Console Logs

To write message in console window. Console logs help developers see what's happening in their code in real-time.

Syntax

```
console.log(message);
```

Example

```
console.log("Hello, World!");
console.log(42);
console.log(1+2)
console.log("Hello", "how are you", num)
```

Output appears in the browser's developer console.

Linking JS File to HTML

To use JavaScript in your HTML document, you need to link the JS file using the ‹script› tag.

Syntax

```
<script src="filename.js"></script>
```

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>My Page</title>
</head>
<body>
   <h1>Hello World</h1>

   <script src="script.js"></script>
</body>
</html>
```

Place the ‹script› tag at the end of the ‹body› for better page load performance.

Template Literals

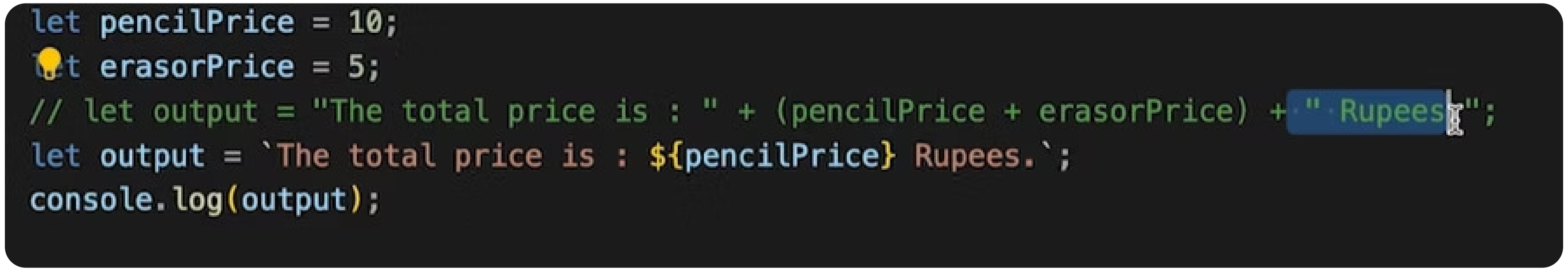
They are used to add embedded expressions in a string.

Syntax

```
string text ${expression} string text
```

Example

```
let a = 5;
let b = 10;

console.log(`Your pay ${a + b} rupees`);
// Output: Your pay 15 rupees
```



```
let pencilPrice = 10;
let erasorPrice = 5;
// let output = "The total price is : " + (pencilPrice + erasorPrice) + " Rupees";
let output = `The total price is : ${pencilPrice} Rupees.`;
console.log(output);
```

Template literals use backticks (`) instead of quotes and allow variables and expressions inside ${}.

Operators in JS

Operators are used to perform operations on variables and values.

Arithmetic Operators

```
let x = 10, y = 5;
console.log(x + y);  // 15
console.log(x - y);  // 5
console.log(x * y);  // 50
console.log(x / y);  // 2
console.log(x % y);  // 0
```

Unary Operators

```
let a = 5;
console.log(++a);  // 6 (increment)
console.log(--a);  // 5 (decrement)
```

Assignment Operators

```
let b = 10;
b += 5;  // b = b + 5 (15)
b -= 3;  // b = b - 3 (12)
b *= 2;  // b = b * 2 (24)
```

Comparison Operators

Comparison operators compare two values and return a boolean result (true or false).

Common Comparison Operators

