

AI Chatbot Assist for Software Development

{codenation}[®]

Learning Objectives

- ✓ Learn to build context, refine prompts and create a collaborative environment.
- ✓ Enable learners to effectively use AI Chatbots as a tool to seek assistance with programming tasks.

Large Language Models

What is a Large Language Model (LLM)?

A large language model (**LLM**) is a type of artificial intelligence (**AI**) program that can recognise and generate text, among other tasks.

LLMs are trained on huge sets of data – hence the name "large."

LLMs are built on machine learning: specifically, a type of neural network called a transformer model.

What is a Large Language Model (LLM)?

In simpler terms, an **LLM** is a computer program that has been fed enough examples to be able to recognise and interpret human language or other types of complex data.

The **LLMs** don't really 'know' anything, but they are very good at figuring out which word follows another, which starts to look like real thought and creativity.

For instance...

Try each of these in ChatGPT:

1. "In 5 sentences, write the plot of Frozen"
2. "In 5 sentences, write the plot of Frozen but in the style of an American cowboy"
3. "In 5 sentences, write the plot of Frozen but in the style of an American cowboy with all of the characters as insects"

An LLM will predict a response based upon the **words** in front of it.



What ChatGPT is not...

- **It is not a human being.**
- **It is not conscious.**
- **It does not have feelings.**
- **It does not understand but responds – based upon data and probability.**

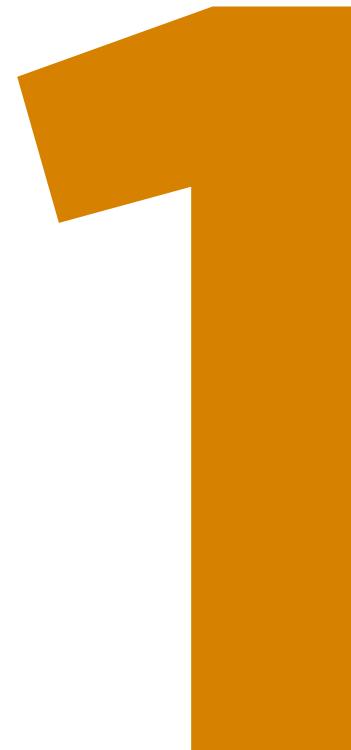


Introduction to ChatGPT and JavaScript Development

Importance of ChatGPT with Software Development

Instant Assistance and Learning

ChatGPT can provide instant assistance to developers by answering queries and providing solutions in real-time.



Importance of ChatGPT with Software Development

Code Generation and Suggestions

ChatGPT can generate code snippets based on provided requirements or suggest improvements to existing code.



Importance of ChatGPT with Software Development

Debugging and Problem Solving

ChatGPT can assist developers in debugging code by interpreting error messages, providing insights into potential issues, and offering step-by-step guidance to resolve them.



Importance of ChatGPT with Software Development

Documentation and Knowledge Sharing

ChatGPT can help developers generate documentation, code comments, and explanations for their projects.



Importance of ChatGPT with Software Development

Prototype and Design Assistance

ChatGPT can aid developers in prototyping and designing software solutions by providing design patterns, architectural recommendations, and user interface suggestions.



Getting Started with ChatGPT

Getting Started with ChatGPT

Start by going to chat.openai.com and then use an email address or a Google or Microsoft account to sign up.

You need to create an account on the OpenAI website to log in and access ChatGPT, but it is **free.**

If you've never created an account, click **Sign up and follow the prompts to enter your information.**

OpenAI does require a valid phone number for verification to create an account on its website.

Getting Started with ChatGPT

Accept the disclaimers from ChatGPT

Next, you may see a few disclaimers before you can chat with the AI chatbot.

Read and accept them one by one to enter the chat.

ChatGPT

Tips for getting started

Ask away

ChatGPT can answer questions, help you learn, write code, brainstorm together, and much more.

Don't share sensitive info

Chat history may be reviewed or used to improve our services. Learn more about your choices in our [Help Center](#).

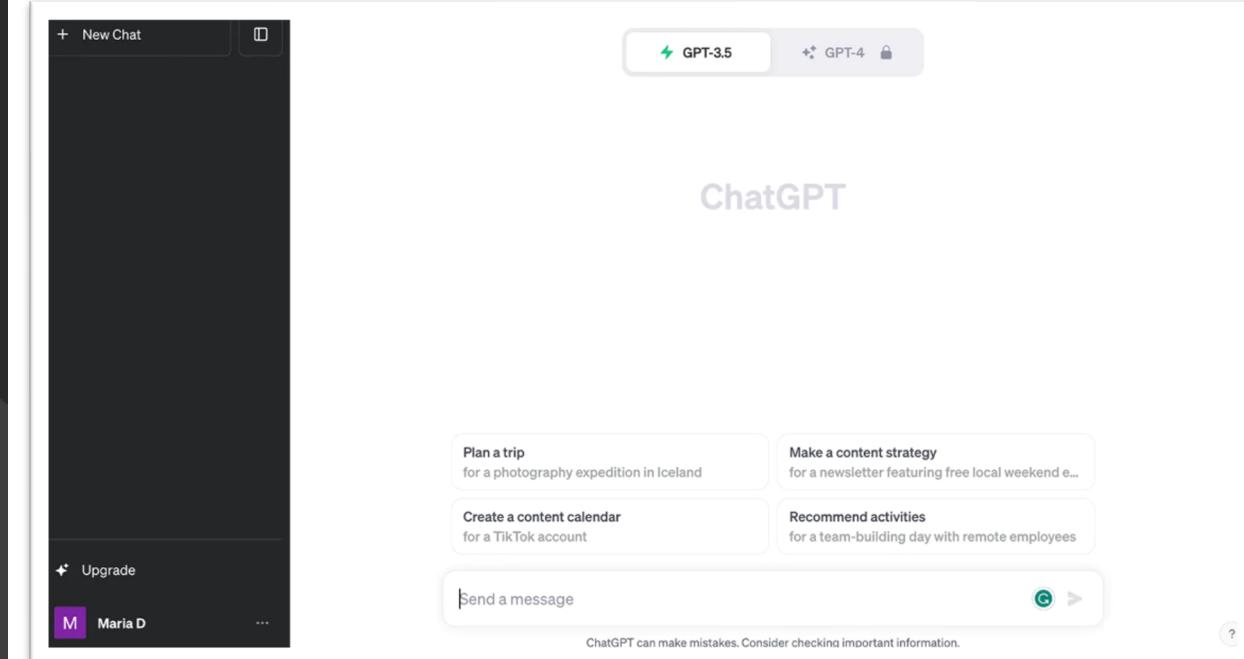
Check your facts

While we have safeguards, ChatGPT may give you inaccurate information. It's not intended to give advice.

Okay, let's go

Getting Started with ChatGPT

Once you've logged in to your OpenAI account on the ChatGPT side of the website, it's time to get to know the AI tool's window.

