নিচে একটি Lead Software Engineer (JavaScript) ইন্টারভিউয়ের জন্য গুরুত্বপূর্ণ প্রশ্ন ও উত্তর দেওয়া হলো — বাংলায় ব্যাখ্যাসহ। এতে আপনি আপনার প্রস্তুতি আরও ভালোভাবে নিতে পারবেন।

**✅ প্রশ্ন ১: JavaScript এর Execution Context কী?**

**উত্তর (বাংলা):**  
JavaScript এ Execution Context মানে হলো একটি পরিবেশ (environment) যেখানে একটি নির্দিষ্ট কোড চলবে। এটি তিন ধরনের হয়:

1. **Global Execution Context (GEC)**
2. **Function Execution Context (FEC)**
3. **Eval Execution Context (কমন নয়)**

প্রথমে Global Context তৈরি হয়। এরপর যদি কোনো ফাংশন কল করা হয়, তখন Function Execution Context তৈরি হয়। প্রতিটা Execution Context তৈরি হয় দুটি ধাপে:

* Creation Phase
* Execution Phase

**✅ প্রশ্ন ২: Closure কী এবং কিভাবে কাজ করে?**

**উত্তর (বাংলা):**  
Closure হলো একটি ফাংশন যেটি তার বাইরের স্কোপ থেকে ভ্যারিয়েবল অ্যাক্সেস করতে পারে, এমনকি সেই বাইরের স্কোপ ফাংশনটি শেষ হয়ে গেলেও।

function outer() {

let count = 0;

return function inner() {

count++;

console.log(count);

}

}

const counter = outer();

counter(); // 1

counter(); // 2

এখানে inner() হলো এক ধরনের Closure কারণ এটি count ভ্যারিয়েবল ব্যবহার করছে যেটি outer() স্কোপে ছিল।

**✅ প্রশ্ন ৩: Event Loop কিভাবে কাজ করে?**

**উত্তর (বাংলা):**  
JavaScript সিঙ্গেল থ্রেডেড, কিন্তু Asynchronous কাজের জন্য এটি **Event Loop** ব্যবহার করে। এটি কাজ করে:

1. Call Stack
2. Web APIs (setTimeout, fetch ইত্যাদি)
3. Callback Queue / Task Queue
4. Event Loop

Event Loop দেখে নেয় Call Stack খালি কিনা, খালি হলে Callback Queue থেকে ফাংশন নিয়ে Call Stack-এ পুশ করে।

**✅ প্রশ্ন ৪: this কী এবং এটি কীভাবে কাজ করে?**

**উত্তর (বাংলা):**  
this হলো একটি কীওয়ার্ড যা runtime-এ নির্ধারিত হয়। এটি নির্ভর করে আপনি কোন context-এ কোড লিখেছেন।

উদাহরণ:

const person = {

name: "Rahim",

sayHi: function () {

console.log(this.name);

}

};

person.sayHi(); // Rahim

Arrow Function এ this কাজ করে lexical context অনুযায়ী।

**✅ প্রশ্ন ৫: আপনি কিভাবে একটি বড় টিমে JavaScript কোড maintainable রাখেন?**

**উত্তর (বাংলা):**  
Lead হিসেবে আমি নিম্নলিখিত স্ট্র্যাটেজিগুলো ব্যবহার করি:

* Code Convention (ESLint, Prettier)
* Component-based architecture (React/Angular)
* SOLID Principles (যথাসম্ভব)
* Code Review Process
* Automated Unit and Integration Tests (Jest, Cypress)
* Documentation (JSDoc, Notion)

**✅ প্রশ্ন ৬: আপনি একটি Legacy JavaScript প্রজেক্টে কীভাবে রিফ্যাক্টর করবেন?**

**উত্তর (বাংলা):**

1. প্রথমে Static Analysis Tools দিয়ে কোড স্ক্যান করি (ESLint, SonarQube)
2. Unit test coverage দেখি
3. Code মডিউলভিত্তিক ভাগ করি
4. ডিপেন্ডেন্সি ম্যানেজ করি (npm audit, package update)
5. প্রয়োজন হলে ES6+ Syntax এ রিফ্যাক্টর করি
6. পুরানো ফিচারগুলোর জন্য Feature Toggle ব্যবহার করি

