
let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
let length = text.length;
```

- There are 3 methods for extracting a part of a string; `slice(start, end)`, `substring(start, end)`, `substr(start, length)`
- The `replace()` method replaces a specified value with another value in a string:

```
let text = "Please visit Microsoft!";
```

```
let newText = text.replace("Microsoft", "Facebook");
```



Conditionals in Javascript

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:
 - Use `if` to specify a block of code to be executed, if a specified condition is true
 - Use `else` to specify a block of code to be executed, if the same condition is false
 - Use `else if` to specify a new condition to test, if the first condition is false
 - Use `switch` to specify many alternative blocks of code to be executed



Example usage

- Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {  
    greeting = "Good day";  
}
```

- If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```



Example usage

- Use the switch statement to select one of many code blocks to be executed.

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```




Loops in Javascript

- JavaScript supports different kinds of loops:
 - `for` - loops through a block of code a number of times
 - `for/in` - loops through the properties of an object
 - `for/of` - loops through the values of an iterable object
 - `while` - loops through a block of code while a specified condition is true
 - `do/while` - also loops through a block of code while a specified condition is true



Example usage

```
for (let i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}
```

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```



Introduction to APIs!

[Back to Agenda Page](#)





Components of APIs

1. **Endpoints:** Specific URLs that represent different resources or actions that the API provides.
2. **HTTP Methods:** Like GET, POST, PUT, DELETE, etc., determine the type of interaction with the API.
3. **Parameters:** Data that can be sent with API requests to customize actions or retrieve specific information.



Examples of some famous APIs

- ***Social Media APIs*** like Twitter, Instagram, etc.
 - Purpose: Advertising products and services.
- ***E-Commerce APIs*** like Shopify, etc.
 - Purpose: Mainly used by developers to build custom storefronts, automate E-Commerce, etc.
- ***Artificial Intelligence APIs*** like ChatGPT, etc.
 - Purpose: mostly used to create highly customizable chatbot for organizations.





Introduction to NodeJS!

[Back to Agenda Page](#)





What is Node.js?

- A Javascript library created by Facebook
- A free open source server environment
- Is a tool for building UI components
- Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Uses JavaScript on the server





What can Node.js do?

- Generate dynamic page content
- Can collect form data
- Create, open, read, write, delete, and close files on the server
- Add, delete, modify data in the database



What is a Node.js File?

- Contain tasks that will be executed on certain events
- Must be initiated on the server before having any effect



Features of Node.js

1. ***Event-Driven Architecture***: Built on an event loop to define asynchronous. Enables applications to respond to events such as **incoming requests or data arriving from the database**.
2. ***Single Programming Language***: Uses Javascript on both the client and server side.
3. ***Open Source***: Freely available for anyone to use, modify and contribute to.



Node.js Application

```
const http = require('http');

const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});

server.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```



Introduction to React!

[Back to Agenda Page](#)





What is React?

- React is one of the most popular front-end JavaScript libraries in the field of web development.
- It is mainly maintained by Facebook and a large community of developers.
- A number of large, established companies (Netflix, Instagram, Airbnb, etc) use it to build their user interfaces and UI components.





What can React.js do?

1. ***Build User Interfaces:*** Build interactive and dynamic UIs by breaking them down into reusable components.
2. ***Updates with Virtual DOM:*** Uses virtual DOM to optimize updates by calculating the minimal changes needed, leading to faster UI rendering.
3. ***Component-Based Development:*** Promotes a component-based architecture, where you break down your user interface into small, reusable components.



Features of React.js

1. **SX** - It is an extension of ReactJS, which is not mandatory to be used but very beneficial if used since it is very easy to use.
2. **Virtual DOM:** React generates a virtual DOM, which is an in-memory data-structure cache. Only the final DOM updates have been updated in the browser's DOM.
3. **Javascript Expressions:** Curly brackets, for example, can be used to insert JS expressions into JSX files.



Advantages of React.js

1. ReactJS employs virtual dom, which makes use of an in-memory data-structure cache, and only the most recent modifications are updated in the browser's dom. This speeds up the app.
2. Using the react component feature, you may design components of your choice. The components are reusable and useful for code maintenance.
3. Since Reactjs is an open-source javascript library, it is simple to learn.
4. ReactJS may be used to create sophisticated user interfaces for both desktop and mobile apps.



Disadvantages of React.js

1. The majority of the code is written in JSX, which means that HTML and CSS are part of the javascript code. This can be perplexing because most other frameworks like to keep HTML separate from the javascript code.
2. ReactJS has a huge file size.



React.js Application