```
== (Equal to)
=== (Strict equal to)
!= (Not equal to)
!== (Strict not equal to)
> (Greater than)
< (Less than)
>= (Greater than or equal to)
<= (Less than or equal to)
```

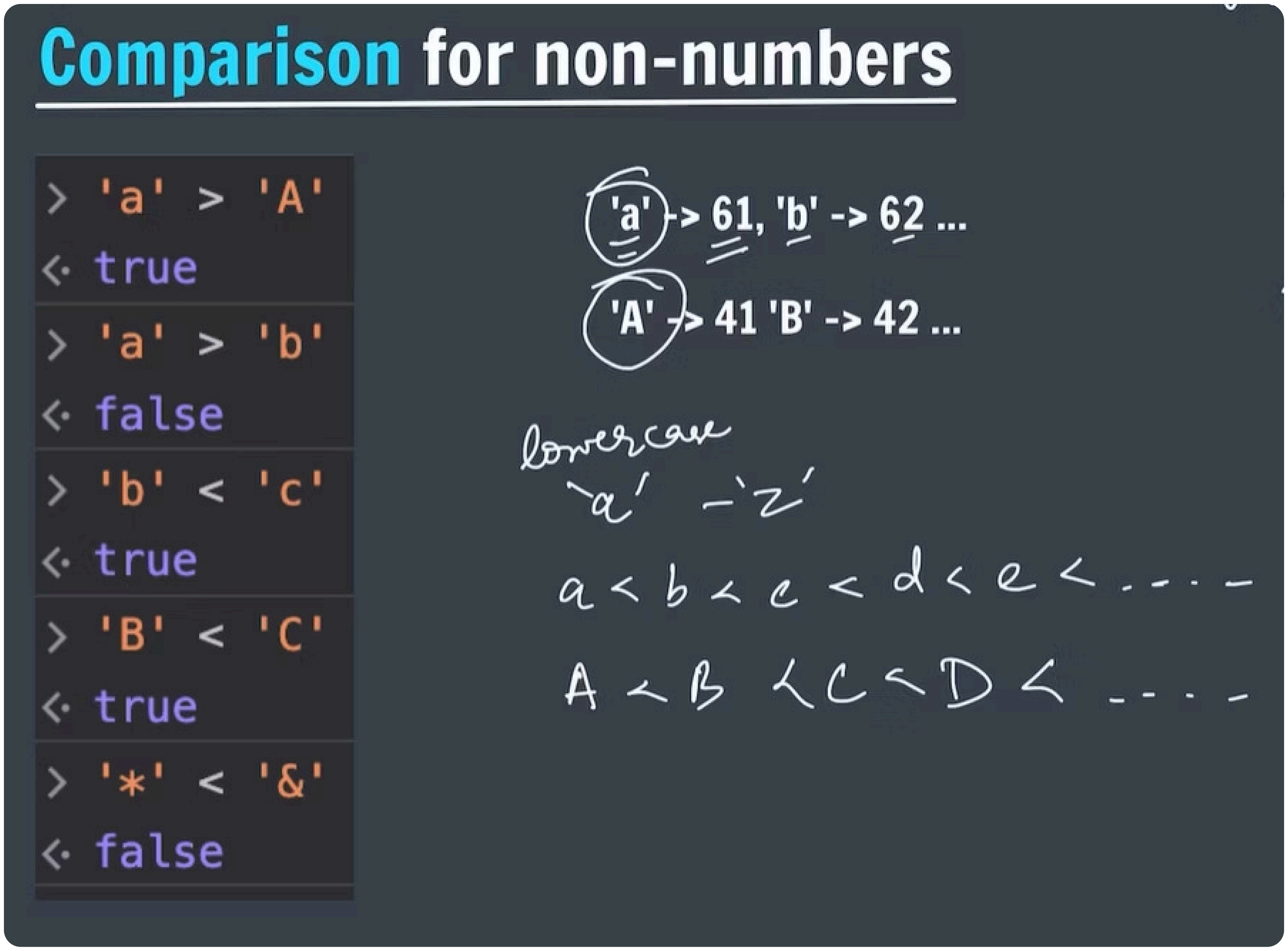
Examples

```
let x = 10;
let y = "10";

console.log(x == y);   // true (compares value only)
console.log(x === y);  // false (compares value and type)
console.log(x != y);   // false
console.log(x !== y);  // true
console.log(x > 5);    // true
console.log(x < 20);   // true
```

Use `===` (strict equality) instead of `==` to avoid type coercion issues.

Comparison for Non-Numbers

JavaScript can compare strings using comparison operators. It compares them based on their Unicode values (lexicographic order).



Comparison for non-numbers

```
> 'a' > 'A'
<· true
> 'a' > 'b'
<· false
> 'b' < 'c'
<· true
> 'B' < 'C'
<· true
> '*' < '&'
<· false
```

'a' -> 61, 'b' -> 62 ...

'A' -> 41 'B' -> 42 ...

lowercase
'a' -'z'

a < b < c < d < e < . . . . -

A < B < C < D < . . . -

How It Works

Strings are compared character by character from left to right. Lowercase letters have higher Unicode values than uppercase letters.

Examples

```
console.log('a' > 'A');    // true
console.log('a' > 'b');    // false
console.log('b' < 'c');    // true
console.log('B' < 'C');    // true
console.log('x' > 'X');    // false
```

For alphabetical comparison: `a < b < c < d...` and `A < B < C < D...`

Conditional Statements in JS

Conditional statements allow you to execute different code based on different conditions.

Types of Conditional Statements

if statement

Executes code if a condition is true.

if-else statement

Executes one block if true, another if false.

else-if statement

Tests multiple conditions sequentially.

switch statement

Selects one of many code blocks to execute.

if Statement Syntax

```
if (condition) {
  // code to execute if condition is true
}
```

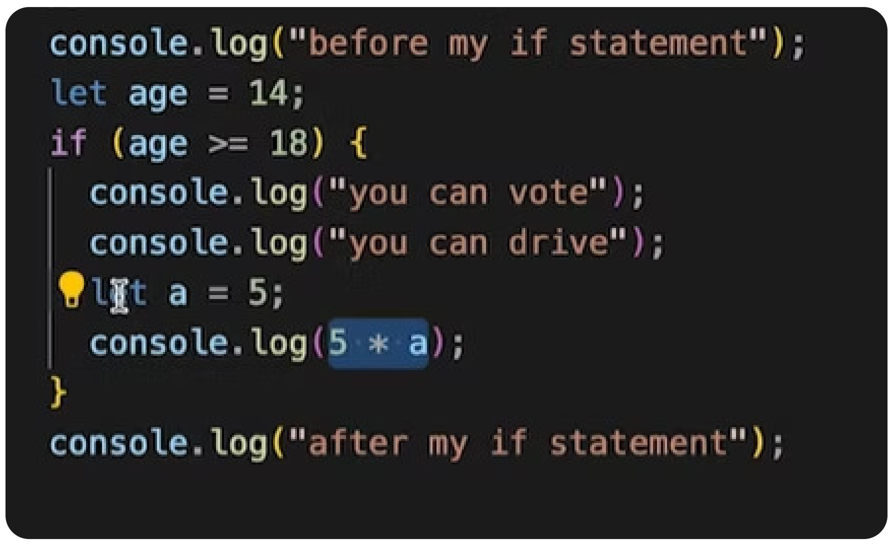
if-else Example

