**Truthy Check**  
Yes, you're absolutely right — in JavaScript:

javascript

CopyEdit

let a = 10;

if (a) {

console.log("This is true");

}

This condition **evaluates to true** because a is 10, and 10 is a **truthy** value.

**✅ What do we call this?**

This is known as a **"truthy" check** (or sometimes called **implicit boolean coercion**).

**🔍 Explanation:**

In JavaScript, values are automatically **coerced to boolean** in conditions like if, while, etc.

* **Truthy** values (treated as true):
  + Any number **except 0** (10, -5, 3.14)
  + Non-empty strings ("hello")
  + Objects ({}, [])
  + true
* **Falsy** values (treated as false):
  + 0, NaN
  + "" (empty string)
  + null, undefined
  + false

**🔧 Console Methods for Debugging:**

| **Method** | **Description** |
| --- | --- |
| console.log() | General-purpose logging |
| console.info() | Informational messages (like log, but styled differently in some browsers) |
| console.warn() | Warnings (shows up in yellow in browser dev tools) |
| console.error() | Errors (shows up in red, often includes stack trace) |
| console.debug() | Debug-level logging (might be hidden unless dev tools are configured to show it) |
| console.trace() | Prints the stack trace at the point where it is called |
| console.dir(obj) | Displays an interactive list of properties of a specified object |
| console.table(array) | Displays tabular data as a table |
| console.assert(condition, msg) | Only logs a message if the condition is false |
| console.group() / console.groupEnd() | Creates a collapsible group of log messages |
| console.time(label) / console.timeEnd(label) | Measures time taken between the two calls |
| console.count(label) | Logs the number of times that label has been logged |
| console.clear() | Clears the console (visually) |

**💡 Example Usage:**

js

CopyEdit

console.warn("This might be a problem!");

console.error("Something went wrong!");

console.trace("Where am I called from?");

console.table([{ name: "Alice" }, { name: "Bob" }]);

console.assert(user.isLoggedIn, "User is not logged in!");

console.time("loop");

for (let i = 0; i < 1000000; i++) {}

console.timeEnd("loop");

The debugger statement is a **built-in JavaScript keyword** used to pause code execution and invoke any available debugging functionality, like browser DevTools.

**🔍 debugger Statement**

js

CopyEdit

function test(x) {

let y = x + 5;

debugger; // Execution pauses here if DevTools are open

return y \* 2;

}

test(10);

**🔧 When to Use It:**

* Insert debugger; where you want execution to pause.
* Once paused, you can inspect variables, call stack, scopes, etc.
* Works only if the browser's developer tools (like Chrome DevTools) are open.