JavaScript Short Notes

Purpose of JavaScript in Front-end Development:

1 . JavaScript is mainly used for adding interactivity to websites, running directly in web browsers like Chrome or Firefox :

Here are some simple examples of interactive things:

1. **Interactive Website**: Imagine you're on a shopping website. You can click on pictures of products, add them to your cart, and enter your details to buy. The website responds to your actions, like showing you a new page when you click a button.
2. **Interactive Game**: Think of a mobile game where you tap the screen to make a character jump or avoid obstacles. The game changes depending on what you do, like scoring points when you avoid something.
3. **Interactive Learning Tool**: On an online quiz, you click on answers, and the website immediately shows if you were correct or wrong. It changes based on your choices.

In all these examples, you're interacting with the system by making choices, and the system responds to you, which makes it **intera1.ctive**.

2. JavaScript is Used for Client-Side Validation.

How JavaScript Performs Client-Side Validation:

1. **Instant Feedback**: JavaScript checks the data entered in forms (like email or password) before it is submitted to the server.
2. **Checks Data Format**: It makes sure the entered data follows specific rules, e.g., checking if an email address contains "@" or if a password is strong enough.
3. **User Input**: If the data is incorrect or missing, JavaScript shows an error message immediately, without the need to refresh the page or send data to the server.

Advantages of Using JavaScript for Client-Side Validation:

1. **Faster**: Provides instant feedback to users without waiting for a server response.
2. **Less Server Load**: Reduces unnecessary requests to the server by catching errors early on the client side.
3. **Better User Experience**: Users can correct errors right away, leading to a smoother interaction.

What Happens if JavaScript is Not Used for Client-Side Validation:

1. **Slower User Experience**: Users would have to submit the form, wait for the server to respond, and then see an error. This delay could be frustrating.
2. **More Server Load**: Every time a user submits incorrect data, it sends a request to the server, adding extra processing and load.
3. **More Errors**: Users might make more mistakes without immediate feedback, resulting in multiple attempts to submit the form.

Short Answer for Interviews:

JavaScript helps validate data on the user's side, making the process faster and more efficient, while avoiding delays and extra server load. Without it, users face slower, less responsive forms and increased server stress.

Want to learn more ?

- However, with special tools, JavaScript can also be used to build standalone applications, which are independent programs that don’t require a browser to run.
- With the help of **frameworks** like **Electron** and **React Native**, JavaScript can also create standalone apps.

What is Framework and Library ?

- **Library**: A library is a collection of pre-written functions or routines that a programmer can call to perform specific tasks.
- **Framework**: A framework provides a structure and set of guidelines for building an application. Frameworks are Less flexible because it dictates the overall structure and flow of your application. You are required to follow the framework's conventions and structure.

What is Scripting Language?

JavaScript is a **scripting language** because it's used to write small scripts that help in making web pages interactive and dynamic, typically by being executed directly in the browser.

JavaScript code is **interpreted** directly by the browser, meaning it doesn't need to be compiled into a separate program. It runs **line by line** when a web page is loaded or an action occurs.

---

1. Internal JavaScript

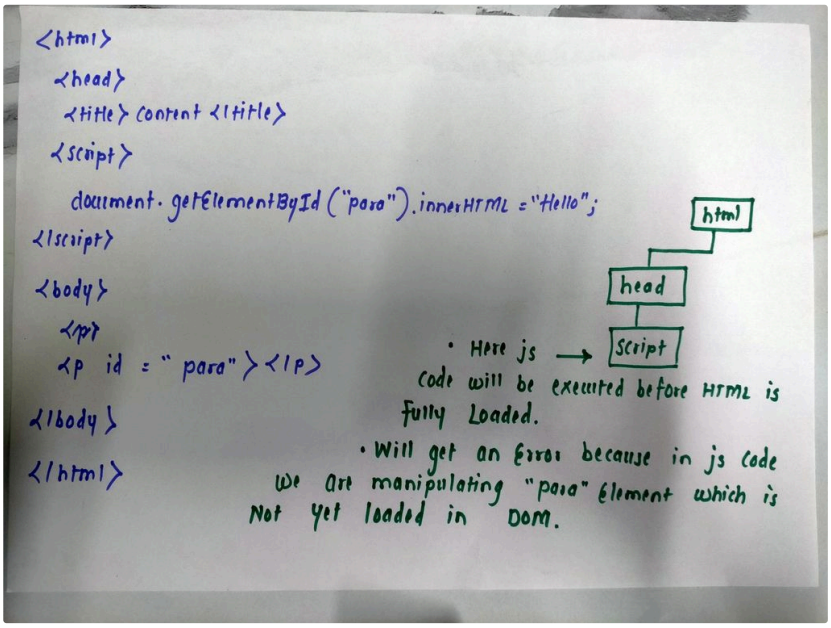
Internal JavaScript is written directly within an HTML file, typically inside a `<script>` tag in the `<head>` or `<body>` section.