```
import React from 'react';

class HelloWorld extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello World!</h1>
      </div>
    );
  }
}

export default HelloWorld;
```



QUIZ TIME! <3

Quiz link:

<https://www.gimkit.com/>

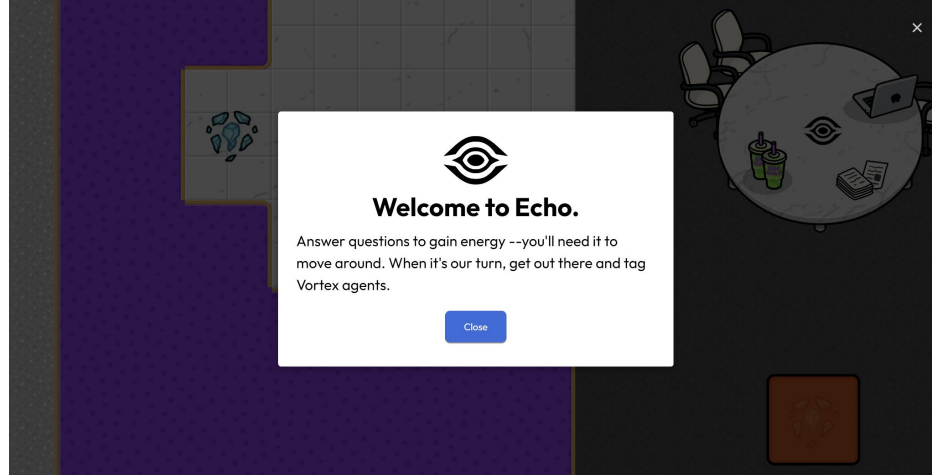


[Back to Agenda Page](#)



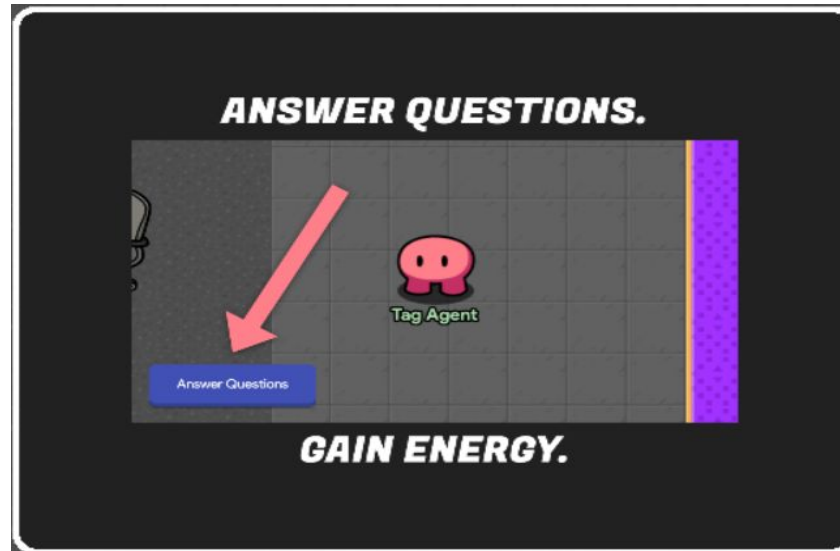
A guide on how to use the GimKit quiz interface:

- Once the quiz starts, you might find yourself on a page looking similar to this. All of you will be assigned either groups (Vortex or Echo):



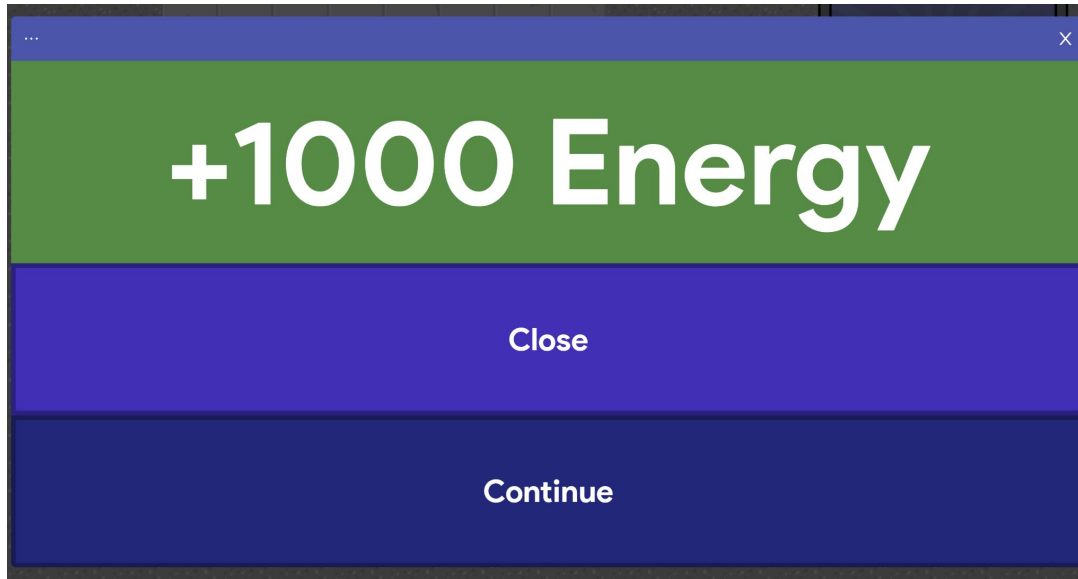
A guide on how to use the GimKit quiz interface:

- Click on the bottom left corner to answer questions (answering questions gives you energy which is required for you to move around):



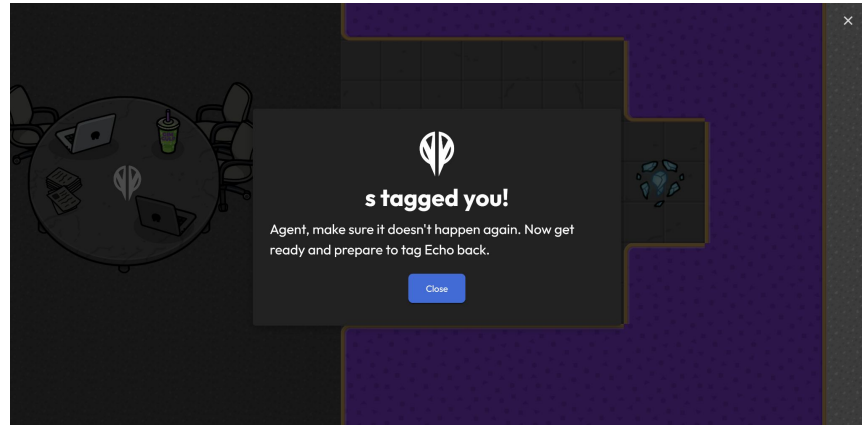
A guide on how to use the GimKit quiz interface:

- As you answer questions, the amount of energy that you own will increase:



A guide on how to use the GimKit quiz interface:

- After answering correct questions, you can close the question-answer pop-up. With the energy gained, you may choose to run away from them or tag them. Upon getting tagged, the opposing team gains points:





A guide on how to use the GimKit quiz interface:

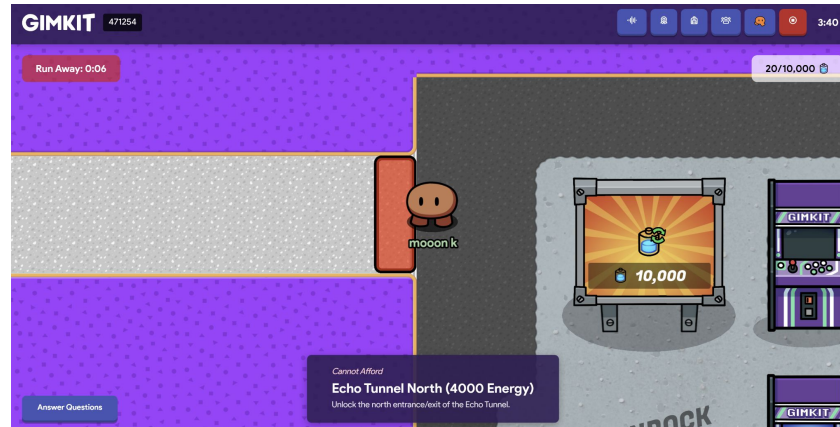
- Each team will have an allocated timer for them. The timer will be shown on the top right of the game. Your team needs to act within that allocated timer. After the timer is over, the roles would switch over.

Tag: 0:25

Run Away: 0:06

A guide on how to use the GimKit quiz interface:

- You may also redeem your energy for power-ups such as these to either teleport to your opposing team's base to tag them easily, or teleport back to your own team's base to evade them from tagging you.



A guide on how to use the GimKit quiz interface:

- Redeem power-ups with energy. Here are the list of power-ups which can be found in the middle of the map.



Speed Upgrade



Endurance Upgrade



Energy Per
Question Upgrade



Efficiency Upgrade

QR code for quiz





Break Time!

[Back to Agenda Page](#)





