

index.html

```
1 The Document Object Model (DOM) is a programming interface that represents the structure of a
  web document, allowing scripts to manipulate its content and layout. JavaScript interacts with
  the DOM to dynamically change elements, styles, and attributes on a web page.
2
3 For working with databases in PHP, the main extensions are MySQLi and PDO. MySQLi offers both
  procedural and object-oriented interfaces specifically for MySQL, while PDO provides a more
  flexible, object-oriented approach that supports multiple database types. 1. What is the
  Document Object Model (DOM)? How does JavaScript interact with it?
4
5 Definition: The Document Object Model (DOM) is a programming interface that represents the
  structure of a web page as a tree of objects. Each part of the HTML document (elements,
  attributes, text) is represented as a node, allowing for dynamic interaction and manipulation.
6
7 Structure:
8
9 The DOM is organized hierarchically, with the document as the root.
10 HTML tags like <html>, <head>, and <body> serve as branches, while attributes and text are the
  leaves.
11 JavaScript Interaction:
12
13 JavaScript acts as a tool to manipulate the DOM, enabling developers to:
14 Access elements using methods like getElementById, getElementsByClassName, and querySelector.
15 Modify content, styles, and attributes of elements.
16 Respond to user events (e.g., clicks, hovers) to create interactive experiences.
17 Dynamically add or remove elements from the document.
18 Example:
19
20 javascript
21
22 Verify
23
24 Open In Editor
25 Run
26 Copy code
27 let button = document.getElementById("myButton");
28 button.addEventListener("click", function() {
29     button.style.backgroundColor = "blue";
30 });
31 2. What are the main PHP extensions for working with databases? Compare MySQLi and PDO
32
33 Main PHP Extensions:
34
35 MySQLi (MySQL Improved):
36
37 Specifically designed for MySQL databases.
38 Supports both procedural and object-oriented programming styles.
39 Offers features like prepared statements, transactions, and support for multiple statements.
40 Limited to MySQL databases only.
41 PDO (PHP Data Objects):
```

42
43 A database access layer providing a uniform method of access to multiple databases.
44 Supports various database systems (MySQL, PostgreSQL, SQLite, etc.).
45 Object-oriented interface with features like prepared statements and transactions.
46 More flexible and secure due to its support for multiple database types.
47 Comparison:
48
49 Flexibility: PDO is more flexible as it supports multiple database types, while MySQLi is
limited to MySQL.
50 API Style: MySQLi can be used in both procedural and object-oriented styles, whereas PDO is
strictly object-oriented.
51 Security: Both support prepared statements, but PDO's abstraction allows for better security
practices across different databases.
52 Performance: MySQLi may offer better performance for MySQL-specific features, while PDO
provides a consistent interface across different databases.
53 In summary, the choice between MySQLi and PDO depends on the specific needs of the application,
such as the database being used and the desired programming style.
54
55 You
56
57 What are the different methods available in JavaScript for manipulating arrays? Provide
examples of push and slice.
58 [C01, Remember]
59
60 What are events in JavaScript? Give examples of at least three types of events. Explain the
difference between addEventListener and inline event handling.
61 [C01, Remember]
62
63 BLACKBOXAI
64
65 1. Different Methods for Manipulating Arrays in JavaScript
66 JavaScript provides a variety of methods for manipulating arrays. Here are some common methods:
67
68 push(): Adds one or more elements to the end of an array and returns the new length of the
array.
69 pop(): Removes the last element from an array and returns that element.
70 shift(): Removes the first element from an array and returns that element.
71 unshift(): Adds one or more elements to the beginning of an array and returns the new length of
the array.
72 slice(): Returns a shallow copy of a portion of an array into a new array object selected from
start to end (end not included).
73 splice(): Changes the contents of an array by removing or replacing existing elements and/or
adding new elements in place.
74 forEach(): Executes a provided function once for each array element.
75 map(): Creates a new array populated with the results of calling a provided function on every
element in the calling array.
76 filter(): Creates a new array with all elements that pass the test implemented by the provided
function.
77 reduce(): Executes a reducer function on each element of the array, resulting in a single
output value.
78 Example of push() and slice()
79 javascript

```
80
81 Verify
82
83 Open In Editor
84 Run
85 Copy code
86 // Example of push()
87 let fruits = ['apple', 'banana'];
88 fruits.push('orange'); // Adds 'orange' to the end of the array
89 console.log(fruits); // Output: ['apple', 'banana', 'orange']
90
91 // Example of slice()
92 let slicedFruits = fruits.slice(1, 3); // Extracts elements from index 1 to 2 (3 is not
    included)
93 console.log(slicedFruits); // Output: ['banana', 'orange']
94 2. Events in JavaScript
95 Events in JavaScript are actions or occurrences that happen in the browser, which can be
    detected and responded to by JavaScript code. Events can be triggered by user interactions
    (like clicks, key presses, etc.) or by the browser itself (like page load, resize, etc.).
96
97 Examples of Three Types of Events
98 Click Event: Triggered when an element is clicked.
99
100 javascript
101
102 Verify
103
104 Open In Editor
105 Run
106 Copy code
107 document.getElementById("myButton").addEventListener("click", function() {
108     alert("Button clicked!");
109 });
110 Mouseover Event: Triggered when the mouse pointer moves over an element.
111
112 javascript
113
114 Verify
115
116 Open In Editor
117 Run
118 Copy code
119 document.getElementById("myDiv").addEventListener("mouseover", function() {
120     this.style.backgroundColor = "yellow";
121 });
122 Keydown Event: Triggered when a key is pressed down.
123
124 javascript
125
126 Verify
127
```

