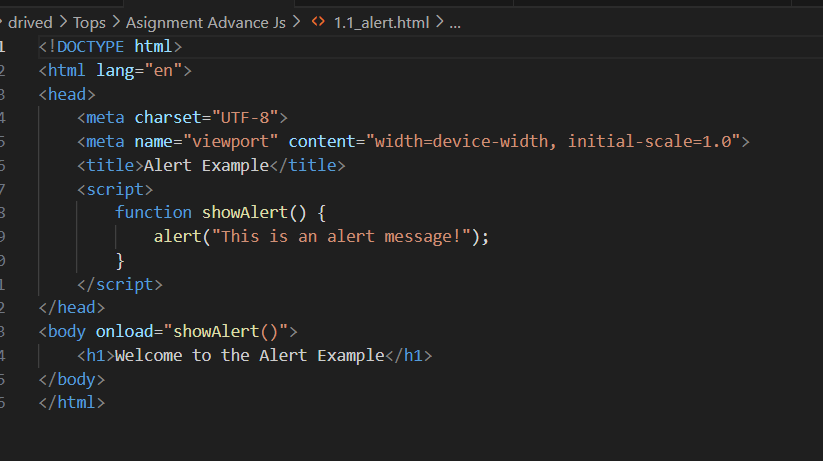
**Advance JavaScript for Front-End**

**Introduction and Code Quality**

● **Write a program to Show an alert**

****

● **What will be the result for these expressions?**

1. 5 > 4

Ans. This is true. Numerical comparison between two numbers. Since 5 is greater than 4.

2. "apple" > "pineapple"

This is false. String comparison is based on dictionary order. “apple” comes before “pineapple”.

3. "2" > "12"

This is true. String comparison is done lexicographically. Since “2” is compared to “1” first ,”2” is greater than “1”.

4. undefined == null

Ans. This is true. In JavaScript, `undefined` and `null` are loosely equal(==), meaning they are considered equal when using the loose equality operator.

5. undefined === null

Ans. This is false. The === operator checks for both value and type. Since undefined and null are of different type, the expression evaluates to false.

6. null == "\n0\n"

Ans. This is false. When using ==. Null is only equal to undefined and no other value. The string “\n0\n” does not coerce to null.

7. null === +"\n0\n"

Ans. This is false. The ===operator checks for both value and type. The unary + operator converts the string “\n0\n” the number 0. Since null is not strictly equal to the number 0,the expression evaluates to false.

● Will alert be shown?

if ("0") { alert( 'Hello'); }

● What is the code below going to output?

alert( null || 2 || undefined );

● The following function returns true if the parameter age is greater than 18. Otherwise it asks for a confirmation and returns its result:

Function

checkAge(age)

{

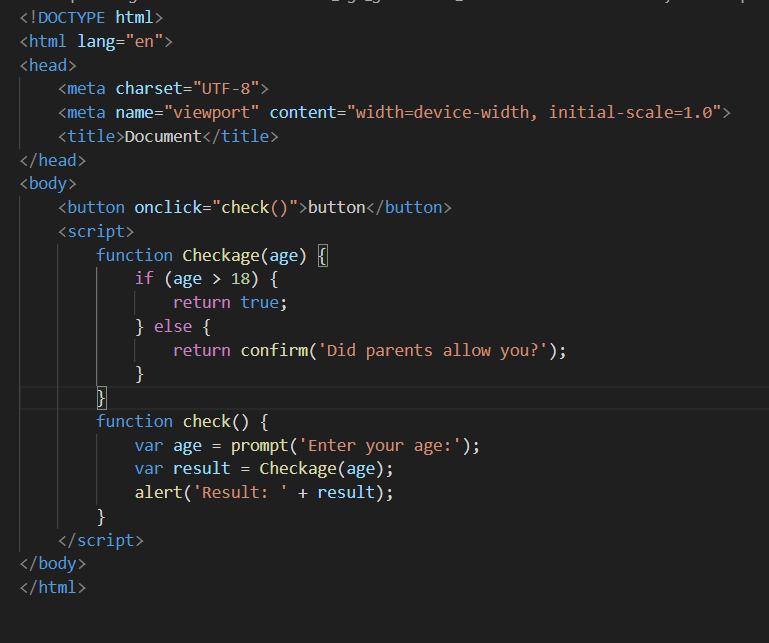
if (age> 18) {return true;}

else {

// ...return confirm (‘did parents allow you?');