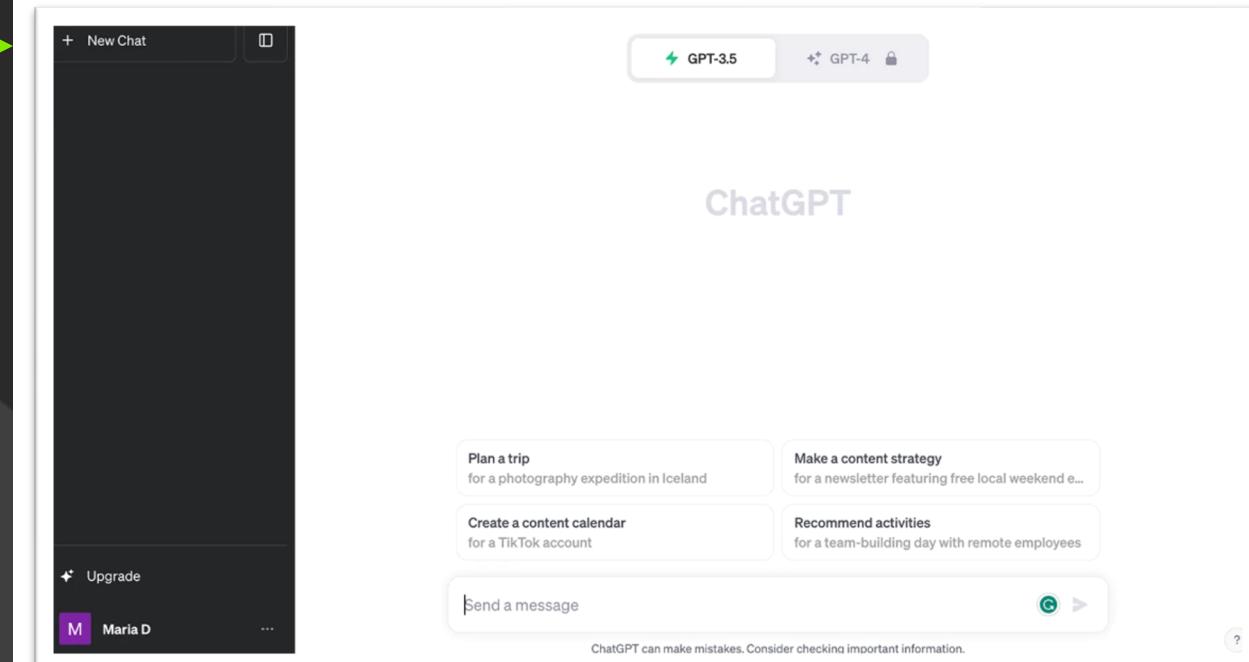


Getting Started with ChatGPT

New chat

Click to start a fresh conversation at any time.

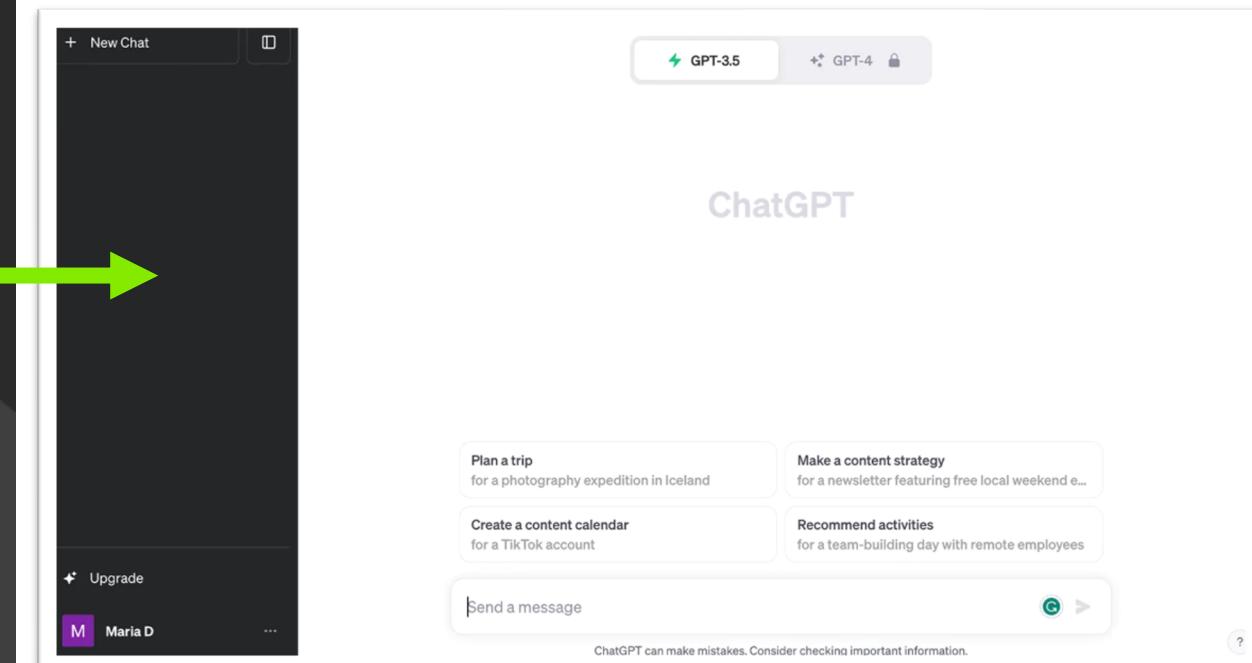
ChatGPT remembers what was previously discussed in a conversation and will respond with context, so starting a new chat will create a new discussion without context.



Getting Started with ChatGPT

Chat history

This keeps your previous conversations accessible in case you need to go back to one.



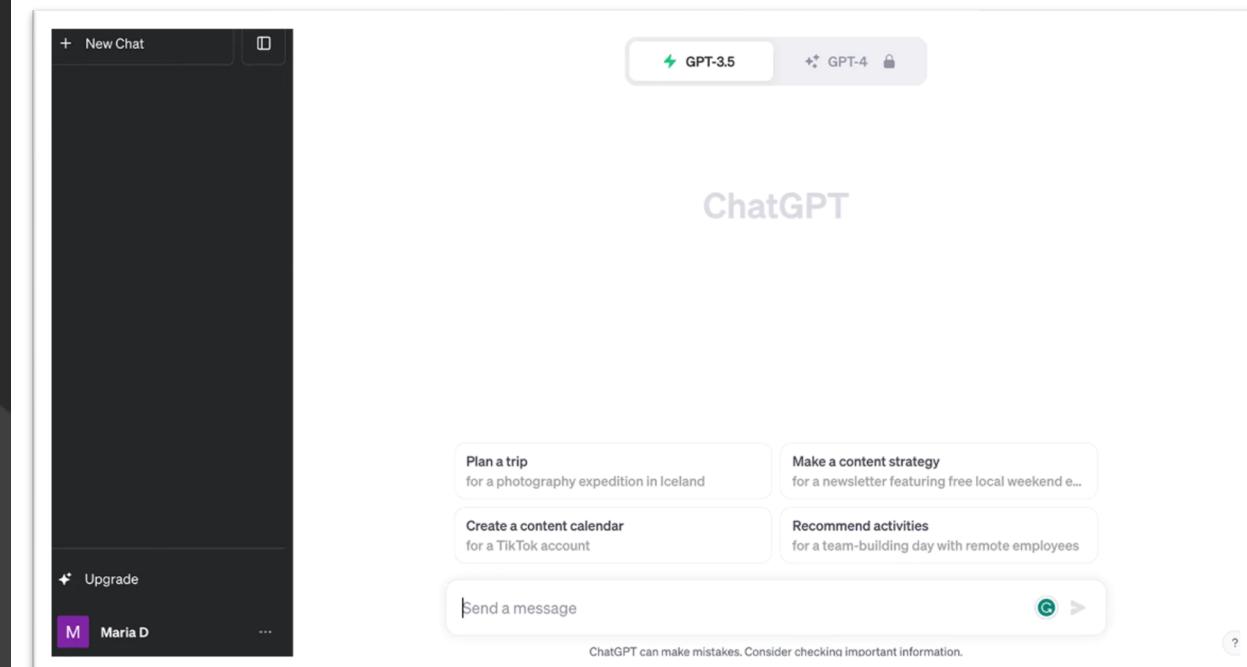
Getting Started with ChatGPT

Account

Clicking on your name on the bottom left of your screen will give you access to your account information, including settings, the option to log out, get help, and access FAQs from OpenAI.



You'll also see an upgrade option here.