**✅ প্রশ্ন ৭: JavaScript এর async/await কী এবং এর সুবিধা কী?**

**উত্তর (বাংলা):**  
async/await হলো Promises ব্যবহারের একটি আধুনিক ও সহজ উপায়।

async function fetchData() {

try {

const response = await fetch("url");

const data = await response.json();

console.log(data);

} catch (error) {

console.error(error);

}

}

**সুবিধা:**

* কোড দেখতে সিঙ্ক্রোনাসের মতো
* Error handling সহজ
* Call chain বড় হলে বোঝা সহজ

**✅ প্রশ্ন ৮: আপনার প্রোজেক্টে আপনি কিভাবে Performance Optimization করেন?**

**উত্তর (বাংলা):**

1. **Code-splitting** (Webpack dynamic import)
2. **Memoization** (React.memo, useMemo, lodash.memoize)
3. **Lazy Loading**
4. **Avoid unnecessary re-renders**
5. **Minification and Compression**
6. **Web Vitals Monitoring (Lighthouse, Core Web Vitals)**

নিচে **Lead Software Engineer (JavaScript)** ইন্টারভিউয়ের জন্য গুরুত্বপূর্ণ প্রশ্নগুলোর আরও বিস্তারিত বাংলা ব্যাখ্যা দেওয়া হলো। এখানে শুধু উত্তর না, বরং প্রতিটি বিষয়ের পেছনের ধারণা ও বাস্তব উদাহরণও তুলে ধরা হয়েছে যাতে আপনি গভীরভাবে বুঝতে পারেন।

**🔹 ১. JavaScript Execution Context বিস্তারিত ব্যাখ্যা**

**Execution Context** হচ্ছে একটি কনসেপ্ট যেখানে JavaScript কোড রান করে। প্রতিটি কোড ব্লক যখন রান হয়, তখন নতুন Execution Context তৈরি হয়।

**Execution Context এর ধাপগুলো:**

**➤ ১. Creation Phase:**

* Variable Object (VO) তৈরি হয়
* Scope Chain তৈরি হয়
* this কী সেট করা হয়

**➤ ২. Execution Phase:**

* Actual কোড Execute হয়
* ভ্যারিয়েবল ও ফাংশনকে মান দেওয়া হয়

**উদাহরণ:**

var name = "Rahim";

function greet() {

var msg = "Hello " + name;

console.log(msg);

}

greet();

➡️ greet() ফাংশন কল করার সময় নতুন Execution Context তৈরি হবে, যেটা আলাদা scope তৈরি করবে।

**🔹 ২. Closure বিস্তারিত ব্যাখ্যা**

Closure এমন একটি ফাংশন যেটা নিজের parent scope-এর ভ্যারিয়েবল retain করতে পারে এমনকি ফাংশন কল শেষ হয়ে যাওয়ার পরও।

**উদাহরণ:**

function counter() {

let count = 0;

return function () {

count++;

console.log(count);

};

}

const inc = counter();

inc(); // 1

inc(); // 2

➡️ এখানে count ভ্যারিয়েবল শুধু counter স্কোপে ছিল, কিন্তু inc ফাংশন এটাকে retain করে রাখছে — একেই বলে Closure।

**ব্যবহার:**

* Private Data তৈরি করতে
* Currying
* Memoization

**🔹 ৩. Event Loop বিস্তারিত ব্যাখ্যা**

JavaScript একধরনের Single-threaded Language, কিন্তু Asynchronous কাজ করার জন্য Event Loop ব্যবহার করে।

**Component গুলো:**

* **Call Stack:** যেখানে synchronous ফাংশনগুলো থাকে
* **Web APIs:** Timer, DOM, HTTP ইত্যাদি
* **Callback Queue:** যেখান থেকে async কাজ শেষ হলে কলব্যাক আসে
* **Event Loop:** Call Stack খালি থাকলে Callback Queue থেকে কাজ নেয়

**উদাহরণ:**

console.log("Start");

setTimeout(() => {

console.log("Timeout");

}, 0);

console.log("End");

**Output:**

Start

End

Timeout

➡️ setTimeout Web API তে যায় এবং পরে Callback Queue তে ফিরে আসে, কিন্তু Event Loop তখনই তাকে Stack এ পাঠায় যখন Stack ফাঁকা হয়।

**🔹 ৪. this কী এবং এর ধরনগুলো**

**this** কী-ওয়ার্ড context অনুযায়ী ভিন্ন আচরণ করে।

**➤ ১. Global Scope:**

console.log(this); // window (browser), global (node)

