📄 Session Notes – JavaScript Functions & Array with Objects (23/09/2025)

---

1. JavaScript Functions (Basics)

A **function** is a reusable block of code designed to perform a specific task.

👉 Syntax:

```
js

function functionName(parameters) {
    // code to be executed
    return value;
}
```

👉 Example:

```
js

function greet(name) {
    return "Hello " + name;
}
console.log(greet("Sky")); // Hello Sky
```

🔑 Why Functions?

- Avoids repetition of code
- Increases modularity & readability
- Reusability across the project

---

2. Classic Functions vs Modern Arrow Functions

Classic JS Functions

- Declared using the `function` keyword.
- Have their own `this` context.
- Can be hoisted.

👉 Example:

```js
function add(a, b) {
    return a + b;
}
console.log(add(5, 3)); // 8
```

Modern Arrow Functions (ES6)

- Introduced in ES6.
- Shorter syntax.
- Do not have their own `this` (they inherit from parent scope).
- Cannot be hoisted.

👉 Example:

```js
const add = (a, b) => a + b;
console.log(add(5, 3)); // 8
```

Comparison

Feature	Classic Function	Arrow Function
Syntax	Longer	Concise
`this` binding	Own `this`	Inherits `this`
Hoisting	Yes	No
Usage	General purpose	Callbacks, functional programming

---

3. Pushing Objects into an Array (Database Scenario)

In real-world projects, we store **data as objects inside arrays** (similar to a database).

👉 Example:

```js
let studentsDB = [];

function addStudent(name, age, course) {
    const student = {
        name: name,
        age: age,
        course: course
    };
    studentsDB.push(student);
}

addStudent("Rajesh", 22, "MERN Fullstack");
addStudent("Sai", 21, "UI/UX");
console.log(studentsDB);
```

✅ Output:

```js
[
  { name: 'Rajesh', age: 22, course: 'MERN Fullstack' },
  { name: 'Sai', age: 21, course: 'UI/UX' }
]
```

📝 Assignment

☞ Write a program to create a **students database** using **functions only**:

1. Define a function to accept student details (`name, age, course`).
2. Convert the arguments into an object.
3. Push that object into a `studentsDB` array.
4. Add multiple students by calling the function.
5. Print the final database.

⚡ *Hint:* Use **functions, objects, and arrays** only.

🎯 Interview Prep Questions – Functions & Objects in JS

---

1. Core Function Concepts

1. What is a function in JavaScript? Why are functions important?
2. Explain the difference between **function declaration** and **function expression**.
3. What is **hoisting** in functions? Provide an example.
4. What is the difference between **return** and **console.log()** in a function?
5. Can a function return another function in JavaScript? Give an example.

---

2. Classic Functions vs Arrow Functions

6. Write a function to add two numbers in both classic and arrow function style.
7. What are the **key differences** between arrow functions and traditional functions?
8. Explain how `this` works differently in **arrow functions** vs **classic functions**.
9. Why can't arrow functions be used as constructors (`new` keyword)?
10. Are arrow functions hoisted? Why or why not?

---

3. Objects in Arrays (Database Scenarios)

11. How do you store multiple objects inside an array? Give an example.
12. Write a function that creates a **student object** and pushes it into an array.

13. Explain the difference between **copy by value** and **copy by reference** in JavaScript with examples.
14. If two objects have the same values, will they be equal (== or ===)? Why not?
15. How would you update a specific student's data inside an array of student objects?

---

4. Practical Coding Challenges

16. Write a function `addStudent(name, age, course)` that pushes student objects into a `studentsDB` array.
17. Write a function that takes an array of student objects and returns the **names only**.
18. Write a function that counts how many students are enrolled in "MERN Fullstack".
19. Write a function that updates a student's age by passing `name` as an argument.
20. Write a function to remove a student from the database using their `name`.