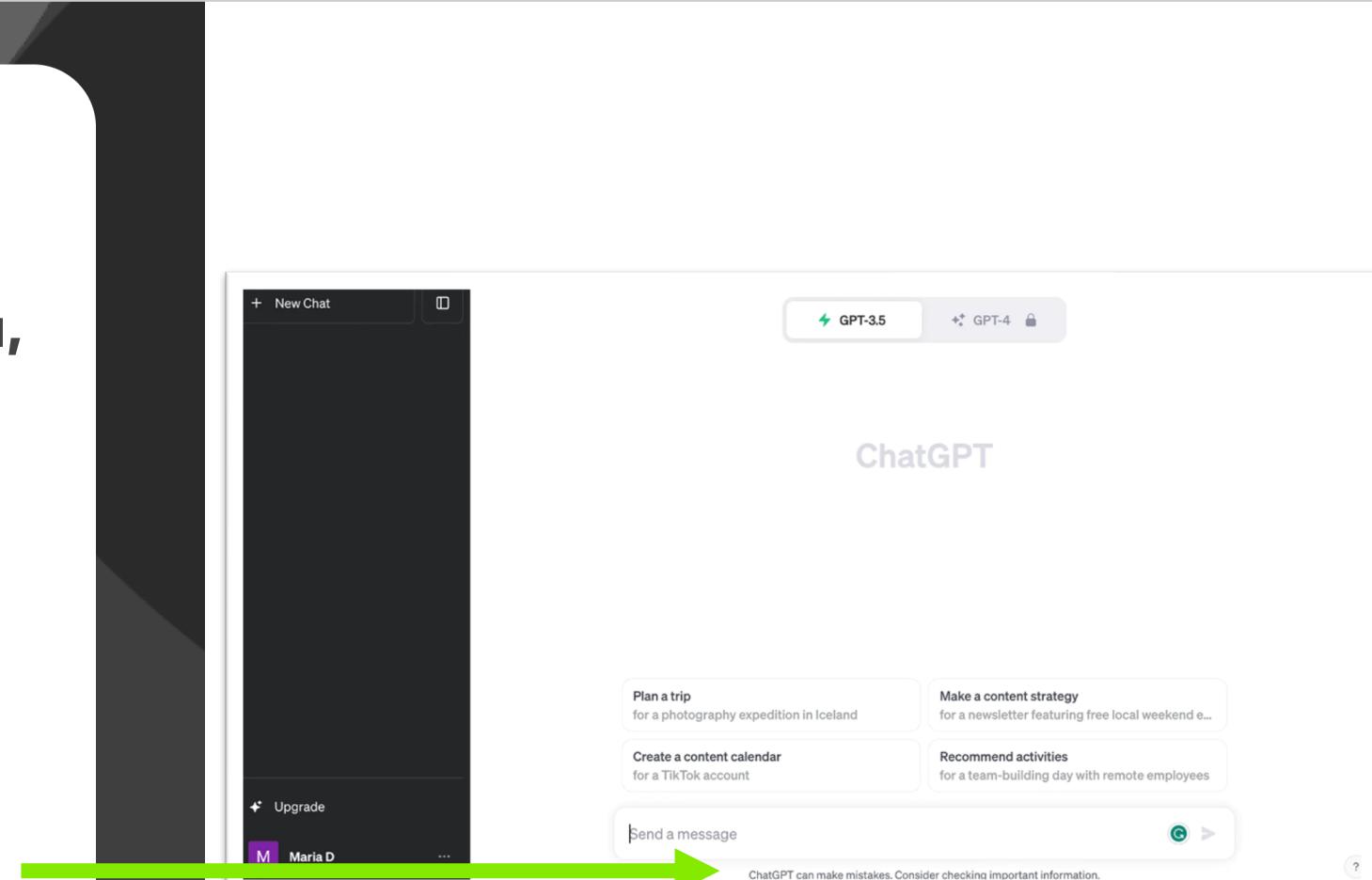
Getting Started with ChatGPT

ChatGPT version

OpenAI includes some fine print below the text input area, where you can read the disclaimer that, "ChatGPT can make mistakes.

Consider checking important information."

This section used to show the version of the ChatGPT model currently in use, but this has been discontinued.



Getting Started with ChatGPT

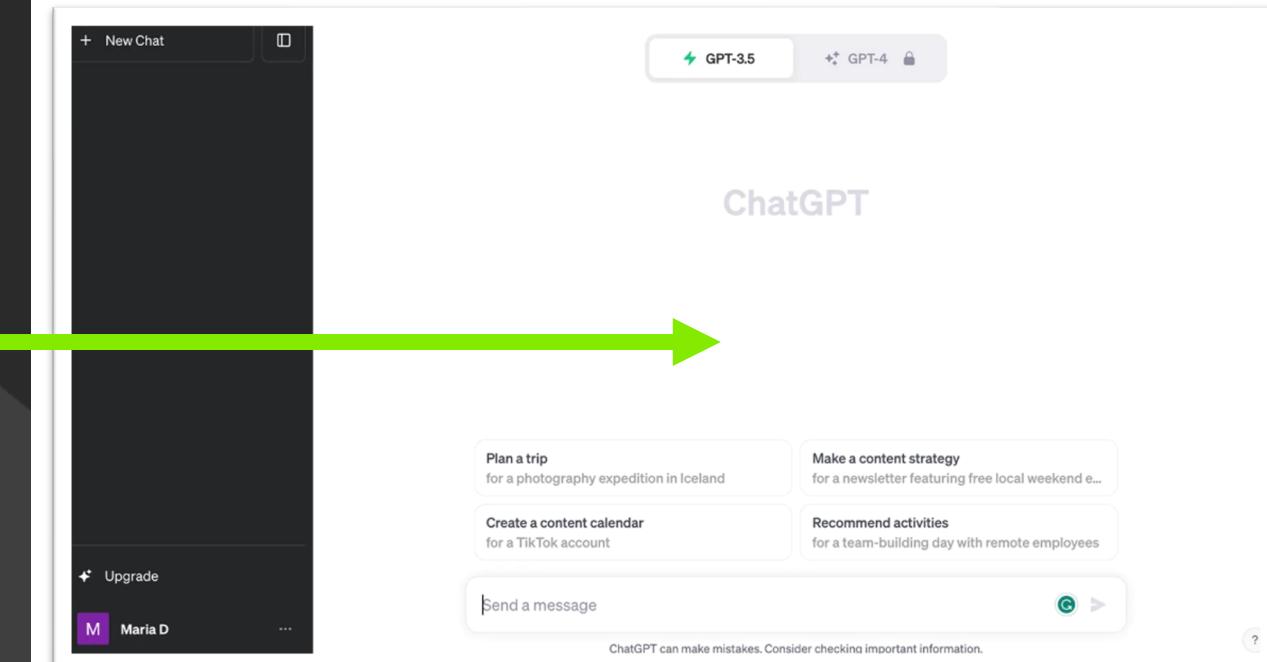
ChatGPT's responses

Whenever ChatGPT responds to your queries, the logo will appear on the left.

The Copy, Thumbs Up, and Thumbs Down buttons appear to the right of each response.

You can copy the text to your clipboard to paste it elsewhere and provide feedback on whether the response was accurate.

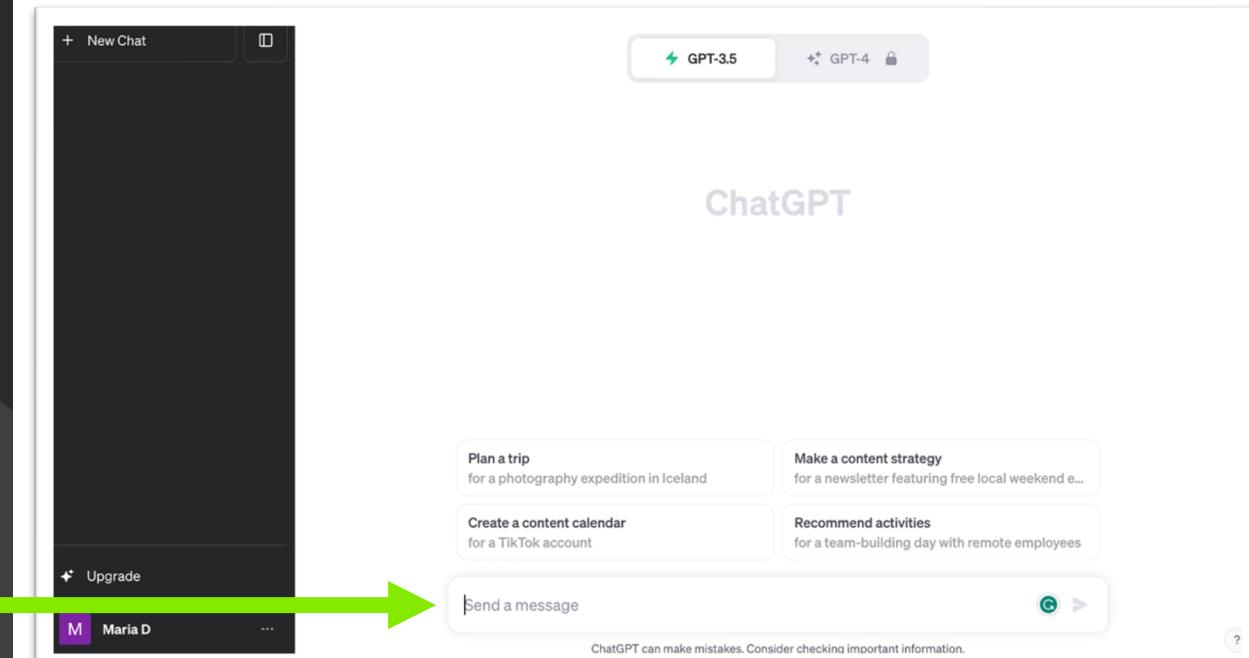
This process helps fine tune the AI tool.



