1. What does the **map()** method return?

A) The original array

**B) A new array with transformed elements**

C) Undefined

D) The length of the array

2. What is the output of this code?

**const nums = [1, 2, 3];**

**const doubled = nums.map((n, b, c) => n \* 2);**

**console.log(doubled);**

A) [1, 2, 3]

B) [2, 4, 6]

C) [1, 4, 9]

D) Error

3. Which of the following correctly uses **map()** to create an array of squares from **nums**?

**const nums = [1, 2, 3];**

**A) const squares = nums.map(n => n \*\* 2);**

B) const squares = nums.forEach(n => n \*\* 2);

C) const squares = nums.filter(n => n \*\* 2);

D) const squares = nums.map(n => n \* n).push(0);

4. What does the callback function in **map()** receive as arguments?

A) Only the current element

B) Current element and index

C) Current element, index, and the original array  
D) No arguments

5. What does the **filter()** method return?

A) A new array with elements that pass the condition

B) The original array

C) A boolean

D) Undefined

6. Which code returns only even numbers from **nums**?

**const nums = [1, 2, 3, 4];**

A) nums.map(n => n % 2 === 0)

B) nums.filter(n => n % 2 === 0)

C) nums.forEach(n => n % 2 === 0)

D) nums.reduce(n => n % 2 === 0)

7. What happens if no elements match the filter condition?

A) Returns undefined

B) Returns null

C) Returns an empty array

D) Throws an error

8. Which of the following creates an array of strings longer than 3 characters?

**const words = ["cat", "horse", "dog", "elephant"];**

A) words.map(word => word.length > 3)

B) words.filter(word => word.length > 3)

C) words.forEach(word => word.length > 3)

D) words.reduce((acc, word) => word.length > 3)

### 

### 

9. What does **forEach()** return?

A) A new array

B) The length of the array

C) Undefined

D) The original array

11. Can **forEach** be used to modify the original array?

**A) Yes**

B) No

12. What does **fetch()** return?

A) A promise

B) The response body

C) An object  
D) Undefined

13. What keyword is used to wait for a promise to resolve?

A) resolve

B) async

C) await

D) promise

14. What is wrong with this code?

**async function getData() {**

**let data = fetch("https://api.example.com/data");**

**console.log(data);**

**}**

A) **fetch()** must have a callback

B) **fetch()** is missing **await**

C) **fetch()** cannot be used with async functions

D) **fetch()** must use **then()**

15. What symbol is used to wrap template literals?

A) ' '

B) " "

C) ``

D) ( )

16. What is the output of this code?

**const name = "Alice";**

**console.log(`Hello, ${name}!`);**

A) Hello, Alice!

B) Hello, ${name}!

C) Hello, name!

D) Error

17. Which method selects a single element by its ID? Circle all that apply.

A) **document.querySelector("#id")**

B) **document.getElementById("id")**

C) **document.querySelectorAll("#id")**

D) **document.querySelectorAll("id")**

18. How do you add a paragraph to an element with an ID of **container**?

**const container = document.getElementById("container");**

A) **container.append("<p>Hello</p>");**

B) **container.innerHTML = "<p>Hello</p>";**

C) **container.addText("<p>Hello</p>");**

D) **container.addHTML("<p>Hello</p>");**

19. Given this array of objects, how would you iterate over it and log each item’s title?

**const products = [**

**{ id: 1, title: "Laptop", price: 1200 },**

**{ id: 2, title: "Phone", price: 800 },**

**{ id: 3, title: "Tablet", price: 500 }**

**];**

A) products.map(product => console.log(product.title));

B) products.filter(product => console.log(product.title));

**C) products.forEach(product => console.log(product.title));**

**D) for (let product of products) console.log(product.title);**

20. You have the following JavaScript starter file (main.js)...

**const container = document.getElementById("container");**

**const products = [**

**{ id: 1, title: "Laptop", price: 1200 },**

**{ id: 2, title: "Phone", price: 800 },**

**{ id: 3, title: "Stereo", price: 600 }**

**];**

…and the following HTML code snippet…

**<section id="container"></section>**

Write a JavaScript program (using the next page) that iterates through the **products** array and appends an HTML representation of each product to the container. After your JavaScript code runs, the DOM should look like this:

**<section id="container">**

**<!-- Card #1 -->**

**<div class="card">**

**<h2>Laptop</h2>**

**<p>Price: $1200</p>**

**</div>  
<!-- Card #2 -->**

**<div class="card">**

**<h2>Phone</h2>**

**<p>Price: $800</p>**

**</div>  
<!-- Card #3 -->**

**<div class="card">**

**<h2>Stereo</h2>**

**<p>Price: $600</p>**

**</div>**

**</section>**

21. What will the following code log?

**const person = { name: "Alice", age: 25, city: "New York" };**

**const { name, age } = person;**

**console.log(name, age);**

**A) Alice 25**

B) { name: "Alice", age: 25 }

C) "Alice"

D) undefined undefined

22. How can you extract the price from this object using destructuring?

**const product = { id: 1, name: "Laptop", price: 1200 };**

A) const price = product.price;

**B) const { price } = product;**

C) const [price] = product;

D) const { price: cost } = product;

23. What will the following code output?

**const nums = [1, 2, 3, 4];**

**const [a, b, c, d, e] = nums;**

**console.log(b, d);**

**A) 1 2**

B) 1 3

C) 2 3

D) 1 4

24. What does the spread operator (...) do in the following code?

**const nums = [1, 2, 3];**

**const moreNums = [...nums, 4, 5];**

A) Merges two arrays into one

B) Creates a reference to the original array

**C) Copies the contents of nums into moreNums**

D) Spreads the array into individual arguments

25. What will this code log to the console?

const obj1 = { a: 1, b: 2 };

const obj2 = { b: 3, c: 4 };

const merged = { ...obj1, ...obj2 };

console.log(merged);

A) { a: 1, b: 2, c: 4 }

**B) { a: 1, b: 3, c: 4 }**

C) { a: 1, c: 4 }

D) { a: 1, b: [2, 3], c: 4 }

26. What will this code log?

const person = { name: "Alice", age: 25 };

const clone = { ...person, age: 30 };

console.log(clone);

A) { name: "Alice", age: 25 }

**B) { name: "Alice", age: 30 }**

C) { name: "Alice" }

D) { age: 30 }

27. How can you extract **name** and collect the rest of the properties into rest?

const person = { name: "Alice", age: 25, city: "New York" };

A) const { name, ...rest } = person;

B) const [name, ...rest] = person;

C) const rest = person - "name";

D) const { ...rest, name } = person;

28. What will the following code output?

const nums = [1, 2, 3];

const [first, ...rest] = nums;

console.log(rest);

**A) [2, 3]**

B) 2

C) [1, 2]

D) 3

**29. What does this code do?**

const arr1 = [1, 2, 3];

const arr2 = [4, 5];

const combined = [...arr1, ...arr2];

console.log(combined);

A) Creates a nested array

B) Merges arr2 into arr1

C) Concatenates arr1 and arr2 into a new array

D) Throws an error

 **B)** A new array with transformed elements

 **B)** [2, 4, 6]

 **A)** const squares = nums.map(n => n \*\* 2);

 **C)** Current element, index, and the original array

 **A)** A new array with elements that pass the condition

 **B)** nums.filter(n => n % 2 === 0)

 **C)** Returns an empty array

 **B)** words.filter(word => word.length > 3)

 **C)** Undefined

 (Missing from the list)

 **A)** Yes

 **A)** A promise

 **C)** await

 **B)** fetch() is missing await

 **C)** ` `

 **A)** Hello, Alice!

 **A, B)** document.querySelector("#id"), document.getElementById("id")

 **B)** container.innerHTML = "<p>Hello</p>";

 **C, D)** products.forEach(product => console.log(product.title)); OR for (let product of products) console.log(product.title);

 *(See JavaScript solution below)*

 **A)** Alice 25

 **B)** const { price } = product;

 **D)** 2 4

 **C)** Copies the contents of nums into moreNums

 **B)** { a: 1, b: 3, c: 4 }

 **B)** { name: "Alice", age: 30 }

 **A)** const { name, ...rest } = person;

 **A)** [2, 3]

 **C)** Concatenates arr1 and arr2 into a new array