**➤ ২. Object Method:**

const obj = {

name: "Rahim",

sayHi() {

console.log(this.name); // Rahim

}

};

**➤ ৩. Arrow Function:**

const obj = {

name: "Rahim",

sayHi: () => {

console.log(this.name); // undefined (lexical scope)

}

};

➡️ Arrow function এ this bind হয় না, parent scope থেকে নেয়।

**🔹 ৫. JavaScript কোড Maintainable রাখার উপায়**

একজন Lead হিসেবে Maintainability নিশ্চিত করার জন্য আমি নিচের practice গুলো অনুসরণ করি:

1. **Linting Rules:** ESLint, Prettier — কোড স্টাইল ও consistency বজায় রাখতে
2. **Type Safety:** TypeScript বা JSDoc
3. **Componentization:** বড় অ্যাপ হলে ছোট ছোট reusable component বানাই
4. **Naming Conventions:** Meaningful variable ও function name
5. **Testing:** Unit Test (Jest), Integration Test (Cypress)
6. **Documentation:** JSDoc, Markdown Readme, Notion page

**🔹 ৬. Legacy JavaScript Code Refactor করার পদ্ধতি**

1. **Code Audit:** পুরানো কোডে কোন deprecated/antipattern ব্যবহার হয়েছে খুঁজে বের করা
2. **Test Coverage:** পুরানো কোডের উপর unit test লিখে নেয়া যেন refactor করলে bug না আসে
3. **Split Responsibility:** বড় বড় ফাংশন ছোট করে ভাগ করা
4. **Convert to ES6+:** var → let/const, function → arrow function, class ব্যবহার ইত্যাদি
5. **Introduce Module System:** CommonJS / ES Module এ ভাগ করা
6. **Progressive Refactor:** একসাথে সব না বদলে ধাপে ধাপে উন্নয়ন

**🔹 ৭. async/await বিস্তারিত ব্যাখ্যা**

**async/await** হলো Promise-ভিত্তিক asynchronous কোড লেখার জন্য সিঙ্ক্রোনাস-এর মত syntax।

**উদাহরণ:**

async function fetchData() {

try {

const res = await fetch("https://api.com/data");

const json = await res.json();

console.log(json);

} catch (e) {

console.error("Error", e);

}

}

**সুবিধা:**

* .then().catch() চেইনের জটিলতা কমে
* কোড আরও clean এবং readable হয়
* Error handling সহজ

**🔹 ৮. Performance Optimization কৌশল**

1. **Debouncing & Throttling:** Input Field, Scroll ইভেন্টে ব্যবহার
2. **Lazy Loading:** Images ও React Components-এ
3. **Memoization:** React.memo, useMemo, lodash.memoize
4. **Tree Shaking & Code Splitting:** Webpack Config
5. **Network Optimization:** Minify, Gzip, Cache-Control headers
6. **Critical Rendering Path কমানো:** Unused CSS অপসারণ, defer JS

নিচে একটি **Lead Software Engineer (jQuery)** ইন্টারভিউতে সাধারণত যেসব প্রশ্ন করা হয়, সেগুলোর বাংলা অনুবাদ সহ উত্তর দেওয়া হলো। এগুলো সিনিয়র লেভেলের, অর্থাৎ যেখানে শুধু সিনট্যাক্স না, গভীরভাবে বোঝা যাচাই করা হয়।

**🔹 ১. jQuery কি? এবং এটি কেন ব্যবহৃত হয়?**

**প্রশ্নের বাংলা ব্যাখ্যা:** jQuery কী এবং আপনি এটি কেন ব্যবহার করবেন?

**উত্তর:**  
jQuery একটি ফাস্ট, ছোট এবং ফিচার-রিচ JavaScript লাইব্রেরি। এটি DOM traversal, event handling, AJAX call এবং animation সহজ করে তোলে।

**উদাহরণ:**

// jQuery দিয়ে একটি DOM element হাইলাইট করা

$("#myDiv").css("background-color", "yellow");

**কেন ব্যবহার করবো?**

* Cross-browser compatibility
* Less code, more work
* Readable syntax

**🔹 ২. jQuery এর .each() ফাংশন কীভাবে কাজ করে?**

**উত্তর:**  
.each() ফাংশনটি একটি লুপের মত কাজ করে যা একটি jQuery object এর প্রতিটি element-এর উপর চালানো যায়।

**উদাহরণ:**

$("li").each(function(index, element){

console.log(index + ": " + $(element).text());