Getting Started with ChatGPT

Your prompts

The questions or prompts you send the AI chatbot appear in the middle of the chat window with your account photo or initials to the left.



Getting Started with ChatGPT

**Now that you know how to log in and access ChatGPT,
it's time to start.**

**You can ask the language model any questions you
might have and see what answers you get - the
possibilities are literally endless.**

**We will focus our interactions on ChatGPT to work
related prompts.**

Getting Started with ChatGPT

**Don't fear AI.
Leverage its capabilities.**

If you're a Software Developer, you might experiment with generative AI tools and wonder what it will mean for your profession and how it will change your work.

Generative AI tools such as ChatGPT have caused a stir among the developer community.

Some fear it will take their jobs, while others prefer to ignore it.

Both attitudes are mistaken because a developer who integrates AI into their workflow can experience an incredible productivity boost.

Getting Started with ChatGPT

Generative AI lacks context.

Software Engineers can add more details about the purpose of the software they're creating and the people who will be using it.

It's not just a bunch of programs sprung together with regurgitated code.

Engineering requires a lot that AI can't replace, like context, making it near impossible for AI to load into a single model, train that model, and incorporate the predictive capability of humans who understand what's going to be necessary in five years.

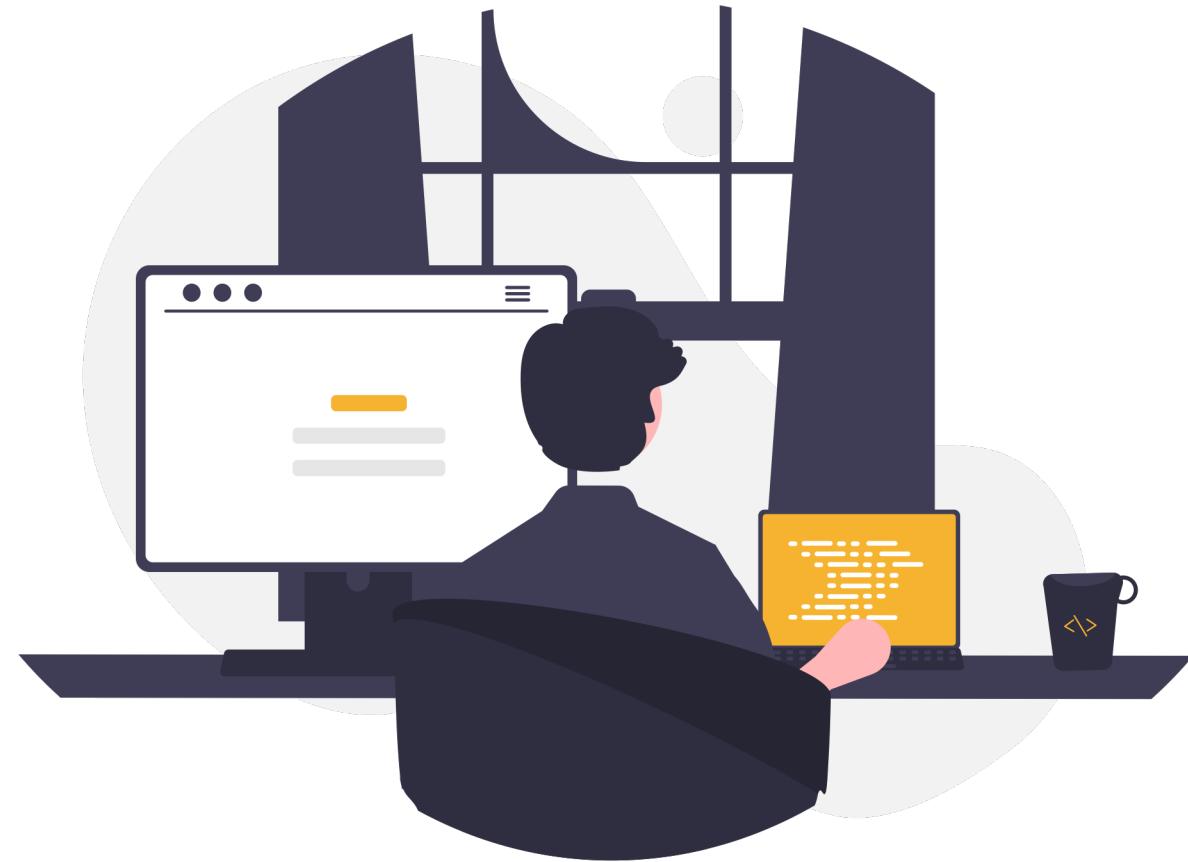
There are a lot of big picture decisions unique to different businesses that AI will simply never be able to handle.

Code Generation and Review

Generating code

Why would Software Developers use AI to generate new code?

- **Reduces the risk of human error**
- **Coders can use plain language to describe the functionality and product**
- **It saves time**
- **It delivers high-quality code**



Generating code

So, how can we generate code with AI?

We will examine some practical ChatGPT / Gemini prompts that can help Software Developers complete their daily tasks and projects.

With just a few clicks, you can easily edit and copy the prompts on the site to fit your specific needs and preferences.

Generating code

Try these example prompts in ChatGPT and/or Gemini:

1. How do you declare a variable in JavaScript?
2. Can you explain the difference between == and === operators?
3. How do you create an array in JavaScript?
4. What are the different ways to iterate through an array?
5. How can you access the value of an object property using a variable?

These example prompts demonstrate how ChatGPT can be used to generate questions and statements that will guide you through various aspects of JavaScript programming, improving your understanding of the language and enhancing your coding skills.

Generating code

Variables and Data Types

Understanding variables and data types is fundamental to mastering JavaScript.

ChatGPT can generate prompts that help users learn how to declare, assign, and manipulate variables, as well as differentiate between various data types.

PROMPT:

How do you create a variable that holds a string value in JavaScript?

RESPONSE:

```
javascript Copy code

// Using var
var myStringVar = "This is a string";

// Using let
let myStringLet = 'Another string';

// Using const
const myStringConst = "Yet another string";
```

Generating code

Variables and Data Types

Understanding variables and data types is fundamental to mastering JavaScript.

ChatGPT can generate prompts that help users learn how to declare, assign, and manipulate variables, as well as differentiate between various data types.

PROMPT:

Can you give an example of type coercion in JavaScript?

RESPONSE:

```
javascript
let x = 10; // x is a number
let y = "5"; // y is a string

let result = x + y; // JavaScript will coerce the number to a string and concatenate
console.log(result); // Output: "105"
```

Generating code

Variables and Data Types

Understanding variables and data types is fundamental to mastering JavaScript.

ChatGPT can generate prompts that help users learn how to declare, assign, and manipulate variables, as well as differentiate between various data types.

PROMPT:

How do you convert a string to a number in JavaScript?

RESPONSE:

```
javascript Copy code
let stringValue = "123";
let numberValue = parseInt(stringValue);

parseFloat(): Similar to `parseInt()`, but it parses a string argument and returns a floating point number.

javascript Copy code
let stringValue = "123.45";
let numberValue = parseFloat(stringValue);

Number() constructor: This converts the argument to a number, either integer or floating point.

javascript Copy code
let stringValue = "123";
let numberValue = Number(stringValue);
```

Generating code

Operators and Expressions

Operators and expressions are essential components of JavaScript programming, enabling users to perform calculations and comparisons.

PROMPT:

How does short-circuit evaluation work with logical operators?

RESPONSE:

1. Logical AND (`&&`):

- If the first operand evaluates to `false`, the entire expression evaluates to `false`, and the evaluation stops, as the logical AND operation requires both operands to be `true` for the whole expression to be `true`.
- If the first operand evaluates to `true`, the evaluation continues to the second operand. If the second operand also evaluates to `true`, the entire expression evaluates to `true`. If the second operand evaluates to `false`, the entire expression evaluates to `false`.