```
let age = 18;

if (age >= 18) {
  console.log("You are an adult");
} else {
  console.log("You are a minor");
}
```

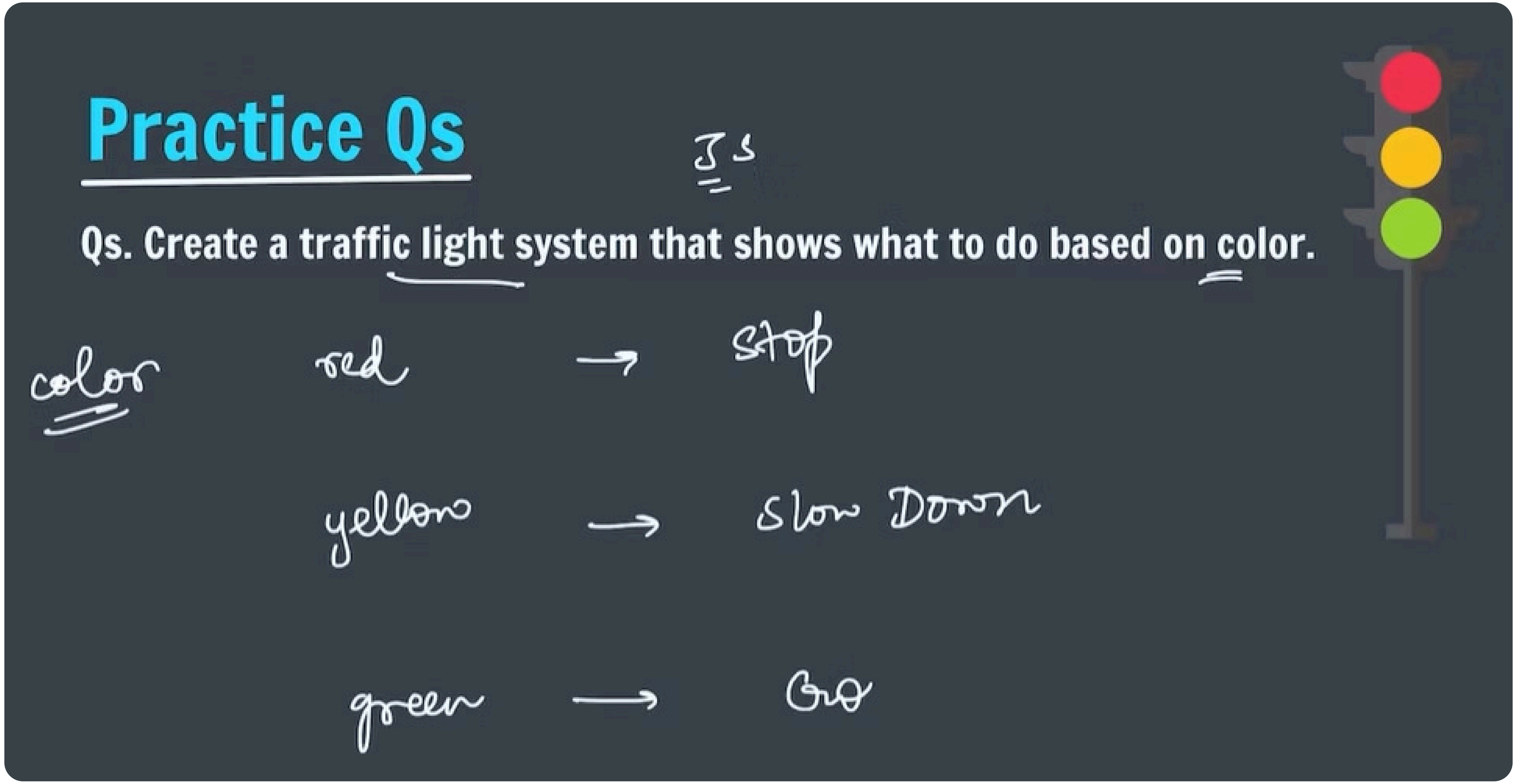
else-if Example

```
let score = 85;

if (score >= 90) {
  console.log("Grade: A");
} else if (score >= 80) {
  console.log("Grade: B");
} else if (score >= 70) {
  console.log("Grade: C");
} else {
  console.log("Grade: F");
}
```

Example