Hands-On!

[Back to Agenda Page](#)

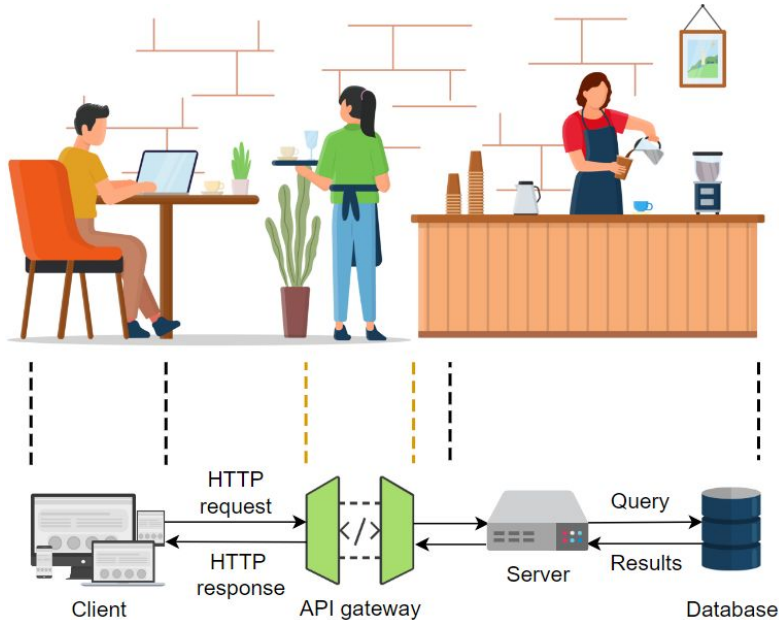




Integrating APIs into Javascript

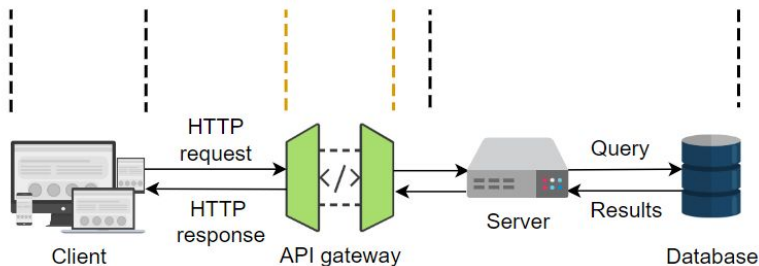
1. Obtain API keys from your preferred API provider.
2. Set up your development environment (code editor or IDE).
3. Create an HTML file for your chatbot interface with a chat window.
4. Link your HTML file to a JavaScript file for API interactions.
5. Capture user input from the chat window in JavaScript.
6. Use fetch function or an HTTP library like Axios to send a request to the API using the captured input, API key, and parameters.
7. Receive and extract the model-generated message from the API response.
8. Add the bot's response to the chat window.

API Recap!



- **Customer**
 - Client (You)
- **Menu**
 - Available endpoints / routes in the API
- **Order**
 - The API Request
 - dish (endpoint)
 - extra details (parameters)
- **Kitchen**
 - The API Server Backend
 - processes your request
 - and prepare response
- **Food**
 - The API Response
 - contains data/results you requested