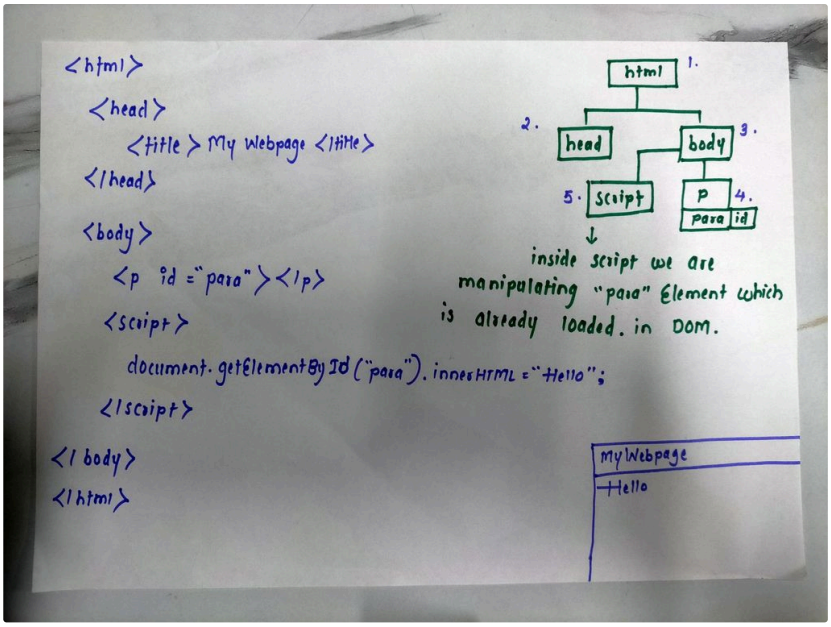
How it works:

- You place JavaScript code within `<script></script>` tags in the HTML file.
- The JavaScript is executed when the HTML document is loaded.

How JS Code will be Executed?

1. **HTML Loads First**: When you load a webpage, the browser first reads and loads the **HTML** file. This HTML file describes the structure of the page, such as headings, paragraphs, images, and links.
2. **Browser Encounters the `<script>` Tag**: When the browser reaches a `<script>` tag in the HTML, it knows that JavaScript code is present. The script can either be embedded directly within the HTML or referenced as an external file (e.g., `<script src="script.js"></script>`).
3. **JavaScript Engine is Invoked**: Once the browser encounters the `<script>` tag, it calls the **JavaScript engine** (like V8 in Chrome, SpiderMonkey in Firefox) to start executing the JavaScript code.
4. **JavaScript Code is Executed Line by Line**: The JavaScript engine reads the code and executes it **line by line**. The engine performs tasks like:
   - Manipulating the HTML elements (DOM)
   - Handling events (like button clicks)
   - Validating data (form validation)
   - Updating the page dynamically (like showing or hiding elements)
5. **Execution Happens While the Page is Loaded**: JavaScript is often executed while the page is still loaded, meaning the user can interact with the page while the script is running.