});

**ব্যাখ্যা (বাংলায়):**  
প্রতিটি <li> এর টেক্সট কনসোলে দেখাবে। index হচ্ছে অবস্থান, আর element হচ্ছে DOM object।

**🔹 ৩. jQuery দিয়ে AJAX কল কীভাবে করবেন?**

**উত্তর:**

$.ajax({

url: '/api/getUser',

type: 'GET',

success: function(response) {

console.log(response);

},

error: function(error) {

console.error("Error:", error);

}

});

**বাংলায় ব্যাখ্যা:**  
এই কোডটি একটি GET request পাঠায় এবং সফল হলে response কনসোলে দেখায়, অন্যথায় error কনসোলে দেখায়।

**🔹 ৪. jQuery এর event delegation কী? কেন এটি ব্যবহার করবেন?**

**উত্তর:**  
Event delegation মানে parent element-এর মাধ্যমে dynamically add হওয়া child element-এর event handle করা।

$("#parent").on("click", ".child", function(){

alert("Clicked on child!");

});

**বাংলায় ব্যাখ্যা:**  
যদি কোন child element ডাইনামিকভাবে add হয়, সরাসরি .click() কাজ করবে না, তাই parent এর মাধ্যমে ধরতে হয়।

**🔹 ৫. jQuery দিয়ে একটি DOM element এর attribute কিভাবে পরিবর্তন করবেন?**

**উত্তর:**

$("#myImage").attr("src", "newimage.jpg");

**বাংলায় ব্যাখ্যা:**  
এই কোডটি id="myImage" ওয়ালা ইমেজ এর src পরিবর্তন করে।

**🔹 ৬. jQuery এর chaining কী? উদাহরণ দিন।**

**উত্তর:**

$("#box").css("color", "red").slideUp(2000).slideDown(2000);

**বাংলায় ব্যাখ্যা:**  
একটি element এর উপর একাধিক method ধারাবাহিকভাবে (চেইন করে) প্রয়োগ করা — এটাকে chaining বলে।

**🔹 ৭. jQuery vs Vanilla JavaScript: আপনি কখন কোনটা ব্যবহার করবেন?**

**উত্তর (বাংলায়):**

* যদি আপনার অ্যাপটি আধুনিক ব্রাউজার সাপোর্ট করে, তাহলে Vanilla JavaScript যথেষ্ট।
* তবে যদি backward compatibility (IE 9 বা পুরাতন) দরকার হয়, তাহলে jQuery সুবিধাজনক।
* বড় অ্যাপ বা framework (React/Vue/Angular) এর ক্ষেত্রে jQuery না ব্যবহার করাই ভালো।

**🔹 ৮. jQuery দিয়ে form validation কীভাবে করবেন?**

**উত্তর:**

$("#myForm").submit(function(e){

if ($("#name").val() === "") {

alert("Name is required!");

e.preventDefault();

}

});

**বাংলায় ব্যাখ্যা:**  
form সাবমিট করার সময় যদি নাম ফাঁকা থাকে তাহলে alert দেখাবে এবং সাবমিট বন্ধ করে দেবে।

**🔹 ৯. jQuery এ $(document).ready() এর গুরুত্ব কী?**

**উত্তর:**

$(document).ready(function(){

// code here

});

**বাংলায় ব্যাখ্যা:**  
এই ফাংশনটি নিশ্চিত করে যে DOM পুরোপুরি লোড হওয়ার পরে স্ক্রিপ্ট চলবে। এতে error কম হয়।

**🔹 ১০. Performance optimization এর জন্য আপনি কীভাবে jQuery কোড লিখবেন?**

**উত্তর (বাংলায়):**

* একাধিক DOM access একবারেই করুন, না হলে performance খারাপ হয়।
* DOM traversal কমান।
* Caching ব্যবহার করুন:

var $element = $("#item");

$element.hide();

$element.css("color", "blue");

নিচে Lead Software Engineer পদের জন্য jQuery সম্পর্কিত আরো বিস্তারিত বাংলা প্রশ্ন ও উত্তর দেওয়া হলো, যেখানে প্রতিটি টপিকের ব্যাখ্যা, কোড উদাহরণ, বাস্তব প্রয়োগ এবং ইন্টারভিউ প্রসঙ্গ আলোচনা করা হয়েছে।

