Below is a **clean, simplified, exam-ready, and concept-focused** explanation of the `void` **keyword in JavaScript**, written in the **same depth and structured note style** you've been following (perfect for NoteLM, revision, and exams).

🔹 `void` Keyword in JavaScript

The `void` **keyword** in JavaScript is an **operator** that **evaluates an expression and always returns** `undefined`.

👉 In simple words:
`void` → **run something, but return nothing**

📌 Key Meaning of `void`

- `void` means **empty / no value**
- It **discards the return value** of an expression
- It **always returns** `undefined`

📘 Syntax

```
js

void expression;
```

- `expression` can be **any valid JavaScript expression**
- Result is **always** `undefined`

🔁 Return Value

```
text

undefined
```

🧪 Basic Example

```
html

<script>
  let result = void 10;
  console.log(result);
</script>
```

✅ Output

```
text
```

```
undefined
```

✔️ `10` is evaluated

✔️ Return value is discarded

⚠️ Operator Precedence of `void`

The `void` operator has **higher precedence** than comparison operators.

Example

```
js

let res1 = void 10 === 20;
let res2 = void (10 === 20);
```

Evaluation

```
js

res1 → (void 10) === 20 → undefined === 20 → false
res2 → void (10 === 20) → void false → undefined
```

Output

```
text

false
undefined
```

📌 **Always use parentheses** when combining `void` with other operators.

🔗 What is `javascript:void(0)`?

Let's split it:

1️⃣ `javascript:` (Pseudo URL)

- Used inside `href`
- Executes JavaScript instead of navigating

Example

```
html
```

```
<a href="javascript:alert('Clicked!')">Click</a>
```

✔️ Runs JavaScript

❌ Bad practice today (security & readability)

2️⃣ `void(0)`

- `0` is evaluated
- Result is discarded
- Returns `undefined`

```
js

void(0) → undefined
```

🔥 `javascript:void(0)` (Most Common Use)

Used to **prevent page reload/navigation** when clicking a link.

Example

```
html

<a href="javascript:void(0)">Click me</a>
```

✔️ Click does nothing

✔️ No page reload

✔️ No URL change

🎯 Practical Use Case

Change page content without navigation

```
html

<a href="javascript:void(0)"
  onclick="document.body.style.backgroundColor='blue'">
  Change Background
</a>
```

✔️ JavaScript executes

✔️ Page does NOT reload

🧠 Modern Alternative (Recommended)

```
html
```

```
<a href="#" onclick="event.preventDefault()">Click</a>
```

✔️ Cleaner

✔️ Better practice

✔️ Accessible

🧩 `void` with Functions

⚠️ Important Rule

When `void` is applied to a function declaration/expression:

- Function is **evaluated**
- Result becomes **undefined**
- Function name becomes unusable

Example (Error Case)

```
html

<script>
 try {
  void function test() {
   console.log("Executed");
  };
  test(); // ❌ Error
 } catch {
  console.log("Function is undefined");
 }
</script>
```

Output

```
text

Function is undefined
```

📌 `void` treats the function as an expression and discards it.

✅ Correct Use: `void` with IIFE

Immediately Invoked Function Expression (IIFE)

```
html
```

```
<script>
  void function () {
    console.log("IIFE executed");
  }();
</script>
```

Output

```
text

IIFE executed
```

✔️ Function runs

✔️ Return value discarded

✔️ No variable pollution

🚀 `void` with Arrow Function

```
js

void (() => console.log("Arrow executed"))();
```

✔️ Executes arrow function

✔️ Returns `undefined`

🧠 Why `void` Exists?

- To **explicitly return** `undefined`
- To **avoid unwanted return values**
- To **prevent page navigation**
- To **force expression execution only**

📋 Summary Table

Usage	Purpose
`void 10`	Returns undefined
`void(expression)`	Discards result
`javascript:void(0)`	Prevent navigation
`void IIFE()`	Execute without return

Usage	Purpose
`void function()`	Makes function unusable

❌ When NOT to Use `void`

- Modern apps → use `event.preventDefault()`
- Avoid `javascript:` URLs
- Avoid confusing precedence cases

🗒️ One-Line Exam Definition

> **The `void` operator evaluates an expression and always returns `undefined`, discarding any result produced by the expression.**

If you want next:

- `delete` vs `void`
- Output-based MCQs
- Interview traps
- Modern best practices
- Comparison with `undefined`

Just say **next** 🚀