```
console.log("before my if statement");
let age = 14;
if (age >= 18) {
  console.log("you can vote");
  console.log("you can drive");
  let a = 5;
  console.log(5 * a);
}
console.log("after my if statement");
```



Practice Qs

JS

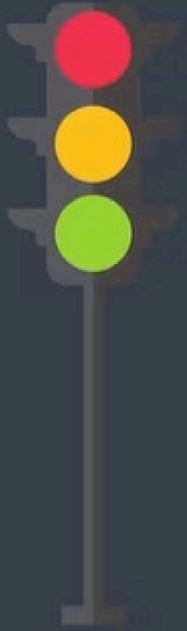
Qs. Create a traffic light system that shows what to do based on color.

color

$$\text{red} \longrightarrow \text{stop}$$

$$\text{yellow} \longrightarrow \text{Slow Down}$$

$$\text{green} \longrightarrow \text{Go}$$



Practice Qs

Qs. Create a system to calculate popcorn prices based on the size customer asked for :

if size is 'XL', price is Rs. 250

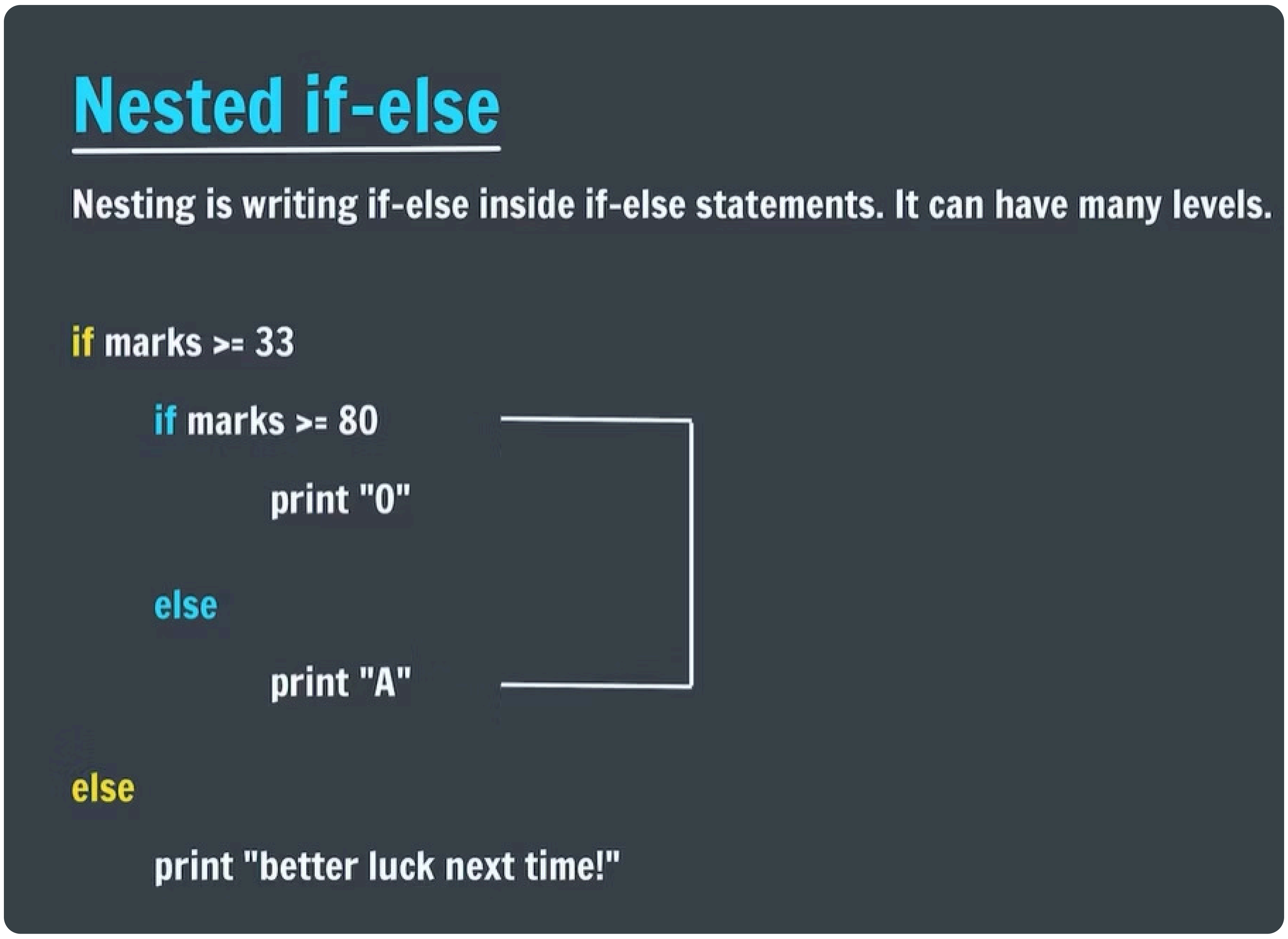
if size is 'L', price is Rs. 200

if size is 'M', price is Rs. 100

if size is 'S', price is Rs. 50

Nested if-else

Nesting is writing if-else inside if-else statements. It can have many levels.


Nested if-else

Nesting is writing if-else inside if-else statements. It can have many levels.