**🔷 প্রশ্ন ১: jQuery কী? এটি কিভাবে কাজ করে?**

**উত্তর (বিশদভাবে):**  
jQuery হলো একটি JavaScript লাইব্রেরি যা DOM manipulation, event handling, animation, এবং AJAX calls সহজ করে তোলে। এটি Cross-browser সমস্যা দূর করে এবং খুব কম কোডে অনেক কাজ করার সুবিধা দেয়।

**✅ মূল ফিচার:**

* DOM Element নির্বাচন ($('#id'))
* CSS পরিবর্তন
* Event সংযুক্তি (click, hover)
* AJAX কল
* Animations (fadeIn, slideUp)

**🎯 ইন্টারভিউ টিপ:**

**প্রশ্ন:** আপনি যদি শুধু Vanilla JS জানেন, তাহলে jQuery শিখার দরকার কী?  
**উত্তর:** jQuery অনেক কোড কমিয়ে দেয় এবং backward compatibility নিশ্চিত করে।

**🔷 প্রশ্ন ২: jQuery Chaining কী এবং এটি কখন ব্যবহার করবেন?**

**উত্তর (বিশদভাবে):**  
Chaining মানে হলো একাধিক jQuery মেথড একসাথে কল করা যাতে প্রতিবার নতুন করে DOM খুঁজে আনতে না হয়।

**✅ উদাহরণ:**

$("#myDiv").css("color", "red").slideUp(1000).slideDown(1000);

**✅ কেন এটা দরকার?**

* Performance বাড়ায়
* কোড পড়তে ও লিখতে সহজ হয়

**🔷 প্রশ্ন ৩: Event Delegation কী?**

**উত্তর (বিশদভাবে):**  
Event Delegation হলো এমন একটি টেকনিক যেখানে আপনি একটি parent element-এর উপর event listener বসিয়ে সেই event child element-এ propagate হতে দেন। এটি খুবই উপকারী যখন আপনি ডাইনামিকভাবে DOM-এ নতুন এলিমেন্ট অ্যাড করেন।

**✅ উদাহরণ:**

$("#todoList").on("click", ".deleteBtn", function(){

$(this).parent().remove();

});

**ব্যাখ্যা:** .deleteBtn যদি পরবর্তীতে ডাইনামিকভাবে অ্যাড হয়, তাহলে সরাসরি .click() কাজ করবে না। তাই #todoList (parent) এর মাধ্যমে event ধরতে হবে।

**🎯 ইন্টারভিউ টিপ:**

**প্রশ্ন:** Event bubbling ও capturing এর মধ্যে পার্থক্য কী?

**🔷 প্রশ্ন ৪: AJAX দিয়ে সার্ভার কল কিভাবে করবেন এবং Error Handling করবেন?**

**উত্তর (বিশদভাবে):**  
AJAX (Asynchronous JavaScript and XML) ব্যবহার করে আপনি পেজ রিলোড না করেই সার্ভার থেকে ডেটা নিতে পারেন।

**✅ উদাহরণ:**

$.ajax({

url: "/api/users",

type: "GET",

dataType: "json",

success: function(data){

console.log("Users:", data);

},

error: function(xhr, status, error){

console.error("Something went wrong:", error);

}

});

**🎯 ভালো অভ্যাস:**

* dataType নির্ধারণ করা
* .done(), .fail() ব্যবহার করা
* try-catch ব্লক প্রয়োগ (async-await এর মতো)

**🔷 প্রশ্ন ৫: Form validation কিভাবে করবেন?**

**উত্তর (বিশদভাবে):**

$("#registerForm").submit(function(e){

var name = $("#name").val();

if(name === ""){

alert("Name is required!");

e.preventDefault(); // Submit বন্ধ করে দেয়

}

});

**✅ Validation স্টেপ:**

1. ফর্ম সাবমিট ইভেন্ট ধরুন
2. ফিল্ড ভ্যালু চেক করুন
3. শর্ত অনুযায়ী সাবমিট allow বা block করুন

**🔷 প্রশ্ন ৬: DOM Traversal নিয়ে আপনার অভিজ্ঞতা ব্যাখ্যা করুন।**

**উত্তর (বিশদভাবে):**  
DOM Traversal মানে হলো DOM-এর ভিতরে উপরে-নিচে যাওয়া।

**✅ উদাহরণ:**

// Parent Element পাওয়া

$("#item").parent();