}

}



● Replace Function Expressions with arrow functions in the code below:

Function

ask(question, yes, no)

{ if (confirm(question))yes();

Else

no();

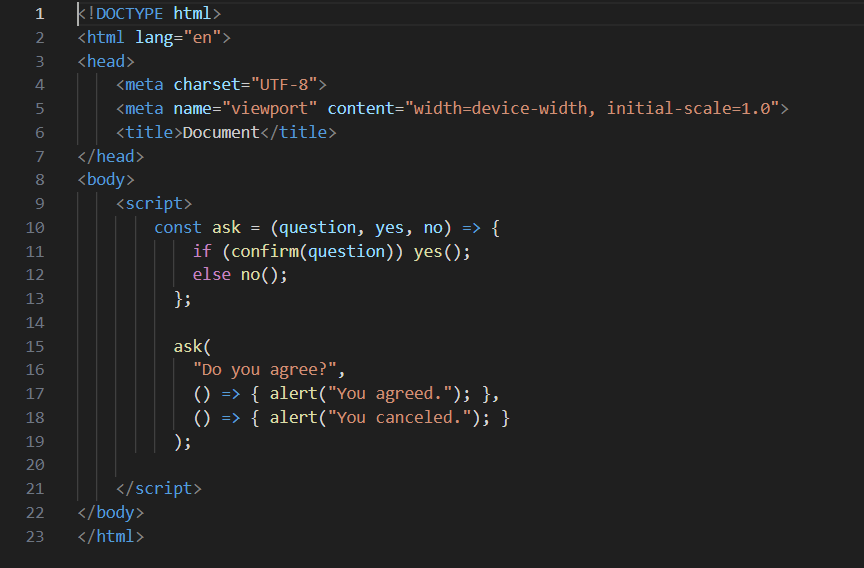
} ask("Do you agree?", function()

{ alert("You agreed."); },

function()

{ alert("You canceled the execution."); }

}



**Data Types and Objects**

● Write the code, one line for each action:

a) Create an empty object user.

Ans. let user = {}

b) Add the property name with the value John.

Ans. User.name = “John”;

c) Add the property surname with the value Smith.

Ans. user. Surname = “smith”;

d) Change the value of the name to Pete.

Ans. user.name = “Pete”;

e) Remove the property name from the object.

Ans. delete user.name;

● Is array copied?

let fruits = ["Apples", "Pear", "Orange"]; // push a new value into the "copy" let shoppingCart = fruits; shoppingCart.push("Banana"); // what's in fruits? alert( fruits.length ); // ?

● Map to names

let john = { name: "John", age: 25 }; let pete = { name: "Pete", age: 30 }; let mary = { name: "Mary", age: 28 }; let users = [ john, pete, mary ]; let names = /\* ... your code \*/ alert( names ); // John, Pete, Mary

● Map to objects

let john = { name: "John", surname: "Smith", id: 1 }; let pete = { name: "Pete", surname: "Hunt", id: 2 }; let mary = { name: "Mary", surname: "Key", id: 3 }; let users = [ john, pete, mary ]; let usersMapped = /\* ... your code ... \*/

/\*

usersMapped = [

{ fullName: "John Smith", id: 1 },

{ fullName: "Pete Hunt", id: 2 },

{ fullName: "Mary Key", id: 3 }

]

\*/ alert( usersMapped[0].id ) // 1 alert( usersMapped[0].fullName ) // John Smith

● Sum the properties There is a salaries object with arbitrary number of salaries. Write the function sumSalaries(salaries) that returns the sum of all salaries using Object.values and the for..of loop.If salaries is empty, then the result must be 0.

let salaries = {

"John": 100,

"Pete": 300,

"Mary": 250

};

alert( sumSalaries(salaries) ); // 650

● Destructuring assignment We have an object: Write the Destructuring assignment that reads: a) Name property into the variable name.

b) Year’s property into the variable age.

c) isAdmin property into the variable isAdmin (false, if no such property)

d) let user = { name: "John", years: 30};

● Turn the object into JSON and back Turn the user into JSON and then read it back into another variable.

user = { name: "John Smith", age: 35};

**Document, Event and Controls**

**● Create a program to hide/show the password**

**● Create a program that will select all the classes and loop over and whenever i click the button the alert should show**