128 Open In Editor
129 Run
130 Copy code
131 `document.addEventListener("keydown", function(event) {`
132 `console.log("Key pressed: " + event.key);`
133 `});`
134 Difference Between addEventListener and Inline Event Handling
135 addEventListener:
136
137 Allows you to attach multiple event handlers to a single event on a single element.
138 Provides more flexibility and separation of HTML and JavaScript.
139 Can specify the event phase (capturing or bubbling) and options (like once, passive).
140 Example:
141 javascript
142
143 Verify
144
145 Open In Editor
146 Run
147 Copy code
148 `document.getElementById("myButton").addEventListener("click", function() {`
149 `alert("Button clicked!");`
150 `});`
151 Inline Event Handling:
152
153 Involves adding event handlers directly in the HTML markup using attributes like onclick, onmouseover, etc.
154 Less flexible as it can only attach one handler per event type on an element.
155 Can lead to less maintainable code as it mixes HTML and JavaScript.
156 Example:
157 html
158
159 Verify
160
161 Open In Editor
162 Run
163 Copy code
164 `<button onclick="alert('Button clicked!')">Click Me</button>`
165 In summary, while both methods can be used to handle events, addEventListener is generally preferred for its flexibility and maintainability.
166
167 You
168
169 What is BOM and how does it differ from DOM? Explain the main objects provided by BOM.
170 [C01, Remember]
171
172 What is a Database API and why is it important in application development? What are the steps involved in using a Database API to interact with a database?
173 [C01, Remember]
174
175 BLACKBOXAI

```
176
177 1. What is BOM and How Does It Differ from DOM?
178 BOM (Browser Object Model):
179
180 The Browser Object Model (BOM) is a set of objects provided by the web browser that allows
    JavaScript to interact with the browser itself, rather than just the content of the web page.
    It provides a way to manipulate the browser window and its components.
181 BOM includes objects that represent the browser window, the document, and other browser-
    specific features.
182 Difference Between BOM and DOM:
183
184 DOM (Document Object Model):
185
186 The DOM represents the structure of the HTML document as a tree of objects. It allows
    JavaScript to manipulate the content, structure, and style of the web page.
187 Focuses on the document (HTML/CSS) and its elements.
188 BOM:
189
190 The BOM provides an interface to the browser itself, allowing interaction with the browser
    window, history, location, and other browser features.
191 Focuses on the browser environment and its functionalities.
192 Main Objects Provided by BOM
193 Window Object: Represents the browser window and is the global object in JavaScript. It
    provides methods to control the browser window, such as alert(), setTimeout(), and open().
194
195 javascript
196
197 Verify
198
199 Open In Editor
200 Run
201 Copy code
202 window.alert("Hello, World!");
203 Document Object: While primarily part of the DOM, the document object is also part of the BOM.
    It represents the HTML document loaded in the browser and provides methods to manipulate it.
204
205 javascript
206
207 Verify
208
209 Open In Editor
210 Run
211 Copy code
212 document.getElementById("myElement").innerHTML = "New Content";
213 Location Object: Represents the current URL of the document and provides methods to get or set
    the URL, reload the page, and navigate to different URLs.
214
215 javascript
216
217 Verify
218
```

```
219 Open In Editor
220 Run
221 Copy code
222 console.log(window.location.href); // Get current URL
223 window.location.href = "https://www.example.com"; // Navigate to a new URL
224 History Object: Provides access to the browser's session history, allowing navigation through
    the user's history (back and forward).
225
226 javascript
227
228 Verify
229
230 Open In Editor
231 Run
232 Copy code
233 window.history.back(); // Go back to the previous page
234 Navigator Object: Contains information about the browser and the operating system, such as the
    browser name, version, and user agent.
235
236 javascript
237
238 Verify
239
240 Open In Editor
241 Run
242 Copy code
243 console.log(navigator.userAgent); // Get user agent string
244 2. What is a Database API and Why is it Important in Application Development?
245 Database API:
246
247 A Database API (Application Programming Interface) is a set of protocols and tools that allows
    applications to communicate with a database. It provides a way for developers to perform
    operations such as querying, updating, and managing data in a database.