// Child গুলো পাওয়া

$("#item").children();

// Next Sibling

$("#item").next();

**🎯 ইন্টারভিউ টিপ:**

**প্রশ্ন:** আপনি .find() এবং .children() এর মধ্যে পার্থক্য জানেন?  
**উত্তর:** .children() শুধুমাত্র direct children খোঁজে, .find() সমস্ত descendant খুঁজে।

**🔷 প্রশ্ন ৭: Custom Plugin তৈরি করা যায় কীভাবে?**

**উত্তর (বিশদভাবে):**

(function($){

$.fn.highlight = function(color){

this.css("background-color", color);

return this;

};

})(jQuery);

// ব্যবহার

$("p").highlight("yellow");

**🎯 ইন্টারভিউ টিপ:**

Custom Plugin তৈরি করতে জানলে আপনি advanced jQuery developer হিসেবে বিবেচিত হবেন।

**🔷 প্রশ্ন ৮: Performance Optimization এর টেকনিক কী কী?**

**উত্তর (বিশদভাবে):**

1. DOM caching:
2. var $el = $("#item");
3. $el.hide();
4. $el.css("color", "blue");
5. Event delegation ব্যবহার করা
6. Animation মিনিমাইজ করা
7. Unnecessary reflows এড়িয়ে চলা

**🔷 প্রশ্ন ৯: jQuery দিয়ে element animate কীভাবে করবেন?**

**উত্তর:**

$("#box").animate({

left: '250px',

opacity: 0.5

}, 2000);

**ব্যাখ্যা:** এই কোডটি #box কে বামে সরায় এবং ধীরে ধীরে অর্ধেক স্বচ্ছ করে ২ সেকেন্ডে।

**🔷 প্রশ্ন ১০: jQuery এর বদলে এখন কী কী আধুনিক প্রযুক্তি ব্যবহার হয়?**

**উত্তর (বাংলায়):**

* Vanilla JavaScript (ES6+): document.querySelector, fetch()
* React/Vue/Angular: DOM manipulation নিজে করে না, Virtual DOM ব্যবহার করে
* Axios বা Fetch API: AJAX এর পরিবর্তে
* CSS Animation: .animate() এর পরিবর্তে

নিচে **Lead Software Engineer পদের জন্য jQuery Advanced Interview Set 2** বাংলা ভাষায় বিস্তারিতভাবে দেওয়া হলো, যেখানে প্রত্যেকটি প্রশ্নের গভীর ব্যাখ্যা, কোড উদাহরণ এবং ইন্টারভিউয়ে আলোচনার প্রাসঙ্গিক দিক তুলে ধরা হয়েছে।

**🔷 প্রশ্ন ১: jQuery Deferred এবং Promise কী? এগুলো কিভাবে কাজ করে?**

**উত্তর (বাংলায়):**  
jQuery এর Deferred এবং Promise object asynchronous operations (যেমন AJAX) হ্যান্ডেল করার জন্য ব্যবহার করা হয়।

**✅ উদাহরণ:**

function getData(){

var deferred = $.Deferred();

setTimeout(function(){

deferred.resolve("ডেটা পাওয়া গেছে");

}, 2000);

return deferred.promise();

}

getData().done(function(data){

console.log(data);

});

**🎯 ব্যাখ্যা:**

* Deferred তৈরি হয় এবং resolve/reject করা হয়।
* Promise হল read-only ভার্সন, যা শুধু .done() বা .fail() হ্যান্ডেল করে।

**🔷 প্রশ্ন ২: $.when() এর ব্যবহার কী?**

**উত্তর (বাংলায়):**  
$.when() ব্যবহার করে আপনি একাধিক asynchronous অপারেশন একসাথে হ্যান্ডেল করতে পারেন।

**✅ উদাহরণ:**

var d1 = $.Deferred();

var d2 = $.Deferred();

$.when(d1, d2).done(function(res1, res2){

console.log("দুইটি কাজই শেষ হয়েছে", res1, res2);

});

setTimeout(() => d1.resolve("d1 সম্পন্ন"), 1000);

setTimeout(() => d2.resolve("d2 সম্পন্ন"), 2000);