Example:

```
javascript Copy code
let a = false;
let b = true;
let result = a && b; // Since 'a' is false, 'b' won't be evaluated
console.log(result); // Output: false
```

Generating code

Function Declaration and Invocation

Are reusable blocks of code that can be defined and called upon to perform specific tasks.

ChatGPT can generate prompts to help users understand the process of declaring, invoking, and working with functions in JavaScript.

PROMPT:

What is an anonymous function, and when would you use one?

RESPONSE:

```
javascript Copy code
let numbers = [1, 2, 3, 4, 5];

// Using the Array.prototype.map() method with an anonymous function
let squaredNumbers = numbers.map(function(number) {
    return number * number;
});

console.log(squaredNumbers); // Output: [1, 4, 9, 16, 25]
```

In this example, we have an array `numbers` containing some numeric values. We're using the `map()` method, which applies a function to each element of the array and returns a new array containing the results of applying the function.

Generating code

Higher-order Functions and Callbacks

Higher-order functions and callbacks are advanced JavaScript concepts that allow users to write more efficient and versatile code.

ChatGPT can create prompts that guide users through understanding these concepts and applying them in their code.

PROMPT:

How do you pass a function as an argument to another function in JavaScript?

RESPONSE:

In JavaScript, you can pass a function as an argument to another function by simply specifying the function name (without parentheses) as the argument when calling the higher-order function.

Here's an example:

javascript Copy code

```
// Define a higher-order function that takes another function as an argument
function higherOrderFunction(callback) {
  // Call the callback function
  callback();
}

// Define a function that will be passed as an argument
function callbackFunction() {
  console.log("This is the callback function");
}

// Pass the function as an argument to the higher-order function
higherOrderFunction(callbackFunction);
```

Generating code

Leveraging ChatGPT / Gemini's ability to generate prompts for functions, users can deepen your understanding of these essential JavaScript building blocks and improve your coding skills.

Generating code

Developers can enjoy an engaging and interactive learning experience, explore alternative solutions, and better understand the logic behind their code.

To use ChatGPT / Gemini effectively you must have a solid foundation in your programming language (JS, Python, C#, etc).

Real world example 1

Sort and Star

You will be given a list of strings.
You must sort it alphabetically (case-sensitive and based on the ASCII values of the chars) and then return the first value.

The returned value must be a string and have "***" between each of its letters.

You should not remove or add elements from/to the array.

Example list: ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows", "perhaps"]

Predicted outcome: b***i***t***c***o***i***n

Real world example 1

Be polite

Include
the list

Provide context

Explain
the
problem
concisely

PROMPT:

Hello ChatGPT. Imagine that you are a beginner programmer. You have a list of strings: ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows", "perhaps"] and you must sort it alphabetically (case-sensitive and based on the ASCII values of the chars) and then return the first value. The returned value must be a string and have "*" between each of its letters. Write the function to do this.**

How do you want the result displayed?

Real world example 1

PROMPT:

Hello ChatGPT. Imagine that you are a beginner programmer. You have a list of strings: ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows", "perhaps"] and you must sort it alphabetically (case-sensitive and based on the ASCII values of the chars) and then return the first value. The returned value must be a string and have "****" between each of its letters. Write the function to do this.

RESPONSE:



ChatGPT

Certainly! Here's a JavaScript function that sorts the list alphabetically, retrieves the first string, and then inserts "****" between each letter:

```
javascript Copy code

function formatFirstString(strings) {
    // Sort the array alphabetically
    strings.sort();

    // Take the first string from the sorted array
    const firstString = strings[0];

    // Convert the first string to have "****" between each letter
    const formattedString = firstString.split('').join('****');

    // Return the formatted string
    return formattedString;
}

// Example usage:
const strings = ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows", "perhaps"];
const formattedFirstString = formatFirstString(strings);
console.log(formattedFirstString); // Output: b****i****t****c****o****w****i****n
```

You can use this function by passing your list of strings to it. It will return the formatted first string from the sorted list.

Try the same prompt in Gemini.

Real world example 1

PROMPT:

Hello ChatGPT. Imagine that you are a pro-level programmer. ... <same prompt> ...

RESPONSE:

ChatGPT
Certainly! Here's a JavaScript function that sorts the list alphabetically, retrieves the first string, and then inserts "****" between each letter:

```
javascript Copy code  
  
function formatFirstString(strings) {  
    // Sort the array alphabetically  
    strings.sort();  
  
    // Take the first string from the sorted array  
    const firstString = strings[0];  
  
    // Convert the first string to have "****" between each letter  
    const formattedString = firstString.split('').join('****');  
  
    // Return the formatted string  
    return formattedString;  
}  
  
// Example usage:  
const strings = ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows",  
const formattedFirstString = formatFirstString(strings);  
console.log(formattedFirstString); // Output: b****i***t***c***o***i***n
```

You can use this function by passing your list of strings to it. It will return the formatted first string from the sorted list.

Real world example 1

PROMPT:
You can do better.

RESPONSE:

ChatGPT

Certainly! Here's an even more optimized implementation that directly sorts the array and then formats the first string without storing any intermediate variables:

javascript

```
function formatFirstString(strings) {
    // Sort the array alphabetically
    strings.sort();

    // Return the formatted first string
    return strings[0].split('').join('***');
}

// Example usage:
const strings = ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows",
const formattedFirstString = formatFirstString(strings);
console.log(formattedFirstString); // Output: b***i***t***c***o***i***n
```

 Copy code

Real world example 1

PROMPT:
You can do even better than that.

RESPONSE:

Certainly! I can leverage modern JavaScript features like arrow functions and template literals to make the code even more concise and expressive:

```
javascript Copy code
const formatFirstString = strings =>
  strings.sort()[0].split('').join('***');

// Example usage:
const strings = ["take", "over", "bitcoin", "the", "world", "maybe", "who", "knows",
const formattedFirstString = formatFirstString(strings);
console.log(formattedFirstString); // Output: b***i***t***c***o***i***n
```

Real world example 2

Vowel Remover

**Create a function called shortcut to remove
the lowercase vowels (a, e, i, o, u) in a given string.
Don't worry about uppercase vowels.**

Examples:

"hello"

--> "hll"

"codenation"

--> "cdntn"

"goodbye"

--> "gdby"

"HELLO"

--> "HELLO"

Predicted outcomes:

Real world example 2

Be polite

Provide context

Explain
the
problem
concisely

PROMPT:

Hello ChatGPT. Imagine that you are a beginner programmer. I would like you to write a function called 'shortcut' which takes in a string and removes all the lowercase vowels, then returns the result as a string.

How do you want the result displayed?

Real world example 2

PROMPT:

Hello ChatGPT. Imagine that you are a beginner programmer. I would like you to write a function called 'shortcut' which takes in a string and removes all the lowercase vowels, then returns the result as a string.

RESPONSE:

Certainly! Here's a simple implementation of the `shortcut` function in JavaScript:

```
javascript Copy code

function shortcut(str) {
    // Define lowercase vowels
    const vowels = ['a', 'e', 'i', 'o', 'u'];

    // Remove lowercase vowels from the string
    const result = str.split('').filter(char => !vowels.includes(char)).join('');

    // Return the result as a string
    return result;
}

// Example usage:
const inputString = "Hello World";
const result = shortcut(inputString);
console.log(result); // Output: Hll Wrl
```

Explanation:

1. We define an array `vowels` containing all lowercase vowels.
2. We split the input string into an array of characters using `split('')`.
3. We use the `filter` method to remove all characters that are present in the `vowels` array.
4. We join the remaining characters back into a string using `join('')`.
5. We return the resulting string.

Try the same prompt in Gemini

Real world example 2

PROMPT:

Hello ChatGPT. Imagine that you are an **expert** programmer. I would like you to write a function called 'shortcut' which takes in a string and removes all the lowercase vowels, then returns the result as a string.

RESPONSE:

ChatGPT

Certainly! As an expert programmer, I can provide a more concise and efficient implementation using regular expressions:

javascript

```
function shortcut(str) {  
    return str.replace(/[aeiou]/g, '');  
  
// Example usage:  
const inputString = "Hello World";  
const result = shortcut(inputString);  
console.log(result); // Output: Hll Wrld
```

Copy code

Try the same prompt in Gemini.