API Recap!



Food

The API Response

- contains data/results you requested



Customer

Client



Menu

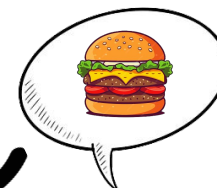
Available endpoints / routes in the API



Kitchen

The API Server Backend

- processes your request
- and prepare response



Order

The API Request

- dish (endpoint)
- extra details (parameters)

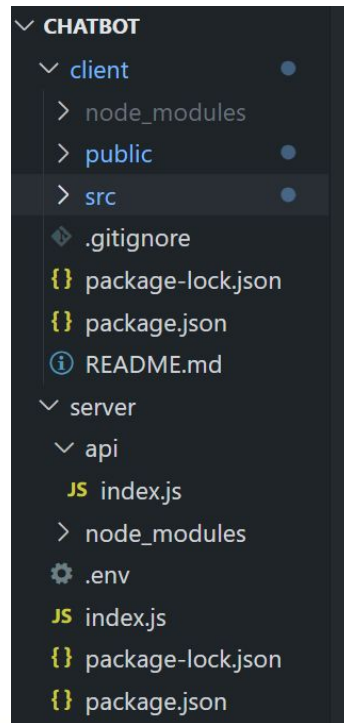
Files & Folders required

client →

- **public:**
 - contains static assets and the HTML entry point
- **src:**
 - holds the source code of the React application

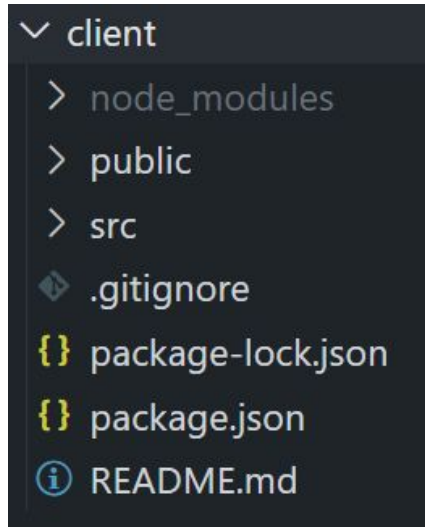
server →

- **api:**
 - to send POST request to the API
- **index.js:**
 - to manage server
- **.env:**
 - (environment) stores the API key

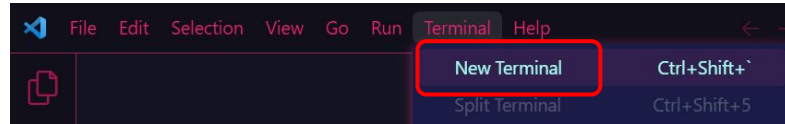


Downloading the Files

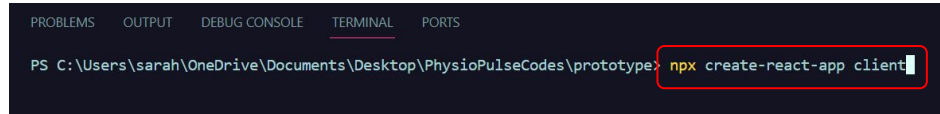
1. What you have Downloaded:



2. Create Required Folders/Files

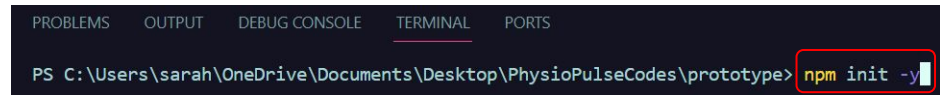


In **main directory** terminal:



→ creates 'client' folder with default React code