**🔷 প্রশ্ন ৩: jQuery এর .data() মেথড কী? এর সুবিধা কী?**

**উত্তর:**  
.data() DOM element এর সাথে কাস্টম ডেটা অ্যাটাচ/রিড করার জন্য ব্যবহৃত হয়।

**✅ উদাহরণ:**

<div id="product" data-id="101" data-name="Laptop"></div>

var id = $("#product").data("id"); // 101

**✅ সুবিধা:**

* Attribute এর মত কিন্তু পারফর্মেন্স ফ্রেন্ডলি
* Data binding সহজ হয়

**🔷 প্রশ্ন ৪: live() vs delegate() vs on() – পার্থক্য ব্যাখ্যা করুন।**

**উত্তর:**

| **মেথড** | **অবস্থা** | **ব্যবহারের ধরন** |
| --- | --- | --- |
| live() | Deprecated | পুরানো, শুধু document level এ |
| delegate() | পুরানো কিন্তু ভাল | parent থেকে child target |
| on() | আধুনিক ও সুপারিশকৃত | সর্বত্র কাজ করে |

**✅ উদাহরণ (on):**

$(document).on("click", ".button", function(){

alert("Clicked!");

});

**🔷 প্রশ্ন ৫: jQuery plugin lifecycle কেমন হওয়া উচিত?**

**উত্তর (বাংলায়):**  
jQuery plugin তৈরির সময় নিচের স্টেপ অনুসরণ করুন:

1. Plugin নাম define করুন
2. Default options সেট করুন
3. DOM element access করুন
4. Methods তৈরি করুন
5. Chaining সাপোর্ট করুন

**✅ উদাহরণ:**

(function($){

$.fn.myPlugin = function(options){

var settings = $.extend({

color: "red"

}, options);

return this.each(function(){

$(this).css("color", settings.color);

});

};

})(jQuery);

$("p").myPlugin({ color: "blue" });

**🔷 প্রশ্ন ৬: jQuery তে throttle ও debounce এর প্রভাব ব্যাখ্যা করুন।**

**উত্তর (বাংলায়):**

* **Debounce:** একটি নির্দিষ্ট সময় অপেক্ষা করে তারপর ফাংশন একবার চলে (যেমন টাইপিং এর পর ৫০০ms পর search শুরু করা)।
* **Throttle:** নির্দিষ্ট সময়ের মধ্যে একাধিক বার ফাংশন চলতে না দেওয়া।

**✅ উদাহরণ (debounce):**

function debounce(fn, delay){

var timer;

return function(){

clearTimeout(timer);

timer = setTimeout(fn, delay);

}

}

$(window).on("resize", debounce(function(){

console.log("Resized!");

}, 500));

**🔷 প্রশ্ন ৭: jQuery দিয়ে complex animation কিভাবে করবেন?**

**উত্তর:**

$("#box").animate({

left: "200px",

height: "toggle",

opacity: 0.7

}, 1000, "swing", function(){

alert("Animation শেষ!");

});

**✅ ব্যাখ্যা:**

* animate() কে multiple properties দেয়া যায়
* callback দিয়ে animation শেষে কাজ করানো যায়

**🔷 প্রশ্ন ৮: jQuery এ .queue() এবং .dequeue() কী করে?**

**উত্তর:**

* .queue() একটি element এর কাজের সিরিজ ধারণ করে
* .dequeue() ঐ queue থেকে কাজ চালায়

**✅ উদাহরণ:**

$("#box")

.queue(function(next){

$(this).css("background", "red");

next();

})

.queue(function(next){

$(this).slideUp();

next();

});

**🔷 প্রশ্ন ৯: jQuery UI নিয়ে আপনার অভিজ্ঞতা কী?**

**উত্তর (বাংলায়):**  
**jQuery UI** হলো jQuery এর উপরে তৈরি একটি লাইব্রেরি যা সহজে UI element (datepicker, dialog, draggable, resizable) তৈরির জন্য ব্যবহার হয়।

**✅ উদাহরণ:**

$("#datepicker").datepicker();

**🔷 প্রশ্ন ১০: কেন jQuery অনেক জায়গায় আর ব্যবহার হয় না? আপনার মতামত দিন।**

**উত্তর (বাংলায়):**

**কারণসমূহ:**

* নতুন ব্রাউজার অনেক API native সাপোর্ট করে
* Modern frameworks (React, Vue) automatic DOM binding দেয়
* Performance ও maintainability এর কারণে framework বেছে নেয়া হয়

**তবে jQuery এখনো যেখানে ব্যবহৃত হয়:**

* Legacy system
* Lightweight static websites
* দ্রুত development এর জন্য