248 Importance in Application Development:
249
250 Data Management: Database APIs enable applications to efficiently manage and manipulate data
    stored in databases.
251 Abstraction: They provide an abstraction layer, allowing developers to interact with databases
    without needing to understand the underlying database management system (DBMS) specifics.
252 Security: Database APIs often include built-in security features, such as prepared statements,
    to help prevent SQL injection attacks.
253 Interoperability: They allow applications to work with different types of databases (e.g., SQL,
    NoSQL) using a consistent interface.
254 Steps Involved in Using a Database API to Interact with a Database
255 Establish a Connection:
256
257 Use the Database API to connect to the database. This typically involves specifying the
    database type, host, username, password, and database name.
258 php
259
260 Verify
```

```
261
262 Open In Editor
263 Run
264 Copy code
265 // Example in PHP using PDO
266 $pdo = new PDO('mysql:host=localhost;dbname=testdb', 'username', 'password');
267 Prepare a Query:
268
269 Write a SQL query to perform the desired operation (e.g., SELECT, INSERT, UPDATE, DELETE).
270 php
271
272 Verify
273
274 Open In Editor
275 Run
276 Copy code
277 $stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");
278 Bind Parameters (if applicable):
279
280 Bind any parameters to the query to prevent SQL injection and ensure safe execution.
281 php
282
283 Verify
284
285 Open In Editor
286 Run
287 Copy code
288 $stmt->bindParam(':id', $userId);
289 Execute the Query:
290
291 Execute the prepared statement to interact with the database.
292 php
293
294 Verify
295
296 Open In Editor
297 Run
298 Copy code
299 $stmt->execute();
300 Fetch Results (if applicable):
301
302 Retrieve the results of the query, if it is a SELECT statement.
303 php
304
305 Verify
306
307 Open In Editor
308 Run
309 Copy code
310 $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
311 Handle Errors:
312
313 Implement error handling to manage any issues that arise during database interactions.
314 php
315
316 Verify
317
318 Open In Editor
319 Run
320 Copy code
321 try {
322     // Database operations
323 } catch (PDOException $e) {
324     echo "Error:
325 You
326
327 . Explain jQuery with an example and how does it simplify JavaScript coding? Explain the
    advantages of using jQuery over plain JavaScript.
328
329 [COL, Remember]
330
331 Explain the difference between setTimeout() and setInterval() methods in BOM. Provide an
    example of each
332 BLACKBOXAI
333
334 1. Explain jQuery with an Example and How It Simplifies JavaScript Coding
335 jQuery: jQuery is a fast, small, and feature-rich JavaScript library that simplifies HTML
    document traversal and manipulation, event handling, animation, and Ajax interactions for rapid
    web development. It provides an easy-to-use API that works across a multitude of browsers,
    making it easier to write JavaScript code.
336
337 How jQuery Simplifies JavaScript Coding:
338
339 Concise Syntax: jQuery allows developers to write less code to achieve the same functionality
    compared to plain JavaScript.
340 Cross-Browser Compatibility: jQuery handles many of the inconsistencies between different
    browsers, allowing developers to write code that works uniformly across all major browsers.
341 Chaining: jQuery allows method chaining, which means you can perform multiple operations on the
    same set of elements in a single statement.
342 Rich Plugin Ecosystem: jQuery has a vast ecosystem of plugins that extend its functionality,
    making it easy to add complex features without writing extensive code.
343 Example of jQuery
344 Here's a simple example that demonstrates how jQuery can be used to hide a paragraph when a
    button is clicked:
345
346 html
347
348 Verify
349
350 Open In Editor
351 Run
```



```

352 Copy code
353 <!DOCTYPE html>
354 <html lang="en">
355 <head>
356     <meta charset="UTF-8">
357     <meta name="viewport" content="width=device-width, initial-scale=1.0">
358     <title>jQuery Example</title>
359     <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
360     <script>
361         $(document).ready(function() {
362             $("#hideButton").click(function() {
363                 $("p").hide(); // Hides all <p> elements
364             });
365         });
366     </script>
367 </head>
368 <body>
369     <p>This is a paragraph that will be hidden when you click the button.</p>
370     <button id="hideButton">Hide Paragraph</button>
371 </body>
372 </html>