```
if marks >= 33
    if marks >= 80
            print "O"
    else
            print "A"
else
    print "better luck next time!"
```

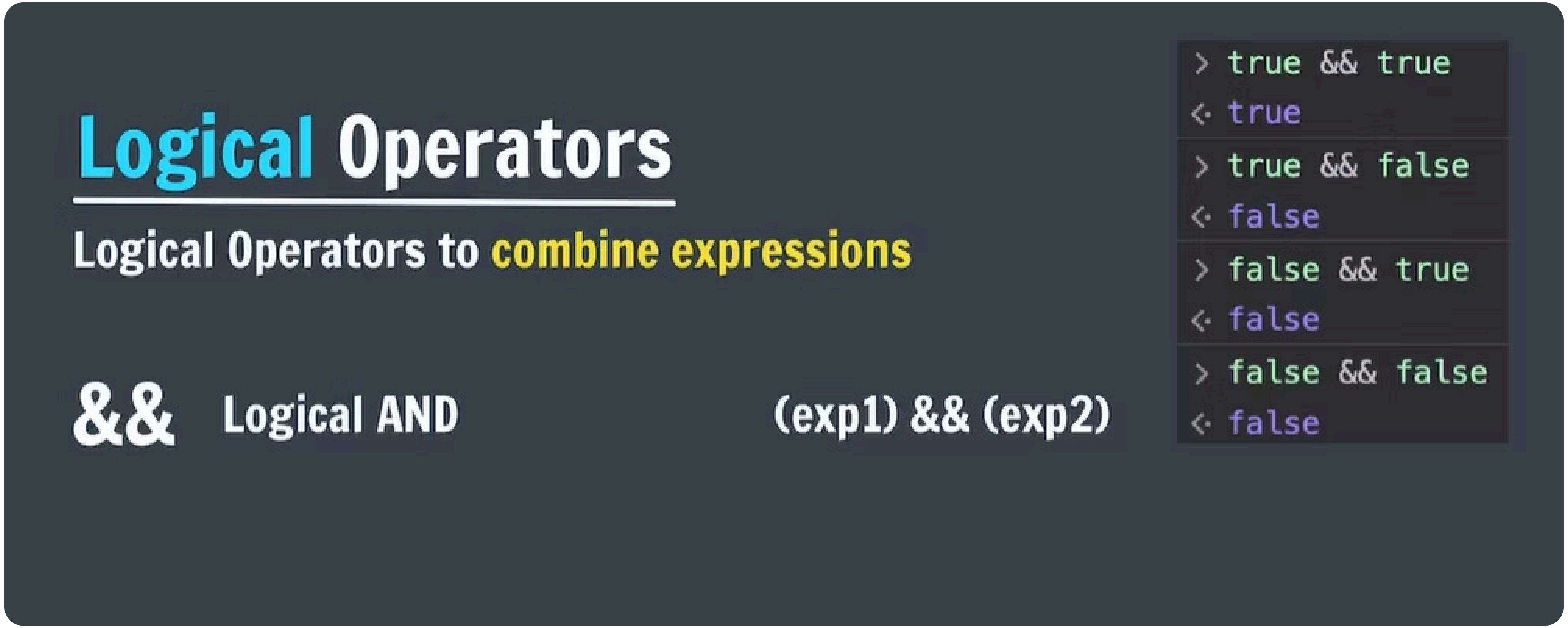
Syntax

```
if (condition1) {
   if (condition2) {
      // code if both conditions are true
   } else {
      // code if condition1 is true but condition2 is false
   }
} else {
   // code if condition1 is false
}
```

Example