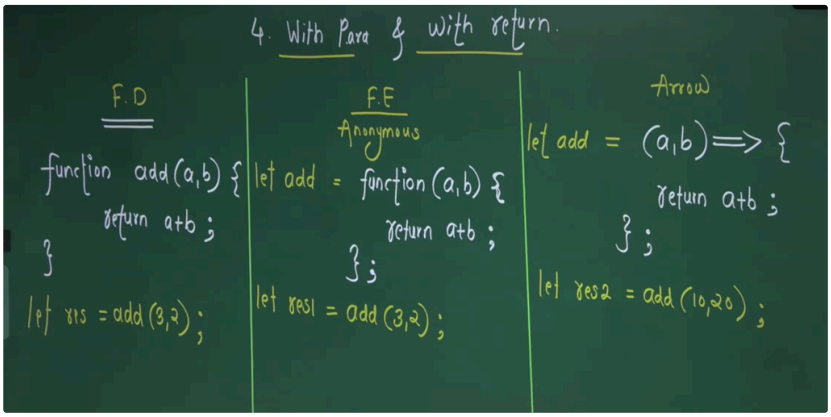


External JS:

```
<script src="script.js"></script>
```

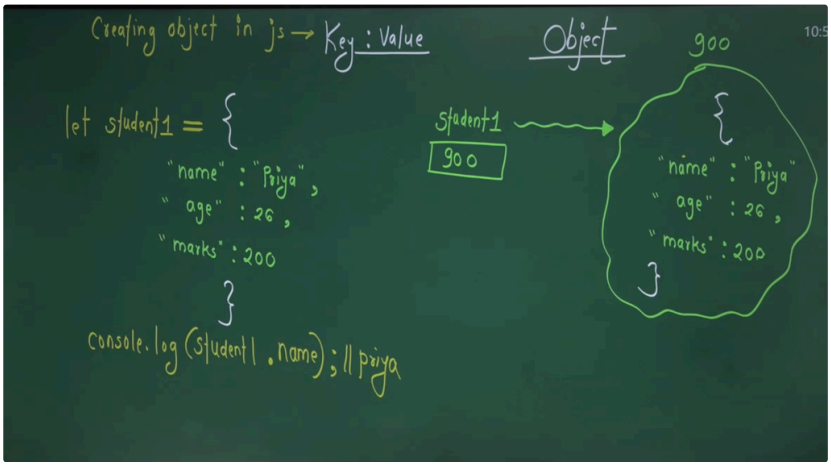
Data Types:

```
<script>
       console.log(typeof 11); //number
       console.log(typeof 11.55); //number
       console.log(typeof 'Kodnest'); //String
       console.log(typeof true); //boolean
       console.log(typeof null); //object
       console.log(typeof undefined); //undefined

       let a;
       console.log( typeof a);//undefined

       const dict = {
        'name':'Priya',
        'age':26,
        'Phone':8767898766,
        'Total_marks':function(arr){
            return arr.reduce((acc,num)=>acc+num,0);
        }
       }

       console.log(typeof dict) //object
       console.log(dict.Total_marks([10,20,30]));

       let fun=function(a,b){
        return a+b;
       }

       console.log(typeof fun)
       fun.color='blue';
       console.log(fun.color); //blue


   </script>
```

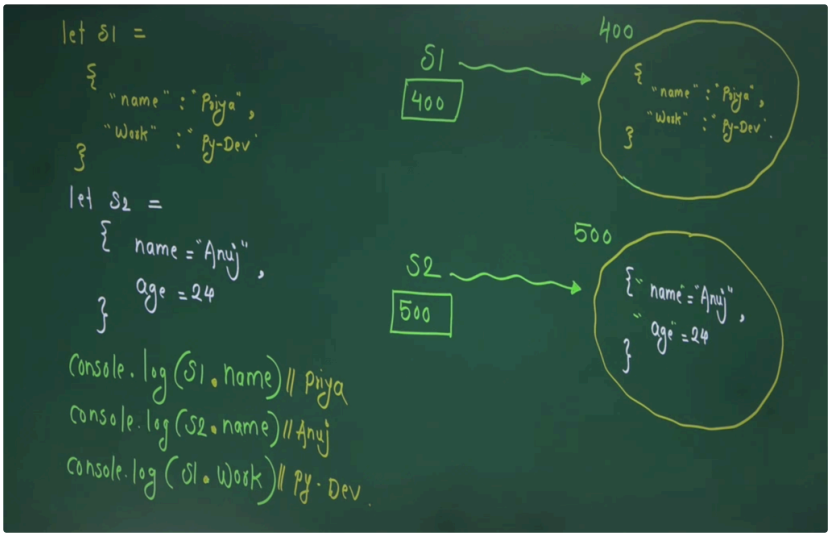


Objects in JavaScript :

