Color Changer JavaScript Study Guide:

JavaScript Code:

```
//Define all variables
const bodyBackground = document.getElementById('bodyBackground');
const colorButton = document.getElementById('colorOpt');
const hexButton = document.getElementById('hexOpt');
const submitButton = document.getElementById('submitButton');
const userInput = document.getElementById('userInput');
//define input type, variable set to “let” so that it can be changed later
let inputType = 'color';
//use arrow functions to make button clicks change inputType based on user input
colorButton.addEventListener('click', () => changeInputType('color'));
hexButton.addEventListener('click', () => changeInputType('hex'));
//uses the type selected by user to change the placeholder text
function changeInputType(type) {
    inputType = type;
    userInput.value = ' '; // clear input field
    userInput.placeholder = type === 'color' ? 'Enter Color Name' : 'Enter Hex Code';
};
//adds an event listener for submit to call the function changeBackground
submitButton.addEventListener('click', changeBackground);
//defines the changeBackground function
function changeBackground() {
//define new variable colorValue and assign it the value of the users input use let since
this variable will change based on user input
    let colorValue = userInput.value;
    // Validate hex code if inputType is 'hex'
    if (inputType === 'hex' && !colorValue.match(/^#(?:[0-9a-fA-F]{3}){1,2}$/)) {
        alert('Please enter a valid hex code.');
        return;
    }
//change the background color to the users input
    bodyBackground.style.backgroundColor = colorValue;
};
```

Study Guide:

Understanding the Document Object Model (DOM)

The DOM is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

Variables and getElementById

```
const bodyBackground = document.getElementById('bodyBackground');
const colorButton = document.getElementById('colorOpt');
const hexButton = document.getElementById('hexOpt');
const submitButton = document.getElementById('submitButton');
const userInput = document.getElementById('userInput');
```

document.getElementById: This method accesses an element from the HTML document using its id attribute. This is how you get a reference to various elements on your page so you can manipulate them with JavaScript.

const: This keyword is used to declare variables which cannot be reassigned. Here, it's used to store references to your HTML elements.

Event Listeners

```
colorButton.addEventListener('click', () => changeInputType('color'));
hexButton.addEventListener('click', () => changeInputType('hex'));
submitButton.addEventListener('click', changeBackground);
```

addEventListener: This method is used to attach an event handler to a specific element. It listens for events (like a click) on the element and executes a function when the event occurs.

Arrow functions (() => {}): This syntax is used for defining functions in a shorter way compared to the traditional function definition. Here, it's used to define what happens when an event is triggered.

Functions

```
function changeInputType(type) {
    inputType = type;
    userInput.value = '';
    userInput.placeholder = type === 'color' ? 'Enter Color Name' : 'Enter Hex Code';
};
```

Function Definition: This syntax (function functionName(parameters) {}) is used to define a reusable block of code. The changeInputType function changes the inputType and updates the userInput element's placeholder text based on the argument passed to it.

Let's break down the changeInputType function to understand its construction and purpose within the script. This function dynamically adjusts the behavior and presentation of the user input field based on the user's choice between entering a color name or a hex code.

Function Declaration

```
function changeInputType(type) {
    ...
}
```

function: This keyword is used to declare a function. Functions are one of the fundamental building blocks in JavaScript. A function is a set of statements that performs a task or calculates a value.

changeInputType: This is the name of the function. Naming functions clearly is crucial for readability and maintainability of your code.

type: Functions can take parameters. type is a parameter for this function, representing the type of input the user will provide ('color' or 'hex'). Parameters allow you to pass information or instructions into functions when you call them.

