**🚀 JavaScript Math Functions & Important Methods for Automation**

JavaScript's Math object provides built-in functions for performing mathematical operations, which are useful in **test automation** for calculations, validations, random data generation, and assertions.

**📌 1. Math Object Overview**

JavaScript's Math is a built-in object with methods and properties for mathematical operations.

console.log(Math.PI); // 3.141592653589793

console.log(Math.E); // 2.718281828459045 (Euler's number)

**📌 2. Important Math Methods & Their Uses in Automation**

**🔹 A. Rounding Methods (Useful for Price & Precision Validation)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| Math.round(x) | Rounds to the nearest integer. | Math.round(4.6); // 5 |
| Math.floor(x) | Rounds **down** to the nearest integer. | Math.floor(4.9); // 4 |
| Math.ceil(x) | Rounds **up** to the nearest integer. | Math.ceil(4.1); // 5 |
| Math.trunc(x) | Removes decimal places. | Math.trunc(4.9); // 4 |

📌 **Example in Automation:**  
Checking price rounding in an **e-commerce website**:

cy.get(".price").invoke("text").then((text) => {

let displayedPrice = parseFloat(text.replace("$", ""));

let expectedPrice = Math.round(9.49);

expect(displayedPrice).to.equal(expectedPrice);

});

**🔹 B. Random Number Generation (Useful for Test Data & CAPTCHA Bypassing)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| Math.random() | Returns a random number between 0 and 1. | Math.random(); // 0.23456 |
| Math.floor(Math.random() \* x) | Generates a random integer from 0 to x-1. | Math.floor(Math.random() \* 100); |
| Math.random() \* (max - min) + min | Generates a random number in a range. | Math.random() \* (50 - 10) + 10; |

📌 **Example in Automation:**  
Generating a **random email** for **test user registration**:

let randomEmail = `user${Math.floor(Math.random() \* 10000)}@test.com`;

cy.get("#email").type(randomEmail);

**🔹 C. Absolute & Sign Methods (Useful for Numeric Validations)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| Math.abs(x) | Returns the absolute value. | Math.abs(-10); // 10 |
| Math.sign(x) | Returns 1 for positive, -1 for negative, and 0 for zero. | Math.sign(-10); // -1 |

📌 **Example in Automation:**  
Validating **discount calculations** on an e-commerce site:

cy.get("#price-before").invoke("text").then((beforeText) => {

let before = parseFloat(beforeText.replace("$", ""));

cy.get("#discount").invoke("text").then((discountText) => {

let discount = parseFloat(discountText.replace("$", ""));

cy.get("#price-after").invoke("text").then((afterText) => {

let after = parseFloat(afterText.replace("$", ""));

// Validate the discounted price

expect(Math.abs(before - discount)).to.equal(after);

});

});

});

**🔹 D. Power & Square Root (Useful for Scientific Calculations)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| Math.pow(base, exponent) | Raises a number to a power. | Math.pow(2, 3); // 8 |
| Math.sqrt(x) | Returns the square root. | Math.sqrt(16); // 4 |

📌 **Example in Automation:**  
Checking **correct exponential calculations** in an **online calculator app**:

cy.get("#input").type("2^3");

cy.get("#calculate").click();

cy.get("#result").should("have.text", Math.pow(2, 3));

**🔹 E. Min & Max Functions (Useful for Range Validations)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| Math.min(a, b, c...) | Returns the smallest number. | Math.min(10, 5, 20); // 5 |
| Math.max(a, b, c...) | Returns the largest number. | Math.max(10, 5, 20); // 20 |

📌 **Example in Automation:**  
Ensuring the **selected product price** is within a **valid range**:

cy.get("#product-price").invoke("text").then((priceText) => {

let price = parseFloat(priceText.replace("$", ""));

let minPrice = 50, maxPrice = 200;

expect(price).to.be.gte(Math.min(minPrice, maxPrice)).and.lte(Math.max(minPrice, maxPrice));

});

**📌 3. Full Example: Automating a Web-Based Calculator**

📌 **Scenario:** Automate the test case to validate mathematical operations in an **online calculator**.

**📌 Steps to Automate**

1. **Generate two random numbers** dynamically.
2. **Choose a random arithmetic operation (+, -, \*, /)**.
3. **Perform the operation** and compare it with the expected result.
4. **Handle division by zero cases**.

**🚀 Cypress Test Script (Using Random Numbers)**

describe("CalculatorSoup Automation with Random Numbers", () => {

beforeEach(() => {

cy.visit("https://www.calculatorsoup.com/calculators/math/basic.php");

});

it("Performs Random Arithmetic Operations", () => {

// Generate two random numbers between 1 and 100

const num1 = Math.floor(Math.random() \* 100) + 1;

const num2 = Math.floor(Math.random() \* 100) + 1;

const operations = ["+", "-", "\*", "/"];

const randomOp = operations[Math.floor(Math.random() \* operations.length)];

// Enter the random numbers

cy.get("#cs\_number1").clear().type(num1);

cy.get("#cs\_number2").clear().type(num2);

cy.get("select[name='op']").select(randomOp);

cy.get("input[value='Calculate']").click();

// Calculate expected result

let expectedResult;

switch (randomOp) {

case "+":

expectedResult = num1 + num2;

break;

case "-":

expectedResult = num1 - num2;

break;

case "\*":

expectedResult = num1 \* num2;

break;

case "/":

expectedResult = num2 !== 0 ? (num1 / num2).toFixed(2) : "undefined"; // Handle division by zero

break;

}

// Validate the result

cy.get("#cs\_result").invoke("text").then((resultText) => {

if (randomOp === "/") {

expect(resultText).to.include(expectedResult.toString()); // Compare as string for division

} else {

expect(parseFloat(resultText)).to.equal(expectedResult);

}

});

});

});

## ****📌 Explanation****

1. **Generates two random numbers (num1 and num2)** between 1 and 100.
2. **Selects a random operation (+, -, \*, /)**.
3. **Handles division by zero scenarios** (displays "undefined").
4. **Performs the operation, clicks "Calculate", and asserts the expected result**.

## ****🎯 Expected Output****

✅ The script runs **multiple times** with different **random inputs**.  
✅ Ensures that the calculator provides the **correct output**.

**📌 4. Summary of JavaScript Math Methods for Automation**

| **Function** | **Use Case in Automation** |
| --- | --- |
| Math.round(x) | Validate rounded prices or percentages |
| Math.random() | Generate random test data (emails, phone numbers) |
| Math.abs(x) | Compare absolute differences in pricing |
| Math.min(a, b, c...) | Ensure values stay within a valid range |
| Math.max(a, b, c...) | Validate max price limits |
| Math.pow(base, exp) | Automate scientific calculations |
| Math.floor(x) | Ensure correct number formatting |
| Math.sign(x) | Validate whether a number is positive/negative |

**🚀 Conclusion**

* JavaScript **Math functions** help automate **price validation, range checking, random data generation, and numeric calculations**.
* **Use Math.random()** to generate test data dynamically.
* **Use Math.round(), Math.floor(), and Math.ceil()** to validate UI values.
* **Use Math.abs()** for precise calculations in testing.

**These functions are useful in Cypress, and Playwright automation scripts.**