In **'server' folder**:



→ loads node modules



importing required packages

Packages to Install:

express, express-http-proxy

- popular Node.js framework for a server handling HTTPS requests.

axios

- JS library widely-used for HTTP requests in browser and Node.js.

http

- create HTTP servers and send HTTP requests in Node.js apps.

dotenv

- simplifies loading environment variables from a .env file into Node.js apps.

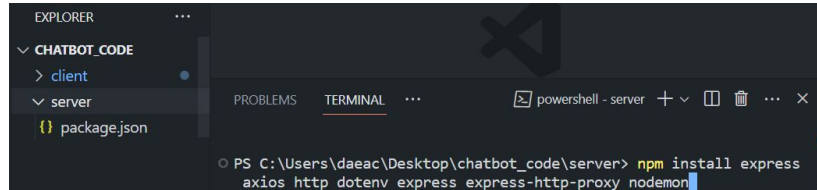
nodemon

- utility for Node.js that helps in the development process

importing required packages

How to Install Packages:

1. In 'server' terminal:

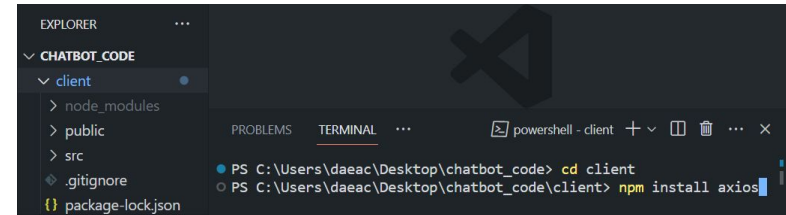


The screenshot shows the VS Code interface with the Explorer panel on the left showing the project structure: CHATBOT_CODE > client > server. The server directory is selected, and its package.json file is open. The Terminal panel at the bottom shows a PowerShell session in the server directory. The command entered is `npm install express axios http dotenv express http-proxy nodemon`.