The use of "type" as the parameter in the changeInputType function is a design choice that provides flexibility and clarity in the function's purpose. Let's explore the reasons behind using "type" as the parameter:

Function Body

```
inputType = type;
userInput.value = '';
userInput.placeholder = type === 'color' ? 'Enter Color Name' : 'Enter Hex Code';
```

inputType = type;: This line assigns the value of the type parameter to the inputType variable. inputType is presumably a variable declared outside this function (in a broader scope) that tracks what kind of input the user is supposed to enter. By setting inputType to type, you're updating this tracking based on the user's selection.

function changeInputType(type): This line defines a new function named changeInputType that accepts one parameter, type. The purpose of type is to specify the kind of input the function should prepare for (e.g., a color name or a hex code).
Inside the Function

inputType = type;: The function starts by assigning the value of the parameter type to a variable named inputType. This line essentially says, "Set the current input type to whatever type was passed into the function." The inputType variable is likely defined outside this function, making it a global or higher scoped variable that keeps track of the current type of input expected from the user.

More about the variable type:

The type parameter in the changeInputType(type) function is defined as part of the function's declaration, meaning it's a variable that takes its value from whatever argument is passed to the function at the time it's called. This parameter doesn't have a predefined value within the function itself; rather, its value is determined by what is passed to the function when it's invoked elsewhere in your code.

How type Gets Its Value
When you call changeInputType, you provide an argument, and that argument becomes the value of type within the function. For example:

changeInputType('color');
In this call, 'color' is the argument passed to the function. When the function runs, type will have the value 'color', because 'color' is what was given to the function at the time of the call.

Similarly, if you call:

changeInputType('hex');
Now, type will have the value 'hex' for the duration of that function call.

Contextual Example
Looking at the broader context of your script, here's how type is used:

colorButton.addEventListener('click', () => changeInputType('color'));
hexButton.addEventListener('click', () => changeInputType('hex'));
For the colorButton, when it's clicked, the changeInputType function is called with the argument 'color'. So, inside changeInputType, type will be 'color'.

For the hexButton, when it's clicked, the function is called with 'hex' as the argument. Thus, within changeInputType, type will be 'hex'.

Understanding Parameters and Arguments
Parameter: A parameter is a named variable in a function definition. type is a parameter of changeInputType. It's like a placeholder for a value that will be provided when the function is called.

Argument: An argument is the actual value you provide to the function when you call it. In the examples above, 'color' and 'hex' are arguments that are passed to the function, and they take the place of the type parameter within the function's execution.

Summary
The script "knows" what type is because you tell it what to be each time you call changeInputType by providing an argument ('color' or 'hex'). The function then uses this information to perform its tasks, such as setting the input field's placeholder text appropriately.

userInput.value = '';: Here, userInput is expected to be a reference to an input element (obtained earlier with document.getElementById('userInput')). This line clears the current value of the input field by setting its value property to an empty string (''). It

ensures that whenever the input type is changed, the field starts fresh without any leftover input from the user.

userInput.placeholder = type === 'color' ? 'Enter Color Name' : 'Enter Hex Code';:

userInput.placeholder: The placeholder property of input elements is used to provide a hint to the user about what kind of information is expected in the field. This hint is displayed in the input field before the user enters a value.

type === 'color' ? 'Enter Color Name' : 'Enter Hex Code': This is a ternary operator, a shorthand for an if-else statement. It checks whether the type parameter equals the string 'color'. If true, it evaluates and returns 'Enter Color Name'; if false, it evaluates and returns 'Enter Hex Code'.

Condition: type === 'color' - This checks if the type parameter passed to the function is exactly equal to the string 'color'.

First Expression: 'Enter Color Name' - This is the value used if the condition is true.
Second Expression: 'Enter Hex Code' - This is the value used if the condition is false.

The Purpose of changeInputType

The function changeInputType is designed to dynamically adjust the user interface based on the type of color input the user wants to provide: a color name (like "red" or "blue") or a hexadecimal color code (like "#ff0000"). By changing the placeholder text of the input field, it gives the user a hint about what format their input should be in. Clearing the input field ensures that any previous inputs do not interfere with the new type of input expected.

This function exemplifies how functions can be used to modularize and manage behaviors in your web applications, making them more interactive and user-friendly.

```
function changeBackground() {
    let colorValue = userInput.value;
    if (inputType === 'hex' && !colorValue.match(/^#(?:[0-9a-fA-F]{3}){1,2}$/)) {
        alert('Please enter a valid hex code.');
        return;
    }
    bodyBackground.style.backgroundColor = colorValue;
};
```

Conditional Statements (if): Used to perform different actions based on different conditions. Here, it checks if the inputType is 'hex' and whether the input matches a hex code pattern.