Real world example 2

Swap the Head and the Tail

You need to swap the head and the tail of a specified array:

- The head (the first half) of array moves to the end.
 - The tail (the second half) moves to the start.
- The middle element (if it exists) remains in the same position.
 - Return the new array.

Examples

```
[ 1, 2, 3, 4, 5 ]    => [ 4, 5, 3, 1, 2 ]  
 \---/   \---/  
  head      tail
```

```
[ -1, 2 ]  => [ 2, -1 ]
```

```
[ 1, 2, -3, 4, 5, 6, -7, 8 ]  => [ 5, 6, -7, 8, 1, 2, -3, 4 ]
```

Real world example 2

Be polite

PROMPT:



Provide context

How do you want the result
displayed?

Explain the
problem
concisely

Write the initial ChatGPT prompt to code this JavaScript problem.

Real world example 2

Swap the Head and the Tail

Try your ChatGPT prompt to see if was interpreted well by the AI.

Test out your code using VS Code.

Did it work for the suggested tests?

Try the same prompt in Gemini.

Code completion

**ChatGPT can analyse
your codebase and
understand the
functionality you need
to implement.**

**Let's find out how you
can make this chatbot
complete the code for
you...**



Code completion: Activity 1

PROMPT: I've written this JavaScript code for a user to play a simple dice roll game where if they roll a 1 they lose their score, but if they roll any other number then they win. Can you complete it for me?

```
1 let roundScore = 0;
2 let gameOver = false;
3
4 const roll = document.getElementById("roll");
5 const reset = document.getElementById("reset");
6 const scoreTotal = document.getElementById("scoreTotal");
7 const resultDisplay = document.getElementById("p1Roll");
8 const diceImage = document.getElementById("diceImage");
9
10 roll.addEventListener("click", function() {
11     let dice1 = Math.floor(Math.random() * 6) + 1;
12     diceImage.src = `./diceimages/${dice1}.gif`;
13     if (!gameOver) { if (dice1 === 1) {
14         roundScore = 0;
15         resultDisplay.innerHTML = "Game Over - You Lose";
16         gameOver = true;
17         roll.disabled = true;
18     } else {
19         roundScore += dice1;
20         scoreTotal.innerHTML = "Total Score: " + roundScore;
```

Code completion: Activity 1

PROMPT: I've written this JavaScript code for a user to play a simple dice roll game where if they roll a 1 they lose their score, but if they roll any other number then they win. Can you complete it for me?

```
1 if (roundScore >= 20) {  
2   resultDisplay.innerHTML = "You Win!";  
3   gameOver = true;  
4   roll.disabled = true; }  
5 }  
6 );  
7 );  
8  
9 reset.addEventListener("click", function() {  
10   roundScore = 0;  
11   scoreTotal.innerHTML =  
12     "Total Score:"; resultDisplay.innerHTML = "";  
13   gameOver = false;  
14   roll.disabled = false;  
15 });
```

```
1 let roundScore = 0;
2 let gameOver = false;
3
4 const roll = document.getElementById("roll");
5 const reset = document.getElementById("reset");
6 const scoreTotal = document.getElementById("scoreTotal");
7 const resultDisplay = document.getElementById("p1Roll");
8 const diceImage = document.getElementById("diceImage");
9
10 roll.addEventListener("click", function() {
11     let dice1 = Math.floor(Math.random() * 6) + 1;
12     diceImage.src = `./diceimages/${dice1}.gif`;
13     if (!gameOver) { if (dice1 === 1) {
14         roundScore = 0;
15         resultDisplay.innerHTML = "Game Over - You Lose";
16         gameOver = true;
17         roll.disabled = true;
18     } else {
19         roundScore += dice1;
20         scoreTotal.innerHTML = "Total Score: " + roundScore;
```

```
1 if (roundScore >= 20) {
2     resultDisplay.innerHTML = "You Win!";
3     gameOver = true;
4     roll.disabled = true; }
5 }
6 });
7 });
8
9 reset.addEventListener("click", function() {
10     roundScore = 0;
11     scoreTotal.innerHTML =
12         "Total Score:"; resultDisplay.innerHTML = "";
13     gameOver = false;
14     roll.disabled = false;
15     diceImage.src = "./diceimages/11.gif";
16 });

});
```

RESPONSE:

ChatGPT

Your code looks well-structured and handles the basic functionality of the dice roll game. However, there are a few improvements we can make to enhance its readability and efficiency.

Firstly, let's ensure that the code initializes all the necessary HTML elements before the JavaScript code runs. This will prevent any errors related to undefined elements.

Next, I'll add comments to explain the purpose of each section of the code.

Here's the completed JavaScript code:

```
resultDisplay.textContent = "Game Over - You Lose";
gameOver = true;
rollButton.disabled = true;
} else {
    // Add roll value to round score
    roundScore += dice1;
    scoreTotal.textContent = "Total Score: " + roundScore;

    // Check if the player's score is 20 or more
    if (roundScore >= 20) {
        resultDisplay.textContent = "You Win!";
        gameOver = true;
        rollButton.disabled = true;
    }
}
});
```

Code completion: Activity 2

I've written this simple JavaScript code for a Coffee Shop customer to order a drink.

```
1 const Bob = new Customer('Bob', 5)
2 const drink1 = new DrinkPurchase('latte', 6);
3 const drink2 = new DrinkPurchase('frappe', 9);
4
5 drink1.showPrice();
6 drink2.showPrice();
7
8 let totalPrice = drink1.price + drink2.price;
9
10 console.log(`Total Cost: ${totalPrice}`);
11
12 if ((Bob.walletAmount - totalPrice) > 0){
13     console.log(`${Bob.customerName} can afford
this order.`);
14 }else{
15     console.log(`${Bob.customerName} cannot
afford this order.`);
16 }
```

Code completion: Activity 2

Task

Draft a ChatGPT prompt asking it to complete the program so that it has 2 classes: one for the Drink Purchase and one for the Customer.

Share your prompt in the group chat.

In VSCode, create a new .js file and add the code on this slide and the ChatGPT response.

Does it work?

```
1 const Bob = new Customer('Bob', 5)
2 const drink1 = new DrinkPurchase('latte', 6);
3 const drink2 = new DrinkPurchase('frappe', 9);
4
5 drink1.showPrice();
6 drink2.showPrice();
7
8 let totalPrice = drink1.price + drink2.price;
9
10 console.log(`Total Cost: ${totalPrice}`);
11
12 if ((Bob.walletAmount - totalPrice) > 0){
13     console.log(`${Bob.customerName} can afford
14         this order.`);
15 }else{
16     console.log(`${Bob.customerName} cannot
17         afford this order.`);
18 }
```

Code explanation and review

Not sure about the functionality of some code?

Simply provide ChatGPT / Gemini with the Code snippet to receive a full explanation.

Code explanation and review

Asking ChatGPT to explain the code can increase the understanding of project goals among all stakeholders and ensure everyone's on the same page.

It also improves collaboration by providing a common language for discussing complex technical concepts.

Code explanation and review

ChatGPT is a lifesaver when it comes to, the sometime daunting, code review.

This chatbot can provide instant feedback on code quality, readability, and potential issues.

You can use ChatGPT for software testing to identify various coding issues, such as syntax errors, inefficient algorithms, or potential security vulnerabilities, helping you to improve overall code quality.