JSON (Key:Value Pair) Way of Object Creation:



Memory Allocation for Objects:



class- 19-Nov

JSON- JavaScript Object Notation Example:

```
<script>

   // JSON- JavaScript Object Notation:
   /*
   Keys will be of string data type but their values can be
   of any data type.
   */

   let s1= {
    'age':22,
    "price":45.89,
    "sal":null,
    'name':"Priya",
    'ismariied':true,
    'work_details': {'DMI':'3 years','Kodnest':'2 Years'},
    'marks':[10,20,30,40],
    'skills':function(){console.log('Java,Py,SQL')}
   }
   console.log(s1, typeof s1)
   console.log(s1.marks);//accessing array object data type
   console.log(s1.marks[1]); //20
   s1.skills();//calling the function object data type


</script>
```

Declaring Array Object

```
<script>
    // Declaring an Array Object:
    let arr1=[10,20,30,40,50];
    console.log(arr1);

    let arr2=[22,22.66,true,"Priya"];
    console.log(arr2);

</script>
<!--
1.In an Array we can Store Homogeneous Type of Data.
2.In an Array we can store Heterogeneous Type of Data.-->
```

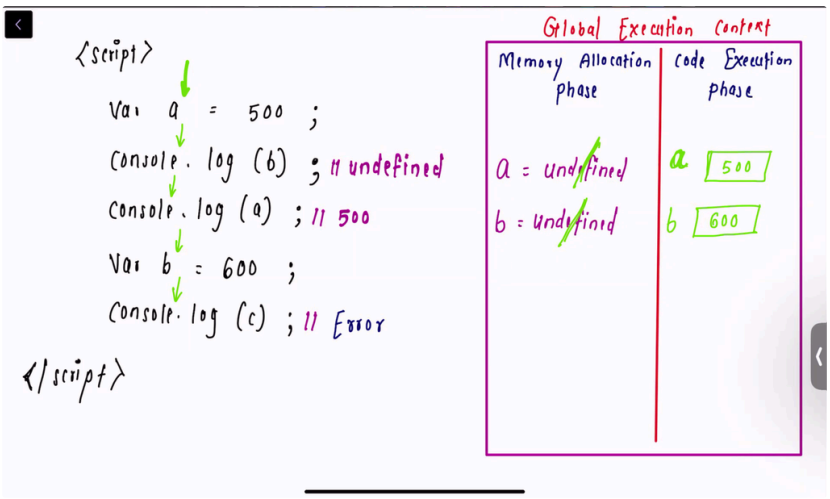
Array Built-in Functions:

```
<script>
    arr1 = [10,20,30];
    console.log(arr1);//[10,20,30]

    arr1.push(40);
    console.log(arr1);//[10,20,30,40]

    arr1.pop();
    console.log(arr1);//[10,20,30]

    arr1.shift();
    console.log(arr1);//[20,30]

    arr1.unshift(100);
    console.log(arr1);//[100,20,30]

    //splice(start,deleteCount,Items to be added)
    numbers = [100,200,300,400]
    new_array = numbers.splice(1,2,600,700);
    console.log(new_array);// [200, 300]
    console.log(numbers);// [100, 600, 700, 400]

