Understanding throw Statements in JavaScript

📌 What is throw?

The throw statement in JavaScript **manually generates (throws) an error** when certain conditions are not met.
It allows you to **define custom errors** in a try...catch block.

💡 Real-Life Example:

Imagine you're **registering on a website** with a password.
If your password **doesn't meet the rules**, the website **alerts you with a custom error message** instead of breaking the system.

📌 How throw Works in try...catch

When an error is **thrown**, JavaScript **stops execution** and jumps to the catch block, handling the error.

Basic Example

```
function checkNumber(num) {
   try {
      if (num < 0) throw "Number cannot be negative!";
      if (num > 100) throw "Number cannot be greater than 100!";

      console.log("Valid number:", num);
   }
   catch (error) {
      console.log("Error:", error); // Custom error message
   }
}

checkNumber(-5);   // ❌ Error: "Number cannot be negative!"
checkNumber(150);  // ❌ Error: "Number cannot be greater than 100!"
checkNumber(50);   // ✅ Valid number: 50
```

✓ How it Works:

1️⃣If num < 0, the throw statement sends an **error message** to the catch block.

2️⃣If `num > 100`, another **custom error message** is thrown.
3️⃣If everything is **valid**, it prints the number.

📌 Validating a Password Using `throw`

Let's apply `throw` to **password validation** for a user signup form.

1️⃣ HTML Form

```
<form onsubmit="return checkPassword();">
   Enter a password<br>
   (8-12 characters, at least 1 number, no spaces)<br>
   <input type="text" id="password">
   <input type="submit" value="Submit">
</form>
```

2️⃣ JavaScript Password Validation

```
function checkPassword() {
   try {
      var pass = document.getElementById("password").value;

      // Check minimum length
      if (pass.length < 8 || pass.length > 12) {
         throw "Password must be between 8 and 12 characters.";
      }

      // Check for spaces
      if (pass.indexOf(" ") !== -1) {
         throw "No spaces allowed in the password.";
      }

      // Check for at least one number
      var numberFound = false;
      for (var i = 0; i < pass.length; i++) {
         if (!isNaN(pass[i])) { // Check if the character is a number
            numberFound = true;
            break;
         }
      }
```

```
        if (!numberFound) {
            throw "Password must contain at least one number.";
        }

        alert("Password is valid!");
        return true;
    }
    catch (error) {
        alert("Error: " + error);
        return false; // Prevent form submission
    }
}
```

✓ How it Works:

1️⃣If the password **is too short or too long**, it throws `"Password must be between 8 and 12 characters."`

2️⃣If the password **contains spaces**, it throws `"No spaces allowed in the password."`

3️⃣If the password **does not contain a number**, it throws `"Password must contain at least one number."`

4️⃣If **everything is correct**, it alerts `"Password is valid!"`

📌 What Can You `throw`?

You can throw **different types of values**:

```
throw "This is a string error"; // A string
throw 404;                     // A number
throw true;                    // A Boolean value
throw { message: "Custom error", code: 500 }; // An object
```

✓ Best Practice:

Always `throw` **strings or objects** so you can provide meaningful error messages.

📌 Using `throw` with `Error` Objects

Instead of throwing a string, it's better to use the built-in `Error` object for better debugging.

```
try {
   throw new Error("Something went wrong!");
}
catch (error) {
   console.log(error.name + ": " + error.message);
}
```

✓ Output:

```
Error: Something went wrong!
```

This is **more structured** than just throwing a string.

📌 Full Working Example with `throw`

```
<!DOCTYPE html>
<html>
<head>
   <title>Password Validator</title>
</head>
<body>

<form onsubmit="return checkPassword();">
   Enter a password:<br>
   (8-12 characters, at least 1 number, no spaces)<br>
   <input type="text" id="password">
   <input type="submit" value="Submit">
</form>

<script>
function checkPassword() {
   try {
      var pass = document.getElementById("password").value;

      if (pass.length < 8 || pass.length > 12) {
         throw new Error("Password must be between 8 and 12 characters.");
      }
```

```
        if (pass.indexOf(" ") !== -1) {
            throw new Error("No spaces allowed in the password.");
        }

        var numberFound = /\d/.test(pass); // Using regex to check for numbers
        if (!numberFound) {
            throw new Error("Password must contain at least one number.");
        }

        alert("Password is valid!");
        return true;
    }
    catch (error) {
        alert("Error: " + error.message);
        return false;
    }
}
</script>

</body>
</html>
```

✓ How it Works:

✅ If the password **meets all conditions**, it displays "Password is valid!".
❌ Otherwise, it shows **custom error messages**.

📌 Summary

Concept	Description
throw	Manually throws a custom error
try...catch	Catches and handles errors to prevent script crashes
Using Error Objects	More structured way to throw errors