Code explanation and review

Here's how to use ChatGPT to review your code:

- 1. Upload the code to be reviewed to ChatGPT.**
- 2. Wait for ChatGPT to analyse the code and provide feedback.**
- 3. Review the ChatGPT's feedback and make any necessary changes to the code.**
- 4. Continue the review process until you solve all the issues.**

RESPONSE:

BR



You

Please explain the following code:

```
gameSelect = document.getElementsByClassName("game-select");
gameSelection = document.getElementById("game-selection");
onePlayer = document.getElementById("one-player");
onePlayerButton = document.getElementById("one-player-button");
twoPlayerButton = document.getElementById("two-player-button");
onePlayerGameBox = document.getElementById("one-player-game-box");
onePlayerGameOverBox = document.getElementById("game-over");
onePlayerWinBox = document.getElementById("one-player-win");
onePlayerDice = document.getElementById("one-player-dice");
onePlayerRollDice = document.getElementById("one-player-roll-dice");
onePlayerRoundScore = document.getElementById("one-player-round-score");
onePlayerTotalScoreDisplay = document.getElementById("one-player-total-score");
onePlayerTotal = document.getElementById("one-player-total");
onePlayerGameOverButton = document.getElementById("one-player-game-over-button");
onePlayerWinButton = document.getElementById("one-player-win-button");
twoPlayer = document.getElementById("two-player");
```



ChatGPT

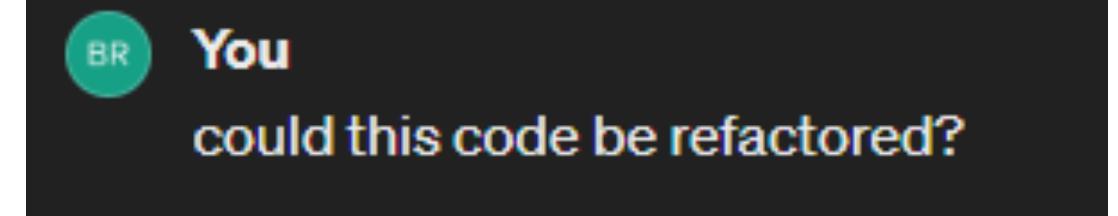
This code appears to be JavaScript code for a simple dice game. Let's break down what each section does:

1. **Variable Declarations:** This section initializes variables by selecting elements from the HTML document using their IDs or classes.
2. **startOnePlayer():** This function is called when the user starts a one-player game. It sets up the game interface and hides the game selection screen.
3. **startTwoPlayer():** Similar to `startOnePlayer()`, but for a two-player game.
4. **checkPlayerTurn():** This function checks whose turn it is in a two-player game and updates the interface accordingly.
5. **rollDice():** This function rolls the dice for a one-player game, updates the score, and checks for win/lose conditions.
6. **playerOneRollDice():** Rolls the dice for player one in a two-player game.
7. **playerTwoRollDice():** Rolls the dice for player two in a two-player game.
8. **onePlayerGameOver():** Handles the game over state for a one-player game.

Code explanation and review (unknown code)

You have a coding expert at your service – at any time of day, for free!

PROMPT:



RESPONSE:



ChatGPT

Yes, this code could be refactored to improve readability, maintainability, and potentially reduce redundancy. Here are some suggestions for refactoring:

1. **Use a Single Function for Rolling Dice:** Instead of having separate functions for rolling dice for each player, you could create a single function that takes player-specific elements as arguments.
2. **Consolidate Game State Handling:** Combine game state handling functions (`onePlayerGameOver`, `onePlayerWin`, `twoPlayerWin`, `twoPlayerReset`) into a single function that takes parameters to determine the game state.
3. **Use Constants for Player Identifiers:** Instead of using string literals like "player 1" and "player 2" throughout the code, define constants at the beginning of the script to improve readability and prevent typos.
4. **Simplify Element Selection:** If there are patterns in how elements are selected, consider using classes or data attributes to group related elements and select them more efficiently.
5. **Separate Event Handling Logic:** Move event listener registration to a separate function to improve code organization and readability.
6. **Reduce Repetitive Code:** Look for opportunities to eliminate duplicate code, such as similar operations performed on different elements.

Here's a simplified and refactored version of the code:

javascript

```
const ELEMENTS = {  
    gameSelection: document.getElementById("game-selection"),  
    onePlayer: document.getElementById("one-player"),  
    twoPlayer: document.getElementById("two-player"),  
    // Add other element selections here  
};  
  
const PLAYER = {  
    ONE: "player 1",  
    TWO: "player 2",  
};  
  
let playerTurn = PLAYER.ONE;  
  
const rollDice = (scoreDisplay, totalDisplay) => {  
    const randomNumber = Math.floor(Math.random() * 6) + 1;  
    scoreDisplay.innerHTML = randomNumber;  
    // Update dice image display  
    // Update total score  
    // Check for win or lose condition  
};  
  
const switchPlayer = () => {  
    playerTurn = playerTurn === PLAYER.ONE ? PLAYER.TWO : PLAYER.ONE;  
    // Update UI to reflect player ↓
```

Understand what it is suggesting.

Code explanation and review (unknown code)

Activity

Choose one (or more) of your project code files from your repository and ask ChatGPT to refactor it.

Try the refactored code in VSCode and check that it works.

How has it improved your code?

What coding techniques does it use?

Be ready to share the differences between your code and ChatGPT's version.

PROMPT:

BR You

Please review the following code:

```
import streamlit as st
import os
import base64
import tempfile
from PIL import Image
import numpy as np
from moviepy.editor import VideoFileClip
import moviepyvideo.fx.all as vfx

## Session state ##
if 'clip_width' not in st.session_state:
    st.session_state.clip_width = 0
if 'clip_height' not in st.session_state:
    st.session_state.clip_height = 0
if 'clip_duration' not in st.session_state:
```

RESPONSE:



ChatGPT

The code you provided seems to be a Streamlit app for creating animated GIFs from uploaded video files. Here are some observations and suggestions for improvement:

1. **Session State Initialization:** You have initialized session state variables to default values at the beginning of your script. While this is fine, Streamlit provides a cleaner way to handle this using `st.session_state.get()` method with default values.
2. **File Uploader and Sidebar Layout:** The file uploader and sidebar layout are well-organized and intuitive. It's good to provide a link to download example files for users' reference.
3. **Displaying Parameters:** Displaying parameters such as video resolution, duration, FPS, etc., is informative and helpful for users to understand the input data.
4. **Image Preview:** The ability to preview a frame of the video is a nice feature. However, ensure that you're properly deleting temporary files after use, especially when generating multiple frames.
5. **Exporting GIF:** Generating GIFs from video clips seems to be functioning well. Ensure to handle

Could this code be refactored? (Better?)

Tips for improving code quality and efficiency

ChatGPT helps you find 'code smells' in almost any codebase – whatever the language.

- **To identify 'smelly code', first, prepare your code.**
- **Extract the relevant code snippets and ensure they're correctly formatted and easily readable.**
- **Break your code into smaller chunks to give the AI enough context to understand the code.**
- **Submit a clear and concise prompt.**
For instance, you could ask ChatGPT to identify code smells in the snippet or check if your code follows a particular SOLID object-oriented design principle.
- **Analyse ChatGPT's response.**
- **Ask ChatGPT to include code comments throughout the code it creates/refactors**

It can also write code documentation for you (if you ask it nicely in your prompts).

Practical Exercises

Brief 1: JS DOM hangman game

Create a HTML/CSS/JavaScript web app for a user to play the traditional game of hangman.

Here are the steps for ChatGPT to help create this web app:

- **Step 1 - Write the HTML markup**
- **Step 2 - Write the CSS classes**
- **Step 3 - Write the JavaScript code**
- **Step 4 - Get the images**
- **Step 5 - Check Functionality**



Brief 1: JS DOM hangman game

Step 1 - Write the HTML markup

In VSCode, create a new
'hangman' folder and a new **.html** file.

Copy ChatGPT's code into this file.

PROMPT:

You
I want to make a html css javascript game of hangman. Can you write the html page for me

RESPONSE:

ChatGPT
Certainly! Here's a simple HTML structure for a Hangman game:

```
html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Hangman Game</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Hangman Game</h1>
    <div id="hangman-display">
      <!-- Hangman image will be displayed here -->
    </div>
    <div id="word-display">
      <!-- Word to be guessed will be displayed here -->
    </div>
    <div id="letters">
      <!-- Alphabet buttons will be generated dynamically here -->
    </div>
    <div id="result">
      <!-- Game result message will be displayed here -->
    </div>
  </div>
</body>
</html>
```

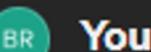
Brief 1: JS DOM hangman game

Step 2 - Write the CSS classes

Create a new **styles.css** file.

Copy ChatGPT's code into this file.

PROMPT:



You

can you create he styles.cs page ffor me

RESPONSE:



ChatGPT

Sure, here's a basic CSS styling for the Hangman game:

CSS

Copy code

```
body {  
    font-family: Arial, sans-serif;  
    margin: 0;  
    padding: 0;  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    height: 100vh;  
    background-color: #f0f0f0;  
}  
  
.container {  
    text-align: center;  
}  
  
h1 {  
    margin-bottom: 20px;  
}
```

Brief 1: JS DOM hangman game

Step 3 – Write the JavaScript code

Create a new `script.js` file.