```
if (marks >= 33) {
   if (marks >= 80) {
      console.log("O");
   } else {
      console.log("A");
   }
} else {
   console.log("Better luck next time!");
}
```

Nested if-else statements allow you to check multiple conditions at different levels.

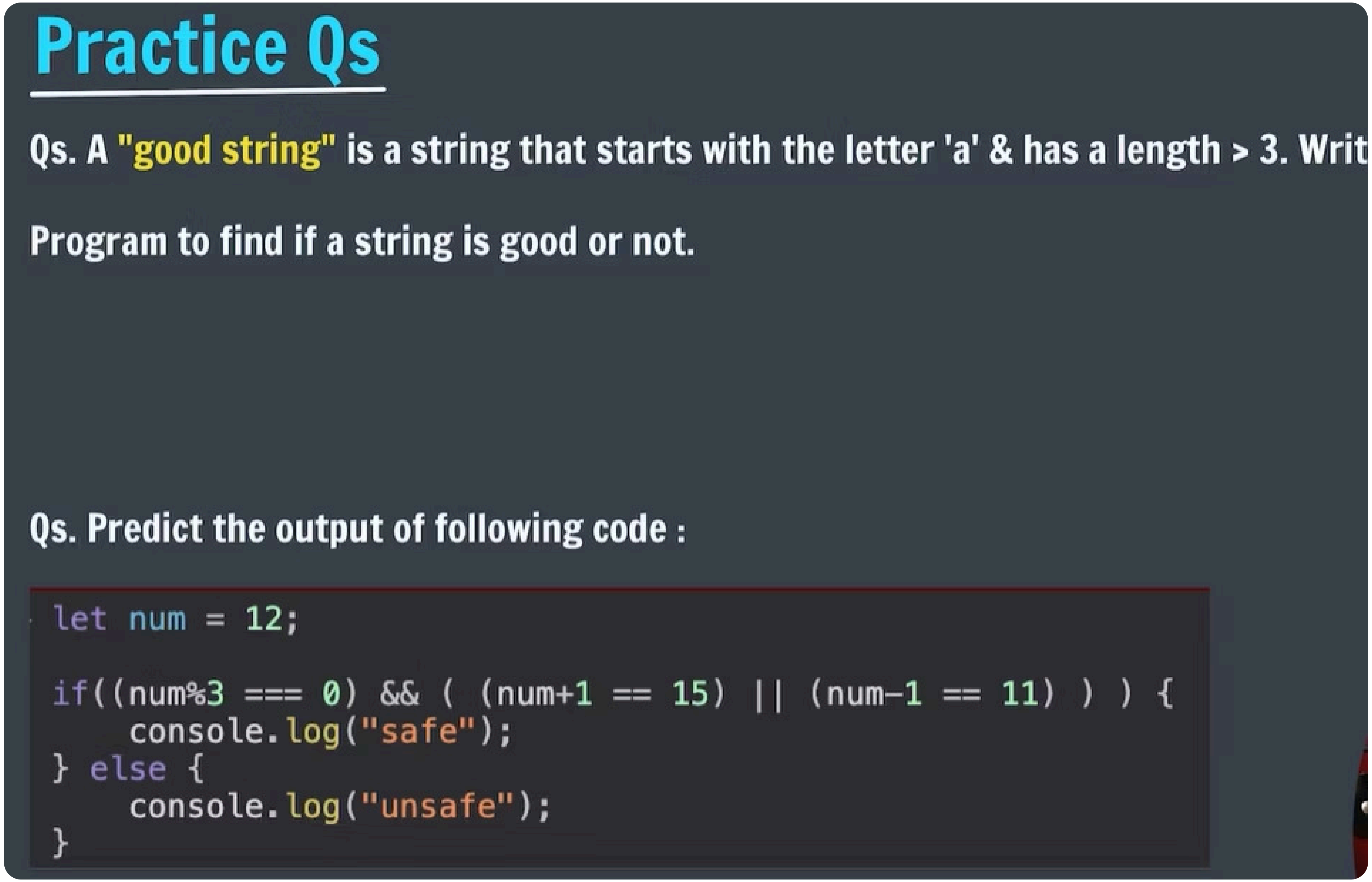


Logical Operators

Logical Operators to **combine expressions**

&& Logical AND (exp1) && (exp2)

```
> true && true
<· true
> true && false
<· false
> false && true
<· false
> false && false
<· false
```



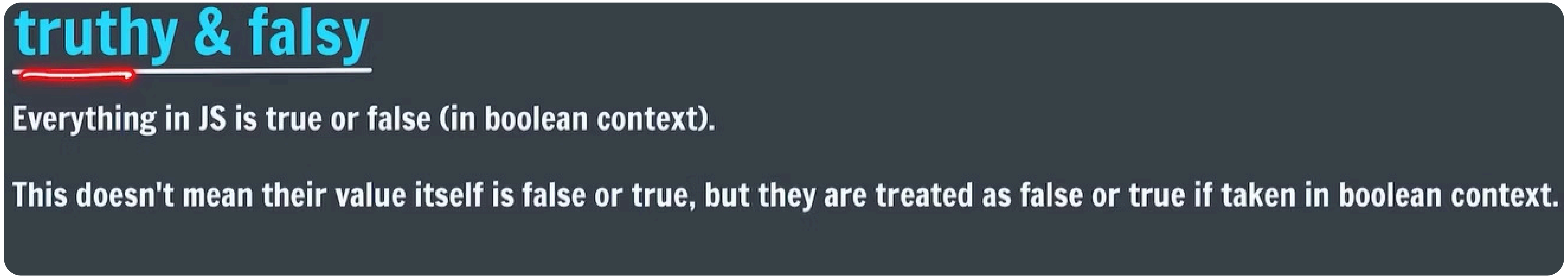
Practice Qs

Qs. A "good string" is a string that starts with the letter 'a' & has a length > 3. Writ

Program to find if a string is good or not.

Qs. Predict the output of following code :

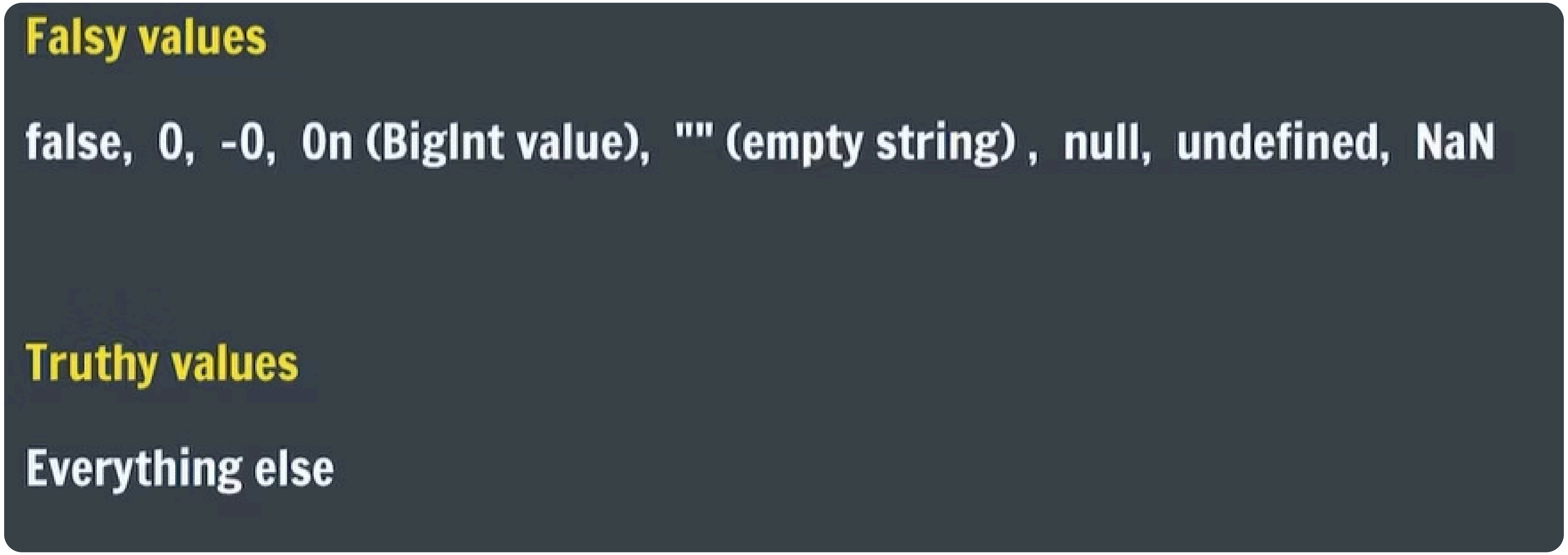
```
let num = 12;

if((num%3 === 0) && ( (num+1 == 15) || (num-1 == 11) ) ) {
    console.log("safe");
} else {
    console.log("unsafe");
}
```



truthy & falsy

Everything in JS is true or false (in boolean context).

This doesn't mean their value itself is false or true, but they are treated as false or true if taken in boolean context.

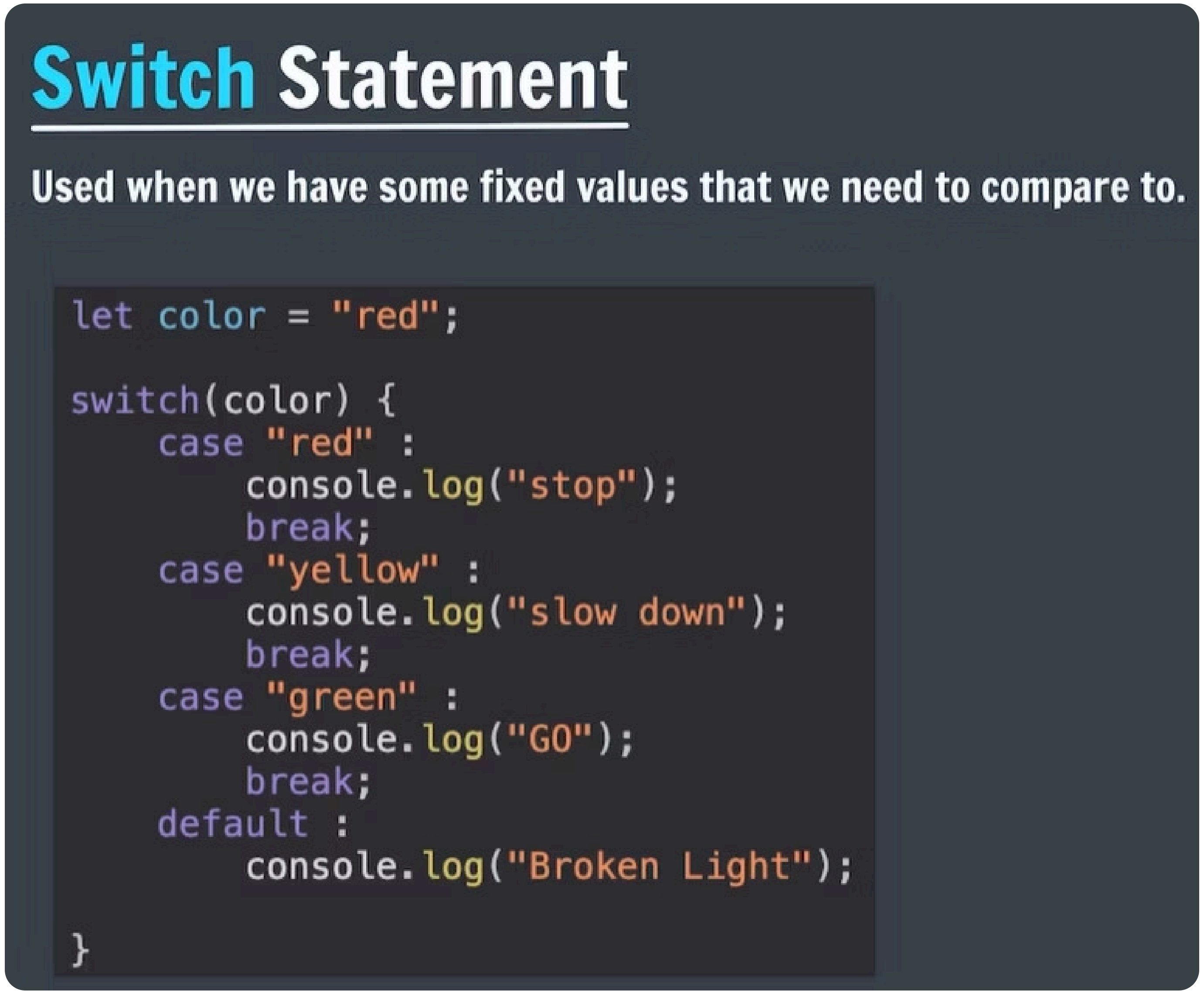


Falsy values

false, 0, -0, 0n (BigInt value), "" (empty string) , null, undefined, NaN

Truthy values

Everything else



Switch Statement

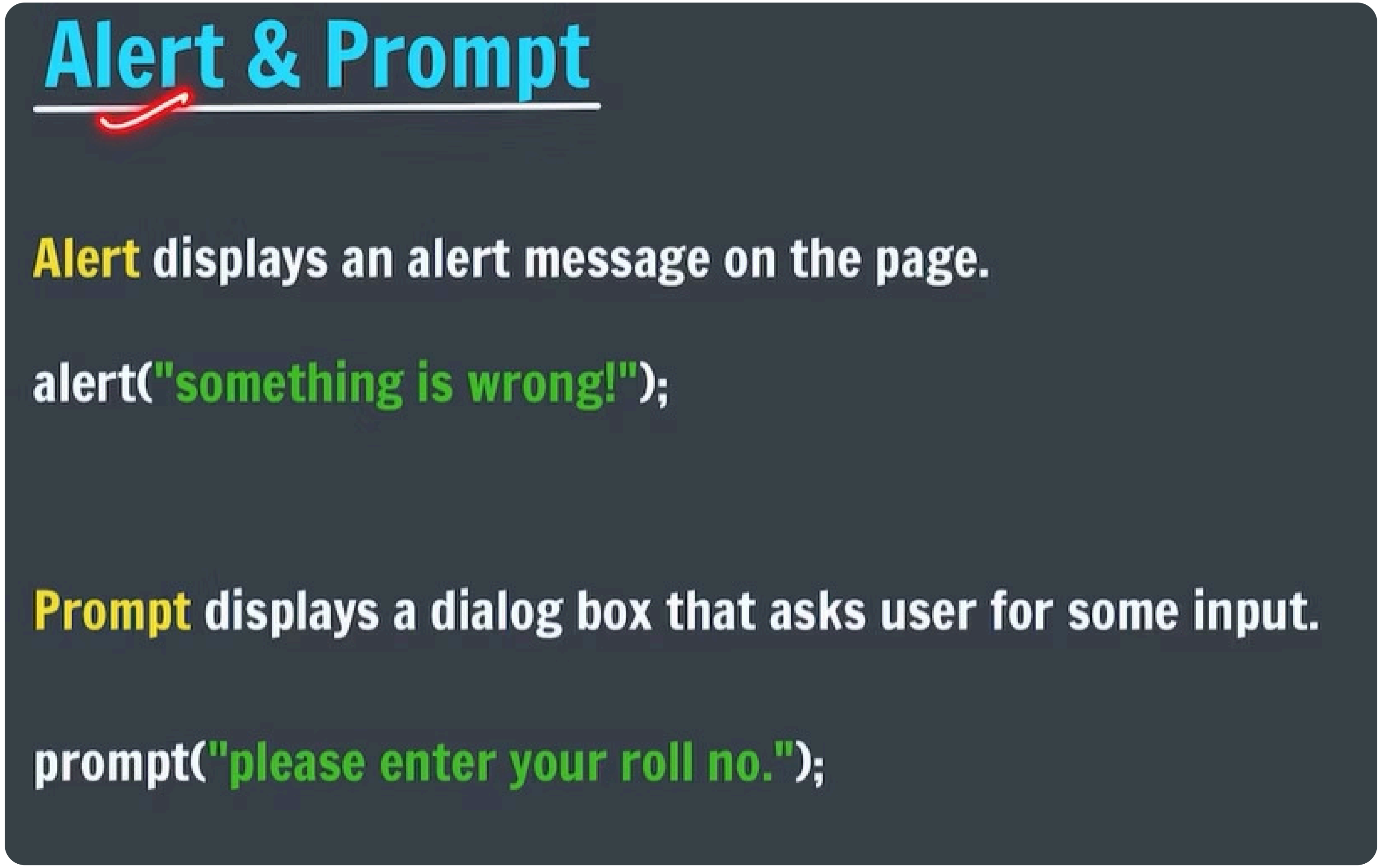
Used when we have some fixed values that we need to compare to.

```
let color = "red";

switch(color) {
    case "red" :
        console.log("stop");
        break;
    case "yellow" :
        console.log("slow down");
        break;
    case "green" :
        console.log("GO");
        break;
    default :
        console.log("Broken Light");

}
```

Practice Qs

Qs. Use switch statement to print the day of the week using a number variable 'day' with values 1 to 7.

1 = Monday, 2 = Tuesday & so on



Alert & Prompt

Alert displays an alert message on the page.

```
alert("something is wrong!");
```

Prompt displays a dialog box that asks user for some input.

```
prompt("please enter your roll no.");
```