Regular Expressions: The match method is used with a regular expression to check if the colorValue follows the hex color format.

Let's dissect the changeBackground function to understand its construction and how it works step by step. This function is responsible for changing the background color of the body of the webpage based on the user's input. It checks if the input is a valid hex code when the input type is set to 'hex'.

Function Declaration

```
function changeBackground() {
    ...
}
```

function: This keyword is used to declare a function in JavaScript. A function is essentially a block of code designed to perform a particular task.

changeBackground: This is the name of the function. The name is descriptive, indicating that the function's purpose is to change the background color of the webpage.

Function Body

```
let colorValue = userInput.value;
```

let colorValue = userInput.value;: This line declares a variable colorValue using let, which allows you to assign a value that can change over time. Here, it's initialized with the current value of the userInput element, which is expected to be an input field where the user can enter a color name or a hex code. The .value property gets the text that the user has entered into this input field.

```
if (inputType === 'hex' && !colorValue.match(/^#(?:[0-9a-fA-F]{3}){1,2}$/)) {
    alert('Please enter a valid hex code.');
    return;
}
```

Conditional Statement (if): This checks whether two conditions are true:
inputType === 'hex': This checks if the inputType variable equals the string 'hex'.
inputType is presumably a variable that tracks whether the user is expected to enter a color name or a hex code.

!colorValue.match(/^#(?:[0-9a-fA-F]{3}){1,2}$/): This uses the match method to check if colorValue does not match a regular expression pattern designed to validate hex color codes.

The regular expression ^#(?:[0-9a-fA-F]{3}){1,2}$ is used to match strings that represent valid hex color codes. Let's break it down:
^ asserts the start of the string.
# matches the literal hash symbol, which is how hex codes start.
(?:...) is a non-capturing group, used here for grouping without capturing the matched substring.
[0-9a-fA-F] matches any hexadecimal digit.
{3} means exactly 3 times, and {1,2} means once or twice, allowing for both short (e.g., #fff) and full (e.g., #ffffff) hex codes.
$ asserts the end of the string.
The ! operator negates the result of the .match method call, so the condition checks if colorValue does not match the pattern.

alert('Please enter a valid hex code.');: If the condition is true (meaning the input is supposed to be a hex code but isn't valid), an alert box is shown to the user with a message to enter a valid hex code.
return;: This immediately exits the function, preventing any further code from running if the input is invalid.

bodyBackground.style.backgroundColor = colorValue;
bodyBackground.style.backgroundColor = colorValue;: This line changes the background color of the webpage. bodyBackground is expected to be a reference to the <body> element of the webpage (or another element intended to have its background changed). .style accesses the inline style of the element, and .backgroundColor sets the background color CSS property to the value of colorValue. If colorValue is a valid color name or hex code, the background color of the page changes accordingly.

Summary
The changeBackground function reads the user's input, validates it if the expected input is a hex code, and changes the background color of the webpage to the inputted value if

it's valid. It showcases the use of conditionals, regular expressions for input validation, and direct manipulation of CSS styles through JavaScript to dynamically update the appearance of a webpage based on user interaction.

CSS Manipulation

bodyBackground.style.backgroundColor = colorValue;
Changing Styles: This line demonstrates how to modify the CSS of an element through JavaScript. It sets the background color of bodyBackground to whatever value is in colorValue.

Variable Scope and Declaration

let inputType = 'color';
let: This keyword declares a block-scoped local variable, optionally initializing it to a value. Here, inputType is used to keep track of whether the user is inputting a color name or a hex code.

Putting It All Together

The script listens for clicks on the color and hex buttons to switch the input mode between a color name and a hex code. It also listens for clicks on the submit button to change the background color based on the user's input. The script showcases how to interact with HTML elements using JavaScript, respond to user actions with event listeners, and dynamically update the webpage based on user input.

By understanding and combining these basic concepts, you can build a wide variety of interactive web pages and applications.