Copy ChatGPT's code into this file.

PROMPT:

BR You

and finally can you write the javascript code for this

RESPONSE:

ChatGPT

Certainly! Here's a basic JavaScript code for implementing the Hangman game logic:

javascript

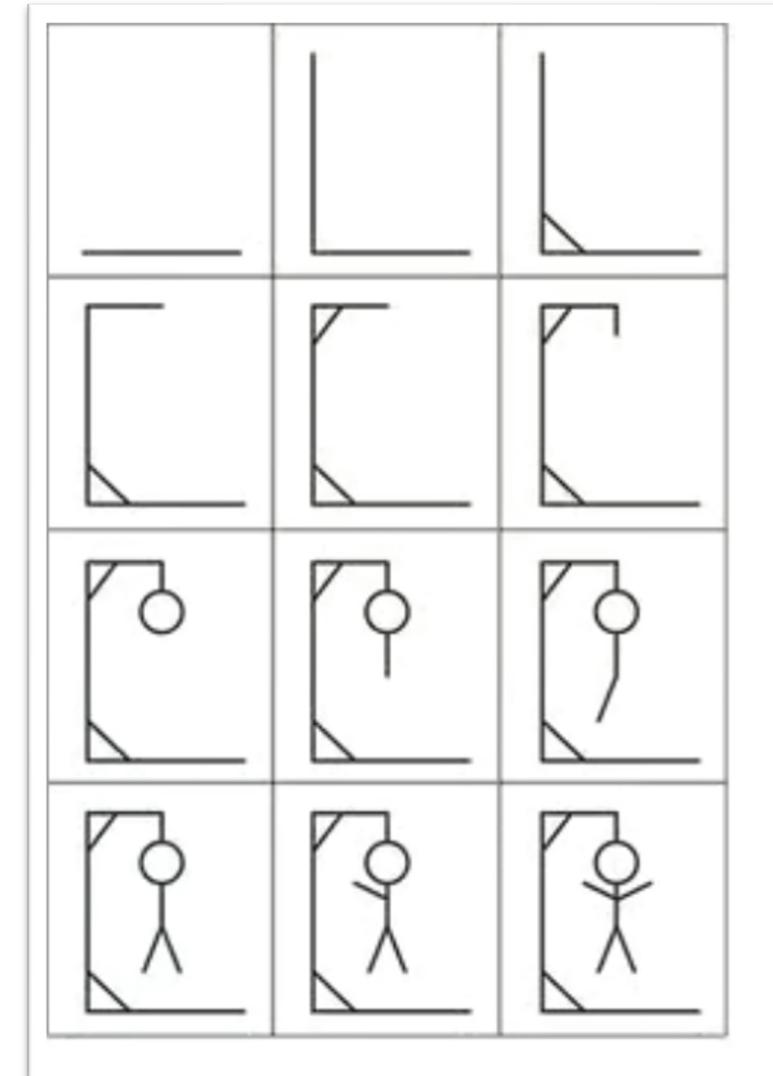
 Copy code

Brief 1: JS DOM hangman game

Step 4 - Get the images

A quick google search will assist you in finding the images you need for the game.

I found 12 images to represent each gallows/man stage the user will see upon a mistake.



Brief 1: JS DOM hangman game

Step 5 – Check functionality

Find the JavaScript code which displays each image according to the user's incorrect guesses.

```
22 // Function to display Hangman images based on incorrect guesses
23 function displayHangman() {
24     hangmanDisplay.innerHTML = ``;
25 }
26
```

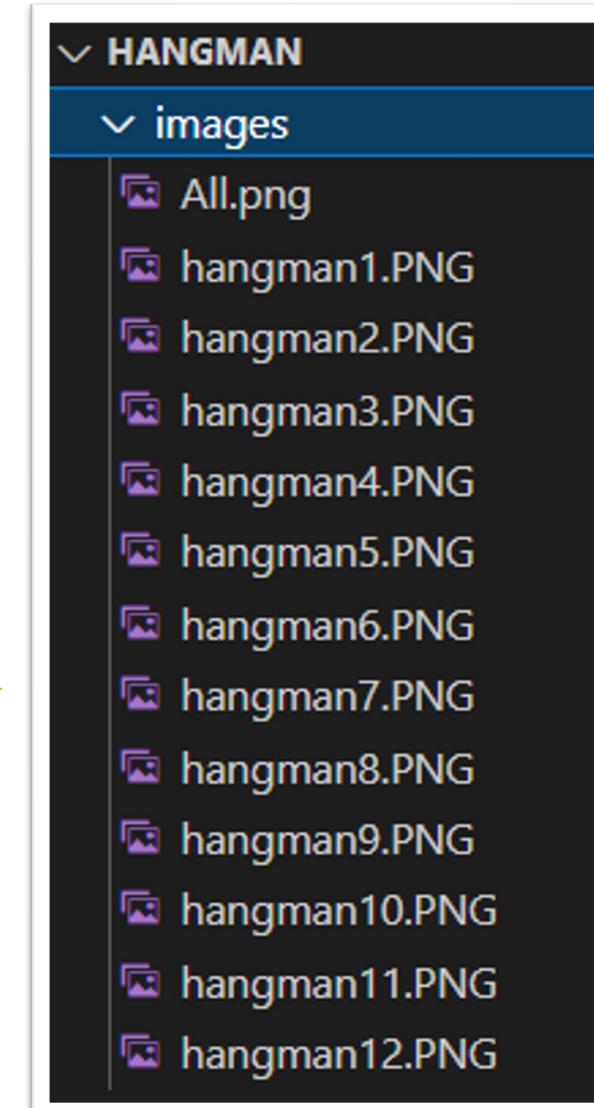


I placed the 12 images into a new folder called 'images' and renamed them to **hangman1**, **hangman2**, etc.

Brief 1: JS DOM hangman game

Step 5 - Check functionality

I placed the 12 images into a new folder called 'images' and renamed them to **hangman1**, **hangman2**, etc.



Brief 1: JS DOM hangman game

Step 5 - Check functionality

Find the JavaScript code which checks to see if the user has finally lost the game with a number of incorrect tries.

Change it to 12 to match the number of images we have.

Run the game!

```
39 // Function to check guessed letter
40 function checkGuess(letter) {
41     if (chosenWord.includes(letter.toLowerCase())) {
42         for (let i = 0; i < chosenWord.length; i++) {
43             if (chosenWord[i] === letter.toLowerCase()) {
44                 guessedWord[i] = letter.toLowerCase();
45             }
46         }
47         wordDisplay.textContent = guessedWord.join(" ");
48         if (!guessedWord.includes("_")) {
49             resultDisplay.textContent = "You Win!";
50             lettersContainer.innerHTML = "";
51         }
52     } else {
53         incorrectGuesses++;
54         displayHangman();
55         if (incorrectGuesses === 12) {
56             resultDisplay.textContent = "Game Over";
57             wordDisplay.textContent = chosenWord;
58             lettersContainer.innerHTML = "";
59         }
60     }
61 }
```

Brief 1: JS DOM hangman game

Run the game

The game worked as intended for both a **WINNING** and a **LOSING** scenario.

How could we make this game better?

Could we refine our prompts to add these new features in?

Spend **20 minutes** refining your ChatGPT prompts and implementing them into the Hangman Game.

Hangman Game



pro_ra__i__

A	B	C	D	E	F	G	H	I
J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	

Reset

Brief 2: JS DOM blackjack game

Create a HTML/CSS/JavaScript web app for a user to play Blackjack (or Pontoon 21).

Use these steps for ChatGPT to help create this web app:

- Step 1 - Write the HTML markup
- Step 2 - Write the CSS classes
- Step 3 - Write the JavaScript code
- Step 4 - Get the images
- Step 5 - Check Functionality

Remember to refine your prompts to add new features, etc.



Recap

Questions?

Learning Objectives

- ✓ Learn to build context, refine prompts and create a collaborative environment.
- ✓ Enable learners to effectively use AI Chatbots as a tool to seek assistance with programming tasks.