```

373 In this example:

374

375 The jQuery library is included via a CDN.

376 The `$(document).ready()` function ensures that the DOM is fully loaded before executing the code.

377 The `$("#hideButton").click()` method sets up an event handler for the button click, and `$("p").hide()` hides the paragraph.

378 Advantages of Using jQuery Over Plain JavaScript

379 Simplified Syntax: jQuery's syntax is more concise and easier to read, which reduces the amount of code needed for common tasks.

380 Cross-Browser Compatibility: jQuery abstracts away many of the differences between browsers, allowing developers to write code that works consistently across all major browsers.

381 Chaining Methods: jQuery allows chaining of methods, which can make code cleaner and more efficient.

382 Built-in Animation and Effects: jQuery provides built-in methods for animations and effects, making it easier to create dynamic user interfaces.

383 Ajax Support: jQuery simplifies Ajax calls with methods like `$.ajax()`, `$.get()`, and `$.post()`, making it easier to work with server-side data.

384 Rich Plugin Ecosystem: jQuery has a large number of plugins available, allowing developers to easily add complex features without having to write everything from scratch.

385 2. Difference Between `setTimeout()` and `setInterval()` Methods in BOM

386 `setTimeout()`:

387

388 The `setTimeout()` method is used to execute a function or a piece of code after a specified delay (in milliseconds). It runs the code only once after the delay.

389 Example of `setTimeout()`:

390

391 javascript

392

393 Verify

```
394
395 Open In Editor
396 Run
397 Copy code
398 console.log("Start");
399
400 setTimeout(function() {
401     console.log("This message is displayed after 2 seconds.");
402 }, 2000); // 2000 milliseconds = 2 seconds
403
404 console.log("End");
405 Output:
406
407
408 Verify
409
410 Open In Editor
411 Run
412 Copy code
413 Start
414 End
415 This message is displayed after 2 seconds.
416 setInterval():
417
418 The setInterval() method is used to repeatedly execute a function or a piece of code at
    specified intervals (in milliseconds). It continues to run the code until it is stopped using
    clearInterval().
419 Example of setInterval():
420
421 javascript
422
423 Verify
424
425 Open In Editor
426 Run
427 Copy code
428 let count = 0;
429
430 const intervalId = setInterval(function() {
431     count++;
432     console.log("This message is displayed every 1 second. Count: " + count);
433
434     // Stop the interval after 5 counts
435     if (count === 5) {
436         clearInterval(intervalId);
437         console.log("Interval cleared.");
438     }
439 }, 1000); // 1000 milliseconds = 1 second
440 Output (will display every second for 5 seconds):
441
```

```
442
443 Verify
444
445 Open In Editor
446 Run
447 Copy code
448 This message is displayed every 1 second. Count: 1
449 This message is displayed every 1 second. Count: 2
450 This message is displayed every 1 second. Count:
```