```

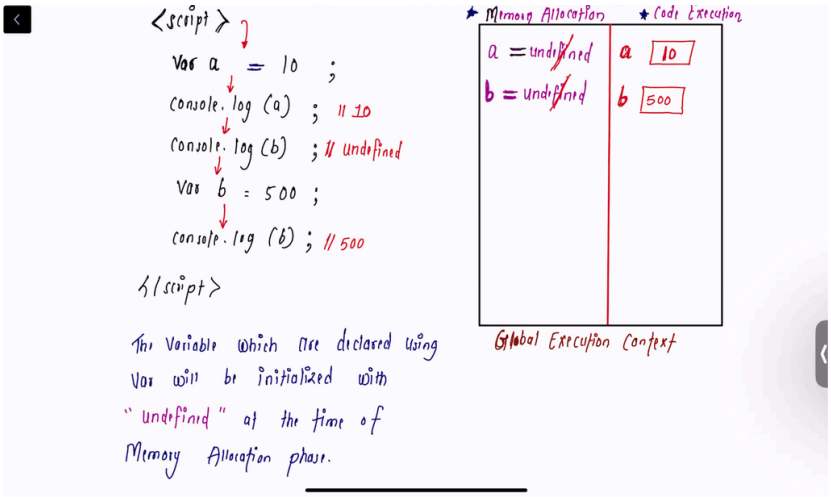
```
    //slice(start, end):
    let num1 = [900,800,700,600,500];
    let num2 = num1.slice(1,4);
    console.log("Value of num1",num1);
    console.log("Value of num2",num2);//[800, 700, 600]

    //indexOf():
    console.log(num1.indexOf(600));//3

    //includes():
    console.log(num1.includes(5000));//false

    //sort():
    num1.sort();
    console.log(num1);//[500, 600, 700, 800, 900]

    //reverse();
    num1.reverse();
    console.log(num1); // [900, 800, 700, 600, 500]


</script>
```

Hoisting:

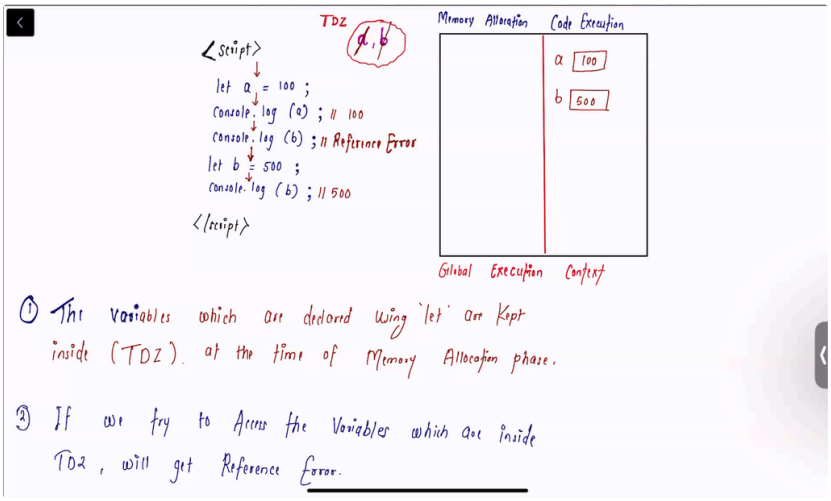
Example-1 : Date: 25th Nov Session



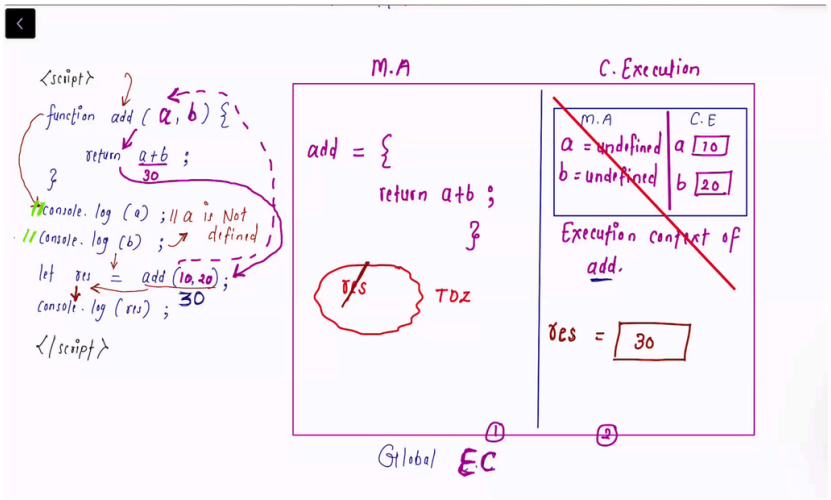
Example-02

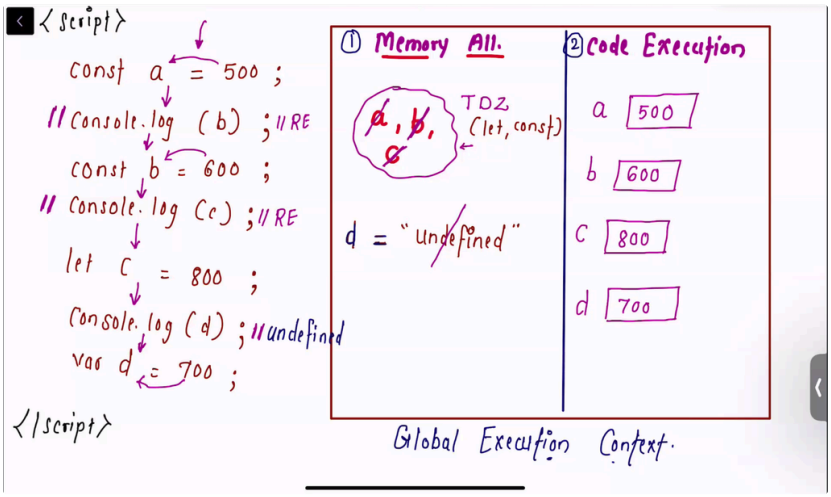


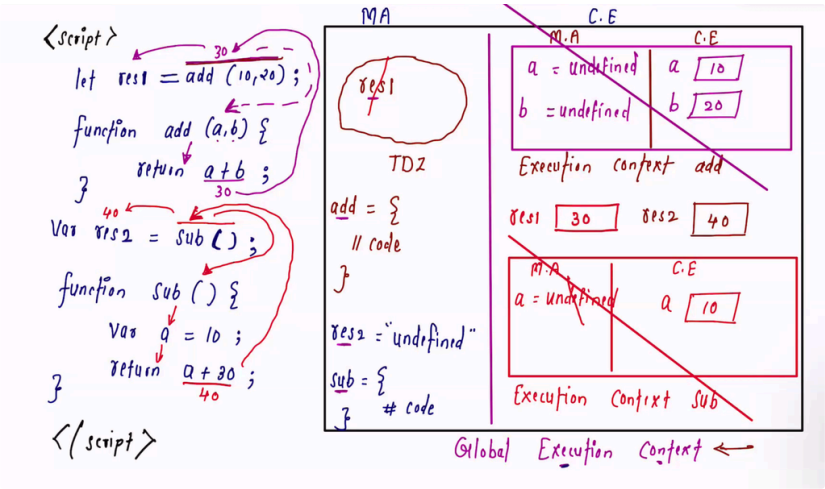
Example:03

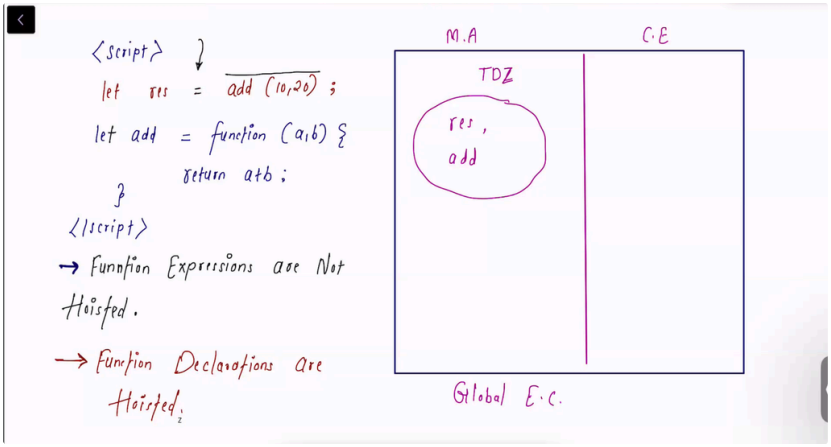


26 NOV









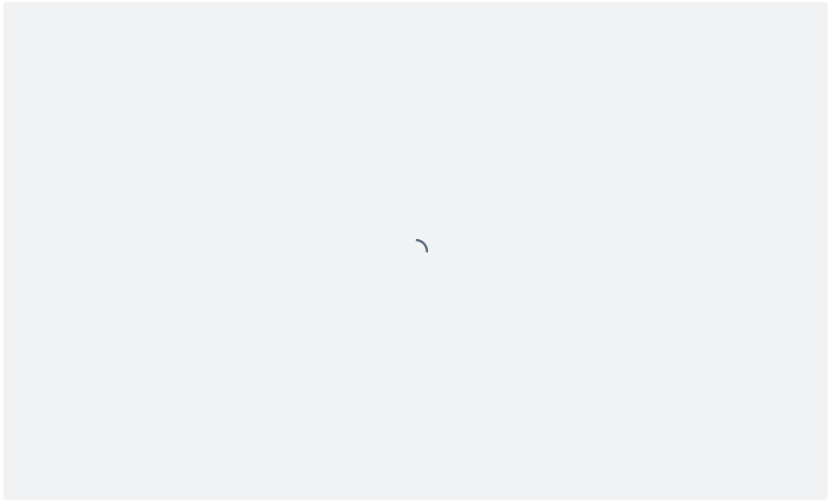
arguments object and spread operator:

```
<script>
  //argumets object:
/*
  function add(){
    console.log(arguments.length);
    console.log(arguments);
    for(i of arguments){
        console.log(i);
    }
  }

  add(10,20,30,40);
  */

  function add(){
    if (arguments.length === 2){
    return arguments[0]+arguments[1];
  }
  else if (arguments.length === 3){
    return arguments[0]+arguments[1]+arguments[2];
  }

  else {
    return "Not valid Parameters...send either 2 or 3"
```

```
  }
}

let res =add(10,20,30,40);