```
PS C:\Users\daaac\Desktop\chatbot_code\server> npm install express  
axios http dotenv express http-proxy nodemon
```

'cd server'
'npm install express express-http-proxy axios
http dotenv nodemon'

2. In 'client' terminal:



The screenshot shows the VS Code interface with the Explorer panel on the left showing the project structure: CHATBOT_CODE > client. The client directory is selected, and its package-lock.json file is open. The Terminal panel at the bottom shows a PowerShell session in the client directory. The command entered is `npm install axios`.

```
PS C:\Users\daaac\Desktop\chatbot_code\client> npm install axios
```

'cd client'
'npm install axios'



Integrating APIs into Javascript

1

Get API key(s)

2

Call Your API

Using fetch function OR axios (HTTP library)

3

Extract & Display the API Response

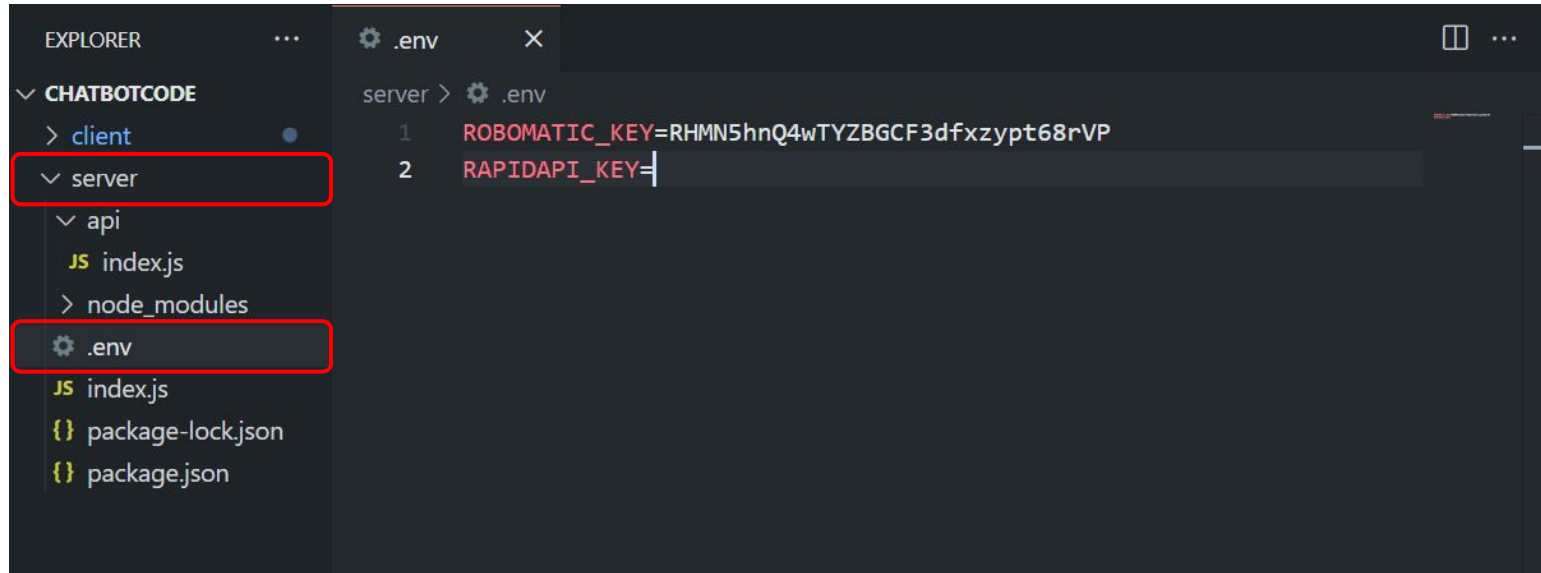


Getting API Key

RoboMatic.AI on RapidAPI

- ☐ Create RapidAPI account
- ☐ Find RoboMatic.AI
- ☐ Copy your
X-RapidAPI-Key and
RoboMaticAI's Key

Where to Put API Key?





Rest of the Code!





Example of Javascript Code for integrating APIs

```
document.getElementById('userInput').addEventListener('keydown', function(event) {  
  if (event.key === "Enter") {  
    let userInput = document.getElementById('userInput').value;  
    document.getElementById('userInput').value = '';  
    fetch('<API website link>', {  
      method: 'POST',  
      headers: {  
        'Content-Type': 'application/json',  
        'Authorization': 'Bearer YOUR_API_KEY'  
      },  
      body: JSON.stringify({  
        model: 'gpt-3.5-turbo',  
        messages: [  
          {  
            role: 'user',  
            content: userInput  
          }  
        ]  
      })  
    })  
    .then(response => response.json())  
    .then(data => {  
      let botResponse = data.choices[0].message.content;  
      document.getElementById('chatWindow').innerHTML += `<div class="botMessage">${botResponse}</div>`;  
    })  
    .catch(error => console.error('Error:', error));  
  }  
});
```



Please help us fill up this feedback form here:

