# JavaScript Advanced #RKIT Dev Sheet

## 1. Session Storage & LocalStorage

**Definition:** Both are part of Web Storage API used to store data in the browser.

**Session Storage:**- Stores data for a single session (data is lost after the browser/tab is closed).  
- Use Case: Temporary data like a shopping cart in an e-commerce site until checkout.  
- Example:

sessionStorage.setItem('name', 'Alice');  
console.log(sessionStorage.getItem('name')); // Output: Alice

**LocalStorage:**- Stores data with no expiration date (persists even after the browser is closed).  
- Use Case: User preferences like theme color or language settings.  
- Example:

localStorage.setItem('theme', 'dark');  
console.log(localStorage.getItem('theme')); // Output: dark

**Difference:** LocalStorage persists data longer, while Session Storage is session-based.

## 2. Basics of Cookies

**Definition:** Small text files stored in the browser, often used for tracking and maintaining session information.

**How Cookies Work:**- Set by server or JavaScript, sent with every HTTP request.  
- Contains attributes like expires, path, domain, secure, etc.  
- Use Case: Authentication (keeping users logged in), tracking (e.g., analytics).  
- Example:

document.cookie = 'username=John; expires=Thu, 18 Dec 2024 12:00:00 UTC; path=/';

**Difference from Web Storage:** Cookies are sent to the server with requests, whereas Web Storage data stays on the client.

## 3. Browser Debugging

**What it means:** Tools in browsers that help inspect, debug, and improve website performance.

**Inspect Element Window:**- A tool that allows you to view and modify the HTML and CSS of a webpage in real time.  
- Useful for testing visual changes without altering the original code.  
- Located in DevTools, under the Elements tab.

**Different Tabs in Inspect Element Window:**- Elements Tab: View and modify HTML & CSS.  
- Console Tab: Log outputs, errors, and test JavaScript code snippets.  
- Sources Tab: Inspect JavaScript files and set breakpoints.  
- Network Tab: Monitor HTTP requests and responses, analyze load times.  
- Application Tab: Manage storage (LocalStorage, Session Storage, Cookies).  
- Performance Tab: Record and analyze website performance to identify bottlenecks.

**Caching:**- A technique to store frequently accessed data in the browser for faster loading.  
- Cached resources include images, stylesheets, and scripts, reducing server requests.  
- Inspect caching in the Application Tab under Cache Storage in DevTools.  
- Useful for optimizing page load times and improving user experience.

## 4. Object-Oriented JavaScript (OOJS) Study

**What is OOJS?**- Object-Oriented JavaScript (OOJS) is a programming where objects are used to model real-world entities.  
- This approach organizes code into objects with properties and methods, making code modular, reusable, and easier to maintain.  
- Key concepts include classes, inheritance, encapsulation, and polymorphism.

**Possible Ways to Implement Classes:**- JavaScript ES6 introduced the 'class' keyword for defining classes in a cleaner syntax.  
- Pre-ES6 Classes:  
 - Using constructor functions with prototype methods.  
- ES6+ Classes:  
 - Use the 'class' keyword to define classes directly.  
 - Supports inheritance with 'extends' and superclasses.

**Example of ES6 Class:**class Car {  
 constructor(brand) {  
 this.brand = brand;  
 }  
 getBrand() {  
 return this.brand;  
 }  
}  
const myCar = new Car('Toyota');  
console.log(myCar.getBrand()); // Output: Toyota

**Static Class and Properties Declaration:**- Static Properties:  
 - Declared with the 'static' keyword, accessed on the class itself (not on instances).  
 - Example: Useful for counters or utility methods.  
- Property Declarations (ESNext):  
 - Public Fields: Declared directly in the class without 'this'.  
 - Private Fields: Declared with '#' prefix, accessible only within the class.

**Example of Static and Property Declarations:**class Counter {  
 static count = 0;  
 #secret = 'hidden';  
 constructor() {  
 Counter.count++;  
 }  
 getSecret() {  
 return this.#secret;  
 }  
}  
const instance = new Counter();  
console.log(Counter.count); // Output: 1  
console.log(instance.getSecret()); // Output: hidden

# ES6 JavaScript

## Variable Declarations: let and const

In ES6, 'let' and 'const' are used for variable declarations. 'let' allows reassigning, while 'const' does not.

Example:

// Using let  
let x = 10;  
x = 20; // Allowed  
  
// Using const  
const y = 15;  
y = 25; // Error: Cannot reassign

## Arrow Functions

Arrow functions provide a concise syntax for writing functions, and they do not bind their own 'this'.

Example:

// Traditional function  
function add(a, b) { return a + b; }  
  
// Arrow function  
const add = (a, b) => a + b;

## Template Literals

Template literals allow embedding variables and expressions inside strings using backticks (`) and ${}.

Example:

const name = 'Priyansh';  
console.log(`Hello, ${name}!`); // Output: Hello, Priyansh!

## Destructuring Assignment

Destructuring allows you to unpack values from arrays or properties from objects into distinct variables.

Example:

// Array destructuring  
const [a, b] = [1, 2];  
  
// Object destructuring  
const person = { name: 'Alice', age: 25 };  
const { name, age } = person;

## Default Parameters

Default parameters allow you to set default values for function parameters.

Example:

function greet(name = 'Guest') {  
 console.log(`Hello, ${name}!`);  
}  
  
greet(); // Output: Hello, Guest!  
greet('Priyansh'); // Output: Hello, Priyansh!

## Spread Operator

The spread operator (...) allows expanding an iterable (like an array) into individual elements.

Example:

const arr1 = [1, 2, 3];  
const arr2 = [...arr1, 4, 5]; // [1, 2, 3, 4, 5]

## Classes

ES6 introduced classes as a blueprint for creating objects, with constructor functions and methods.

Example:

class Person {  
 constructor(name) {  
 this.name = name;  
 }  
 greet() {  
 console.log(`Hello, ${this.name}!`);  
 }  
}  
  
const person = new Person('Priyansh');  
person.greet(); // Output: Hello, Priyansh!