console.log(res);


let mul = (...args) => {
    console.log(args.length);
    console.log("Result is",args[0] * args[1]);
}

mul(1,2);
</script>
```

Function overloading Examples with diagrams:

Example:



Example 2:

Achieving Function Overloading using spread operator:

JavaScript does not support function overloading like Java or C++, where multiple functions can have the same name with different parameter lists. In JavaScript, if you define multiple functions with the same name, the last one will overwrite the previous ones.

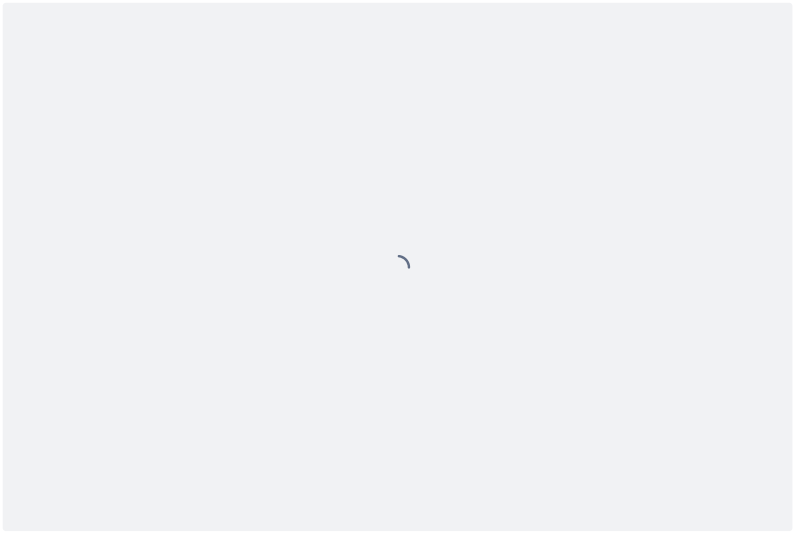
Example: In this check arguments wont support reduce().

spread operator can be used with all functions:

Practice Example of arguments object and spread operator:

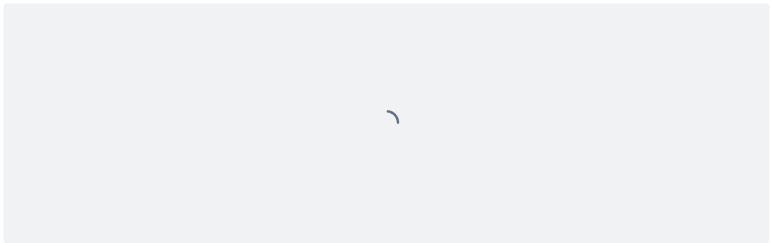
{ Use script Tag while writing the code in html file }

1.



2.

3.



Lexical Environement: