Chapter 3: Operators & Conditions

Q. What are operators? What are the types of operators in JS?

Q. What is the difference between unary, binary, and ternary operators?

Q. What is short-circuit evaluation in JS?

Q. What is operator precedence?

Q. What are the types of conditions statements in JS?

Q. When to use which type of conditions statements in real applications?

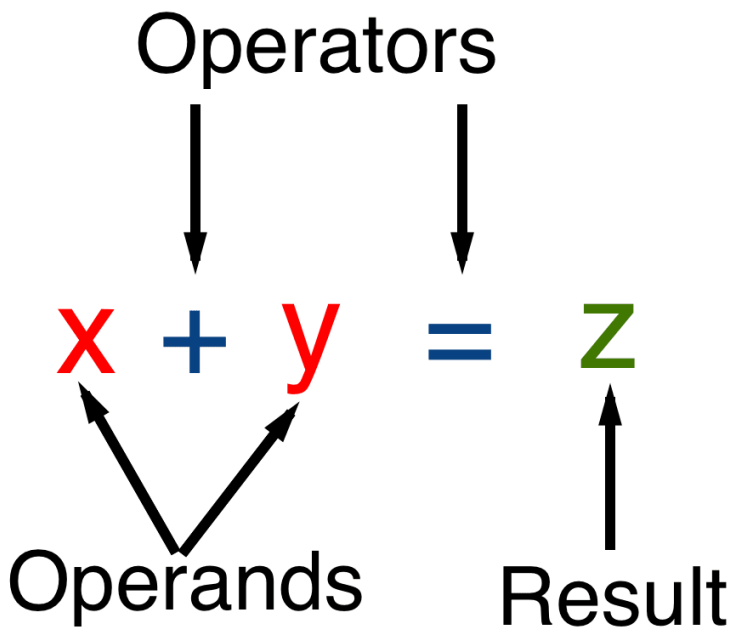
Q. What is the difference between == and ===?

Q. What is the difference between Spread and Rest operator in JS?

Back to main index

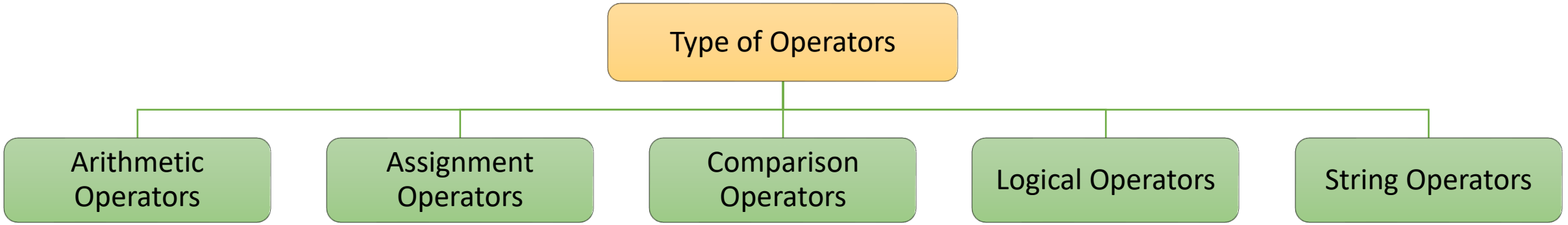
Q. What are operators? What are the types of operators in JS? V. IMP.

- Operators are **symbols or keywords** used to perform operations on operands.



back to chapter index

Q. What are operators? What are the types of operators in JS? V. IMP.




```
let x = 5;
let y = 2;

console.log(x + y);
console.log(x - y);
console.log(x * y);
console.log(x / y);
console.log(x % y);
//Remainder: 1
console.log(x ** y);
//Exponentiation: 25
```

```
let x = 10;

x += 5;
//x = x + 5
console.log(x);
// Output: 15

x *= 2;
//x = x * 2
console.log(x);
// Output: 30
```

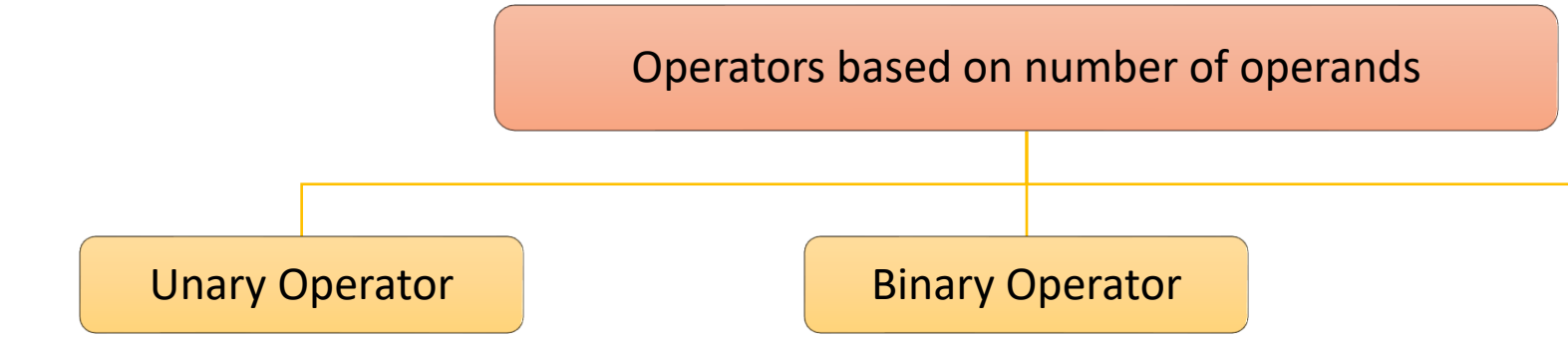
```
let x = 5;
let y = 3;

console.log(x > y);
console.log(x < y);
console.log(x >= y);
console.log(x <= y);
console.log(x === y);
// Equality: false
console.log(x !== y);
// Inequality: true
```

```
let x = true;
let y = false;

console.log(x && y);
//Logical AND: false

console.log(x || y);
//Logical OR: true

console.log(!x);
//Logical NOT: false
```

```
let a = 'Hello ';
let b = 'World';

var c =(a + b);
// Concatenation
// "Hello World"
```

back to chapter index

Q. What is the difference between unary, binary, and ternary operators?




```
let a = 5;

// Unary operator
// Single operand
let b = -a;
console.log(b);
// Output: -5
console.log(++a);
// Output: 6
```

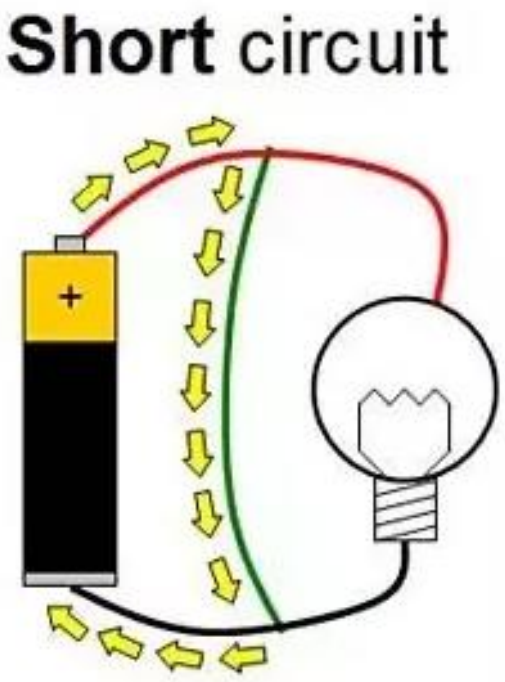
```
let x = 10;
let y = 3;

// Binary operator
// Two operands
let z = x + y;
console.log(z);
// Output: 13
```

```
// Ternary operator
// Threee operands
let result = condition ? 'Yes' : 'No';

console.log(result);
// Output: 'Yes'
```

back to chapter index

Q. What is short-circuit evaluation in JS?

- Short-circuit evaluation stops the execution as soon as the **result can be determined without evaluating** the remaining sub-expressions.



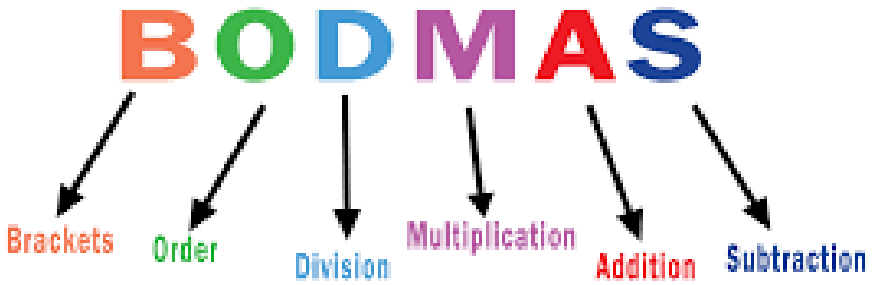

```
// Short-circuit evaluation with logical AND
let result1 = false && someFunction();
console.log(result1);
// Output: false
```

```
// Short-circuit evaluation with logical OR
let result2 = true || someFunction();
console.log(result2);
// Output: true
```

back to chapter index

Q. What is operator precedence?

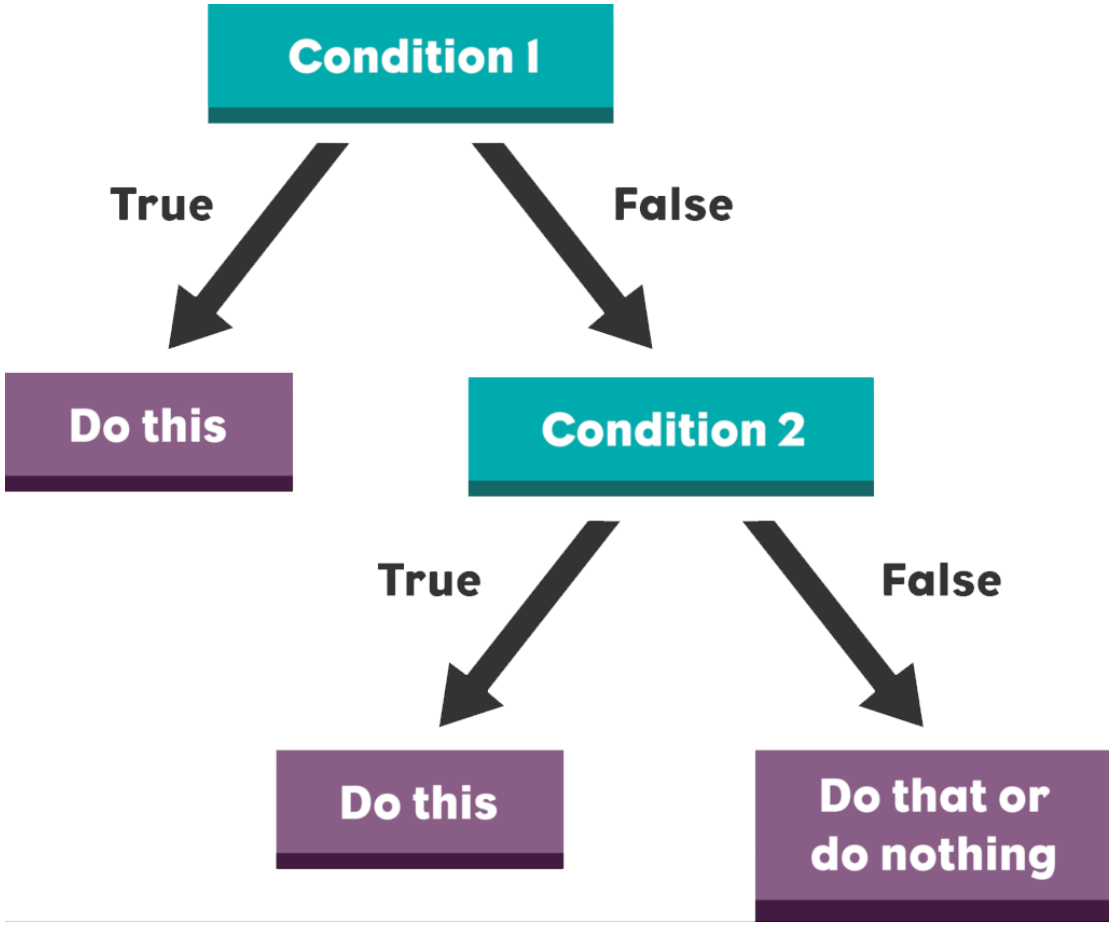
- As per operator precedence, operators with higher precedence are **evaluated first.**




```
let a = 6;
let b = 3;
let c = 2;

//BracketOf-Division-Multiplication-Add-Sub
let result = a + b * c + (a - b);

console.log(result);
// Output: 15
```

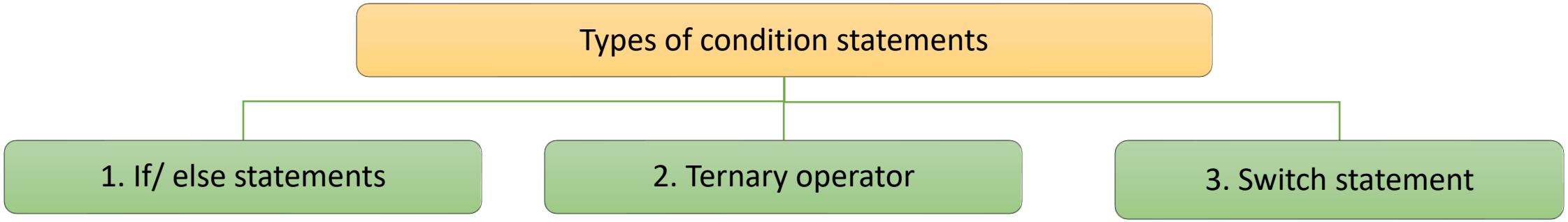
back to chapter index

Q. What are the types of conditions statements in JS? V. IMP.



back to chapter index

Q. What are the types of conditions statements in JS? V. IMP.




```
let x = 5;

if (x > 10) {
  console.log("1");
} else if (x < 5) {
  console.log("2");
} else {
  console.log("3");
}
// Output: '3'
```

```
let y = 20;
let z = y > 10 ? "1" : "0"
console.log(z);
// Output: '1'
```

```
let a = 5;
switch (a) {
  case 1:
    console.log("1");
    break;
  case 5:
    console.log("2");
    break;
  default:
    console.log("3");
}
// Output: '2'
```

back to chapter index

Q. When to use which type of conditions statements in real applications? V. IMP.

- If…else : for complex, different & multiline execution.
- Benefit: Cover all scenarios.

```
const age = 25;
const height = 6

if (age < 25 && height < 5) {
  console.log("You are a minor.");
  console.log("You are a short.");
} else if (age >= 18 && height > 6) {
  console.log("You are an adult.");
  console.log("You are an tall.");
} else {
  console.log("You are average");
}
// Output: "You are average"
```

- Ternary operators : for simple conditions & single value evaluations.
- Benefit: Short one line syntax.

```
const isUser = true;

const user = isUser ? 10 : 20;

console.log(user);
// Output: "10"
```

- Switch case: For same left side values.
- Benefit: More structured code.

```
const dayOfWeek = "Tuesday";

switch (dayOfWeek) {
  case "Monday":
    console.log("Start ");
    break;
  case "Tuesday":
  case "Sunday":
    console.log("Weekend!");
    break;
  default:
    console.log("Invalid");
}
// Output: "Weekend!"
```

back to chapter index

Q. What is the difference between == and ===? V. IMP.

```
//Loose Equality
console.log(1 == '1');
console.log(true == 1);
// Output: true
```

- Loose Equality (==) operator compares two values for equality after performing **type coercion**

```
//Strict Equality
console.log(1 === '1');
console.log(true === 1);
// Output: false
```

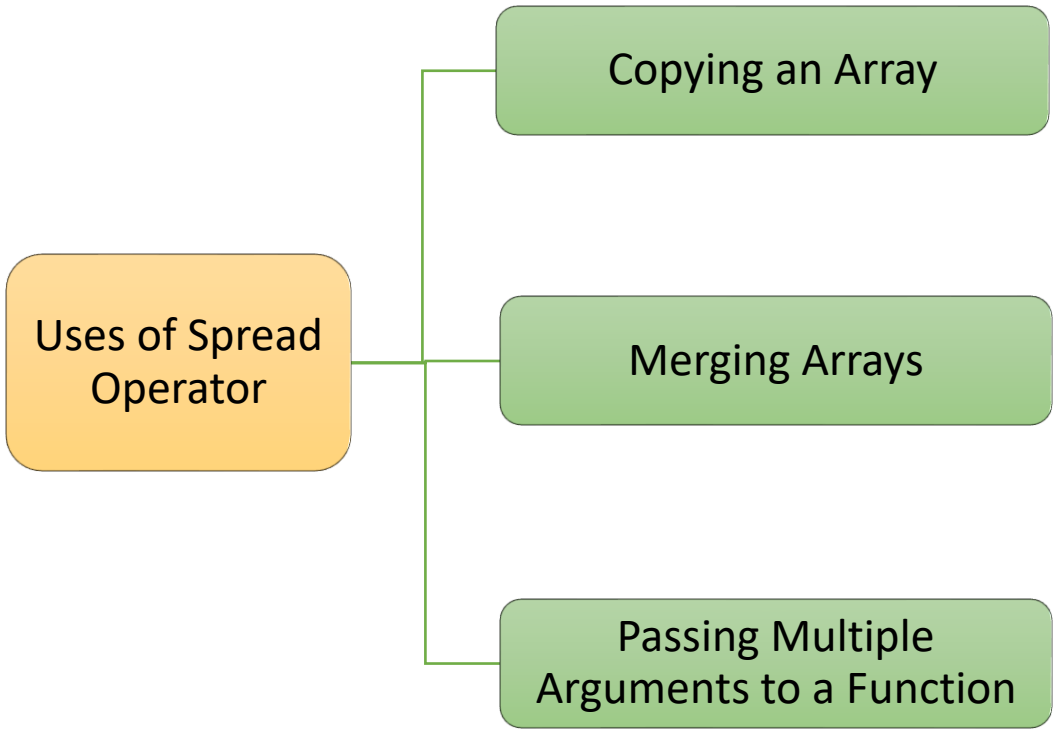
- Strict Equality (===) operator compares two values for equality **without** performing type coercion.

- Normally **=== is preferred** in use to get more accurate comparisons.

back to chapter index

Q. What is the difference between Spread and Rest operator in JS?

- The spread operator(...) is used to **expand or spread elements** from an iterable (such as an array, string, or object) into individual elements.




```
// Spread Operator Examples
const array = [1, 2, 3];
console.log(...array); // Output: 1, 2, 3
```

```
// Copying an array
const originalArray = [1, 2, 3];
const copiedArray = [...originalArray];
console.log(copiedArray); // Output: [1, 2, 3]
```

```
// Merging arrays
const array1 = [1, 2, 3];
const array2 = [4, 5];
const mergedArray = [...array1, ...array2];
console.log(mergedArray); // Output: [1, 2, 3, 4, 5]
```

```
// Passing multiple arguments to a function
const numbers = [1, 2, 3, 4, 5];
sum(...numbers);
function sum(a, b, c, d, e) {
  console.log(a + b + c + d + e); //Output: 15
}
```

back to chapter index

Q. What is the difference between Spread and Rest operator in JS?

- The rest operator is used in function parameters to collect all **remaining arguments** into an array.

```
// Rest Operator Example
display(1, 2, 3, 4, 5);

function display(first, second, ...restArguments) {
  console.log(first); // Output: 1
  console.log(second); // Output: 2

  console.log(remaining); // Output: [3, 4, 5]
}
```

back to chapter index