**● Create a responsive header using proper JavaScript**

**● Create a form and validate using JavaScript**

**● Create a modal box using css and Js with three buttons**

**● Use external js library to show slider**

**● Prevent the browser when i click the form submit button**

**New Request**

**● What is JSON**

**JSON (JavaScript Object Notation) is a file format used in object-oriented programming that uses human-readable language, text, and syntax to store and communicate data objects between applications.**

**● What is promises**

In JavaScript, a Promise is an object that will produce a single value some time in the future. If the promise is successful, it will produce a resolved value, but if something goes wrong then it will produce a reason why the promise failed

**JavaScript Promises**used to simplify managing multiple asynchronous operations, preventing callback hell and unmanageable code. They represent future values, associating handlers with eventual success or failure, resembling synchronous methods by postponing value delivery until later.

● Write a program of promises and handle that promises also

● Use fetch method for calling an api <https://fakestoreapi.com/products>

● Display all the product from the api in your HTML page

**JavaScript Essentials**

**• Calculate subtotal price of quantity in JavaScript?**

**• What is JavaScript Output method?**

JavaScript Output Methods and PropertiesThere are a total of six ways to display text and/or content with JavaScript, depending on where you want to write it to; the first three are properties, while the other three are methods:

The **innerHTML**, **innerText** and **textContent** properties written into an HTML element.

The **document.write()** method writes directly into the HTML output.

The **window.alert()**method displays an alert box, suitable for displaying error messages and the like.

The **console.log()** method writes to the browser console, making it ideal for displaying debugging information.

The rest of this programming tutorial will delve into each of these methods and properties.

**• How to used JavaScript Output method?**

JavaScript provides several methods for outputting information, each serving different purposes depending on whether you want to display content in the browser, debug your code, or interact with the user. Below are examples and explanations of how to use each JavaScript output method effectively:

1. console.log()

The console.log() method is used to output messages to the web console. It's primarily used for debugging and displaying information during development.

**Example:**

javascript

Copy code

console.log("Hello, world!");

**Usage:**

To output variables, objects, or any other JavaScript data to the console.

Useful for inspecting values, debugging code logic, or tracking the flow of execution.

2. alert ()

The alert () method displays a dialog box with an optional message and an OK button. It's commonly used for showing alerts or notifications to the user.

**Example:**

JavaScript

Copy code

Alert ("Hello, world!");

**Usage:**

To provide information or warnings to the user.

Not recommended for extensive use as it blocks the JavaScript execution until the user clicks OK.

3. document.write()

The document.write() method writes HTML expressions or JavaScript code to a document. It's mainly used for testing and dynamically adding content to a webpage.

**Example:**

JavaScript

Copy code

Document.write("Hello, world!");

**Usage:**

To dynamically add content to a webpage at the time of page load or script execution.

Should be used cautiously as it can overwrite existing content if used after the page has loaded.

**4. Modifying HTML Content**

You can directly modify HTML content using JavaScript by accessing elements and updating their properties like innerHTML, textContent, or innerText.

**Example using innerHTML:**

JavaScript

Copy code

document.getElementById("myElement").innerHTML = "Hello, world!";

**Usage:**

To update the content of specific HTML elements dynamically based on user interaction or data retrieval.

Provides flexibility in manipulating the document structure and content.

5. console.dir()

The console.dir() method outputs a collapsible list of the properties of a specified JavaScript object in the console. It's useful for inspecting complex objects or arrays.

**Example:**

javascript

Copy code

const obj = { name: "John", age: 30 };

console.dir(obj);

**Usage:**

* To explore the structure and properties of JavaScript objects or arrays.
* Provides a detailed view of nested objects and their properties.

**• How to used JavaScript Events to do all examples?**

JavaScript events are actions or occurrences that happen in the browser, such as a user clicking a button, hovering over an element, or submitting a form. You can use these events to trigger specific JavaScript functions. Below, I will demonstrate how to use JavaScript events to do the following examples:

1. Output a message to the console when a button is clicked.
2. Show an alert when a button is clicked.
3. Write content to the document when a button is clicked.
4. Change the content of an HTML element when a button is clicked.
5. Log object properties to the console when a button is clicked.
6. Log different types of messages (error, warning, info) when buttons are clicked.