LOOPS & CONDITIONAL STATEMENTS

TOPICS OUTLINE

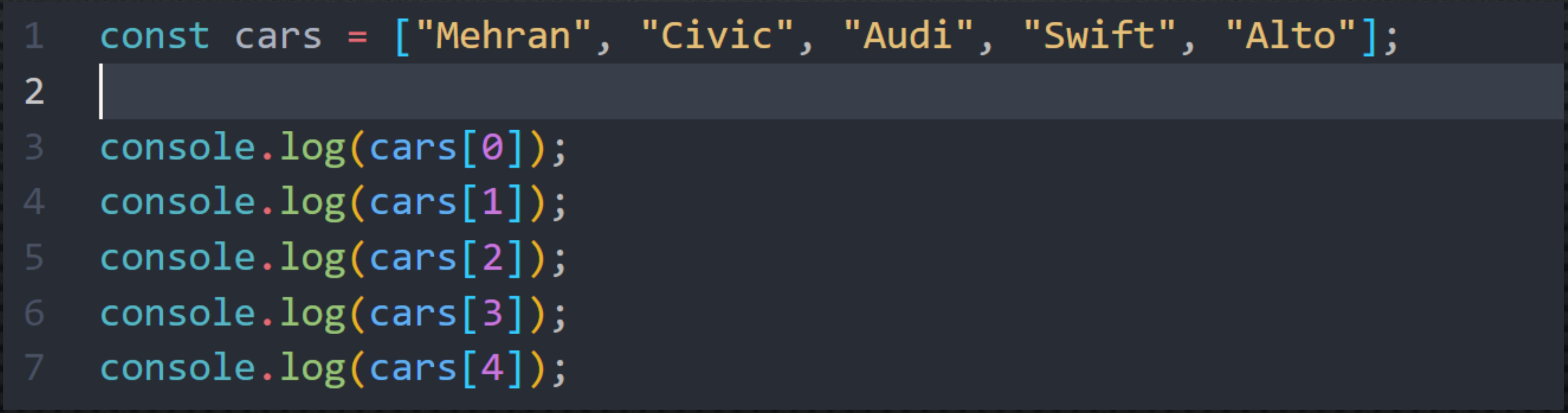
- Importance/Advantages of topic.
- Introduction to topic.
- Topic Explanation using Real World Scenario
- Coding Examples.
- Questions.

LOOPS & CONDITIONAL STATEMENTS

- We going to learn today **Loops and Conditional Statements**.

LOOPS

- LOOPS CAN EXECUTE A BLOCK OF CODE A NUMBER OF TIMES.
- IF YOU WANT TO RUN THE SAME CODE OVER AND OVER AGAIN, EACH TIME WITH A DIFFERENT VALUE.
- OFTEN THIS IS THE CASE WHEN WORKING WITH ARRAYS:



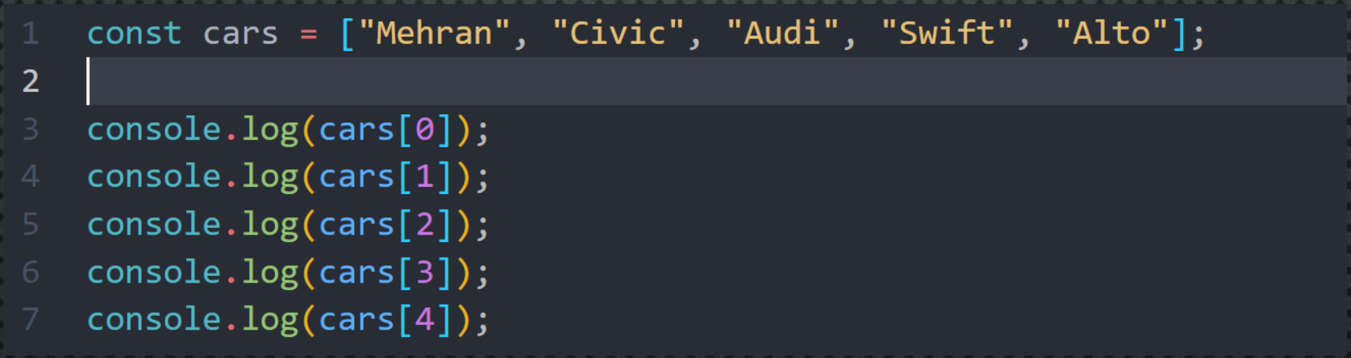
```
const cars = ["Mehran", "Civic", "Audi", "Swift", "Alto"];

console.log(cars[0]);
console.log(cars[1]);
console.log(cars[2]);
console.log(cars[3]);
console.log(cars[4]);
```

COMPARISON BETWEEN TRADITIONAL APPROACH AND MODERN APPROACH.

Loops

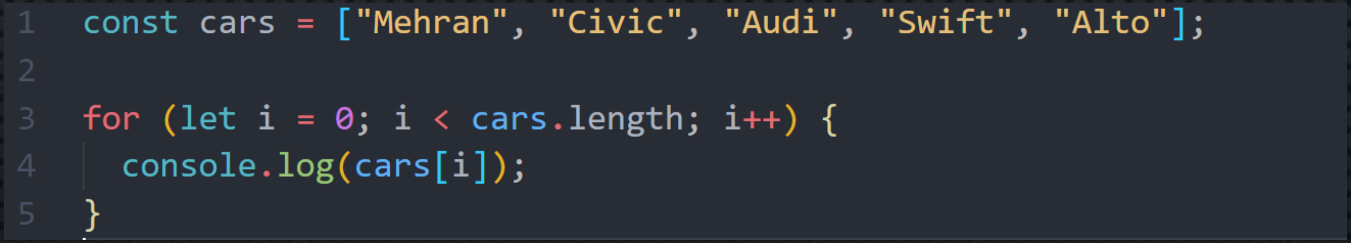
- Traditional Approach



```
const cars = ["Mehran", "Civic", "Audi", "Swift", "Alto"];

console.log(cars[0]);
console.log(cars[1]);
console.log(cars[2]);
console.log(cars[3]);
console.log(cars[4]);
```

- Modern Approach



```
const cars = ["Mehran", "Civic", "Audi", "Swift", "Alto"];

for (let i = 0; i < cars.length; i++) {
  console.log(cars[i]);
}
```

ADVANTAGES OF MODERN APPROACH.

Loops

- Helps to ignore **writing one statements** many times.
- Helps to **reduce many lines of codes**.
- Make program **less complex.**

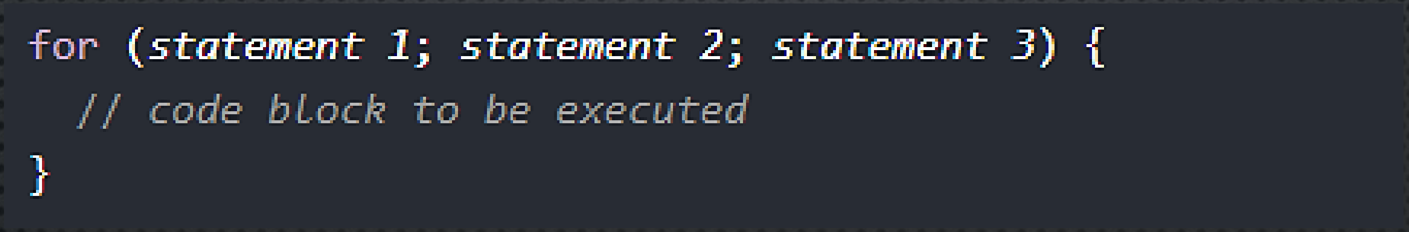
DIFFERENT KIND OF LOOPS:

WE HAVE DIFFERENT KIND OF LOOPS

- **FOR LOOP**
- **WHILE LOOP**
- **DO WHILE LOOP**

THE FOR LOOP

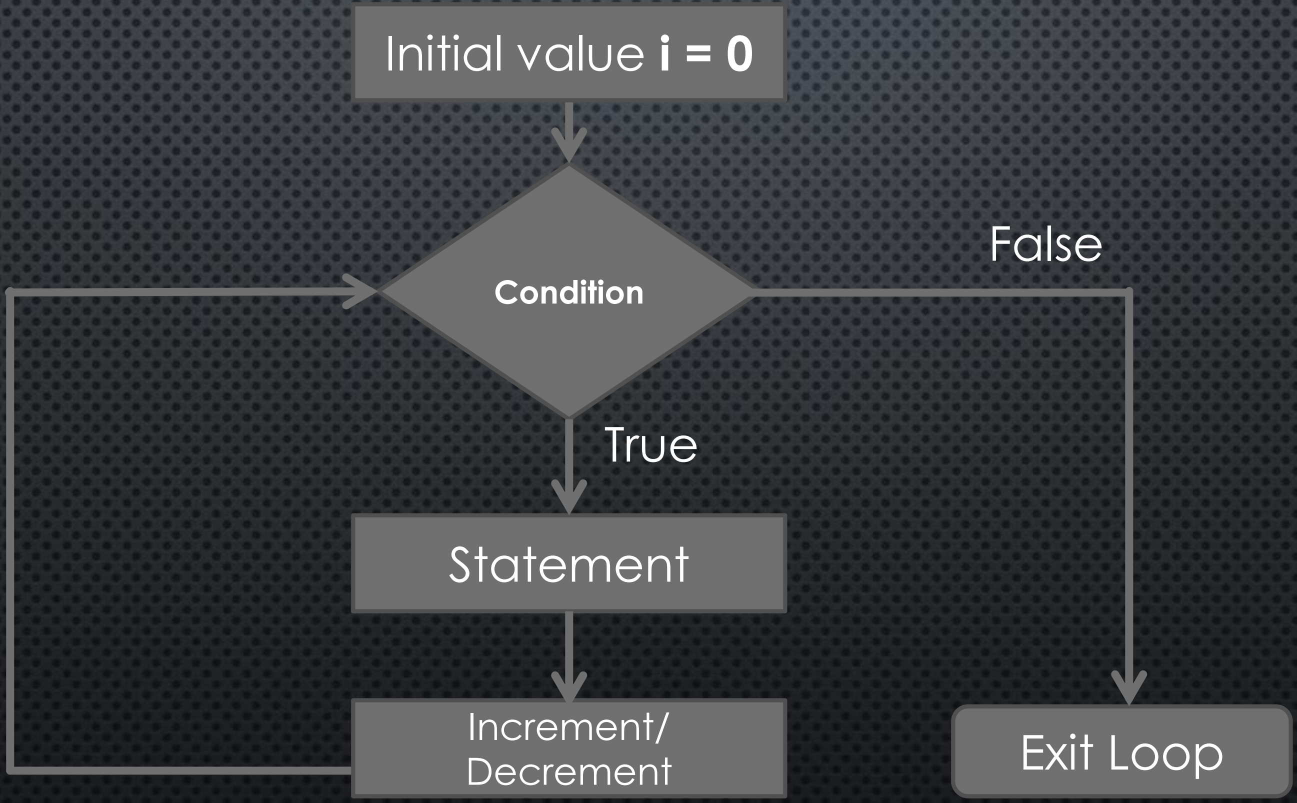
- **FOR LOOP** THROUGH BLOCK OF CODE A NUMBER OF TIMES.
- THE **FOR LOOP** HAS THE FOLLOWING SYNTAX:



```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

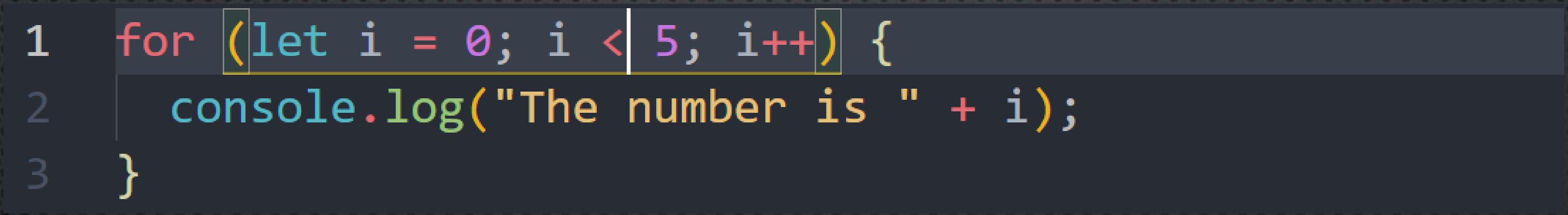
- **STATEMENT 1** IS EXECUTED (ONE TIME) BEFORE THE EXECUTION OF THE CODE BLOCK.
- **STATEMENT 2** DEFINES THE CONDITION FOR EXECUTING THE CODE BLOCK.
- **STATEMENT 3** IS EXECUTED (EVERY TIME) AFTER THE CODE BLOCK HAS BEEN EXECUTED.

THE FOR LOOP GRAPHICAL PRESENTATION



THE FOR LOOP

- **FOR LOOP** THROUGH BLOCK OF CODE A NUMBER OF TIMES.
- **EXAMPLE:**



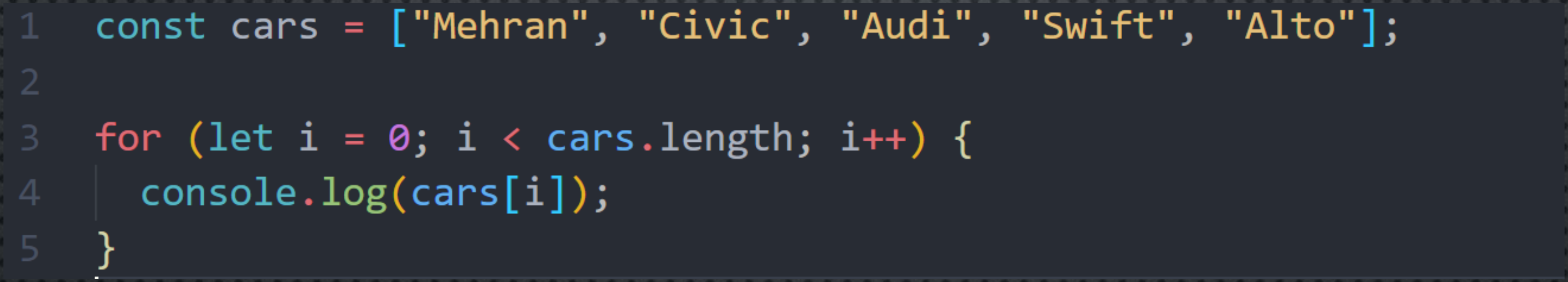
```
for (let i = 0; i < 5; i++) {
  console.log("The number is " + i);
}
```

FROM THE EXAMPLE ABOVE, YOU CAN READ:

- **STATEMENT 1** SETS A VARIABLE BEFORE THE LOOP STARTS (LET I = 0).
- **STATEMENT 2** DEFINES THE CONDITION FOR THE LOOP TO RUN (I MUST BE LESS THAN 5).
- **STATEMENT 3** INCREASES A VALUE (I++) EACH TIME THE CODE BLOCK IN THE LOOP HAS BEEN EXECUTED.

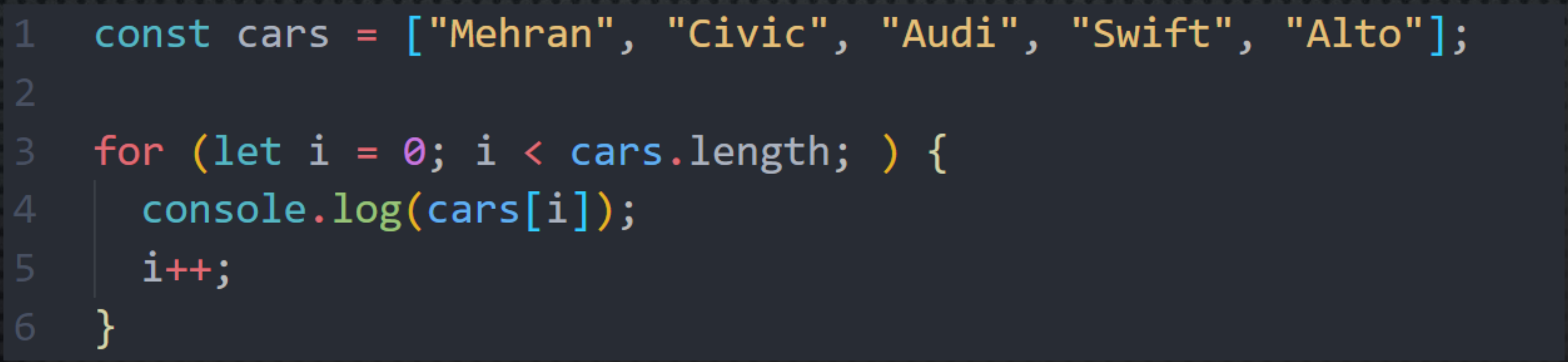
THE FOR LOOP

Code Example



```javascript
const cars = ["Mehran", "Civic", "Audi", "Swift", "Alto"];

for (let i = 0; i < cars.length; i++) {
  console.log(cars[i]);
}
```

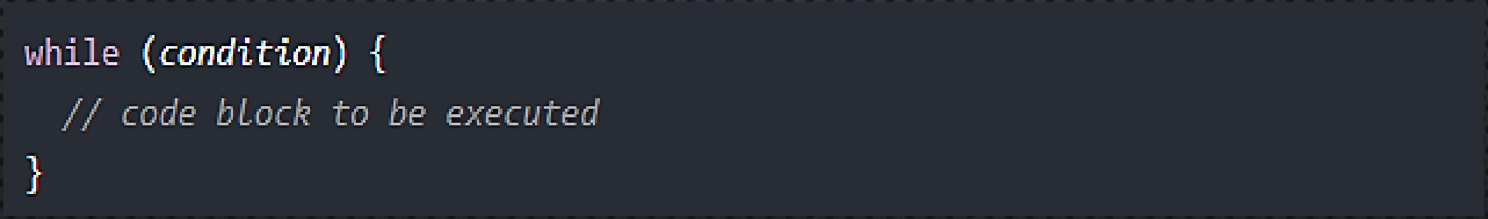


```javascript
const cars = ["Mehran", "Civic", "Audi", "Swift", "Alto"];

for (let i = 0; i < cars.length; ) {
  console.log(cars[i]);
  i++;
}
```

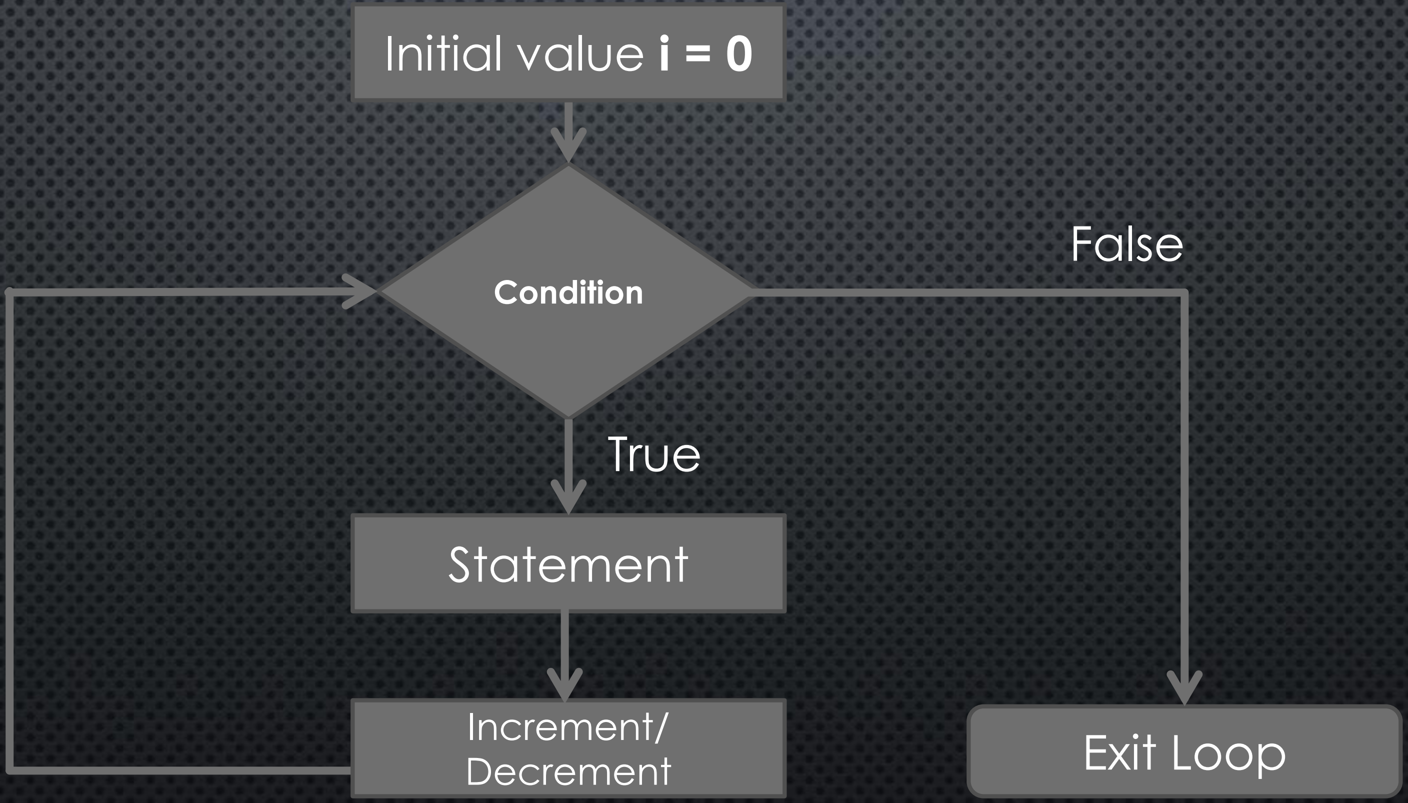
THE WHILE LOOP

- THE **WHILE LOOP** THROUGH A BLOCK OF CODE AS LONG AS A SPECIFIED **CONDITION IS TRUE**.
- THE **WHILE LOOP** HAS THE FOLLOWING SYNTAX:



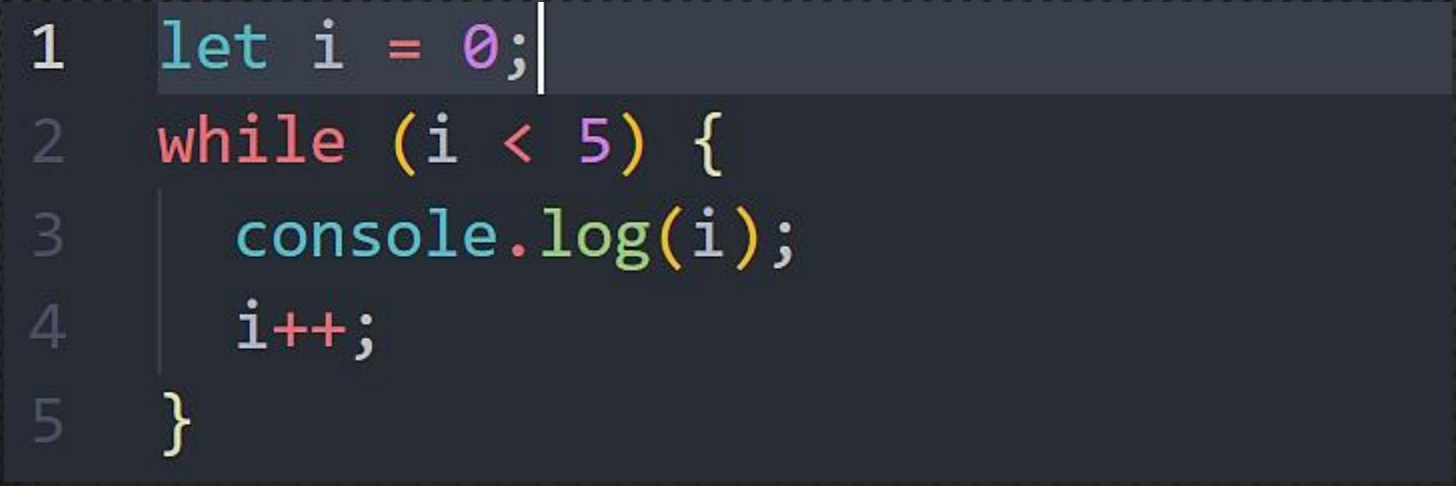
```
while (condition) {
  // code block to be executed
}
```

THE WHILE LOOP GRAPHICAL PRESENTATION



THE WHILE LOOP

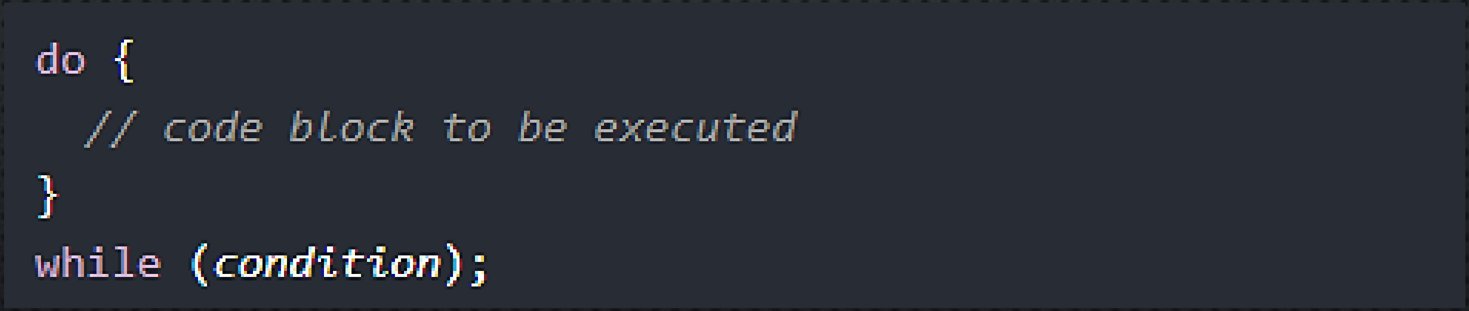
<u>**Code Example**</u>



```
let i = 0;
while (i < 5) {
  console.log(i);
  i++;
}
```

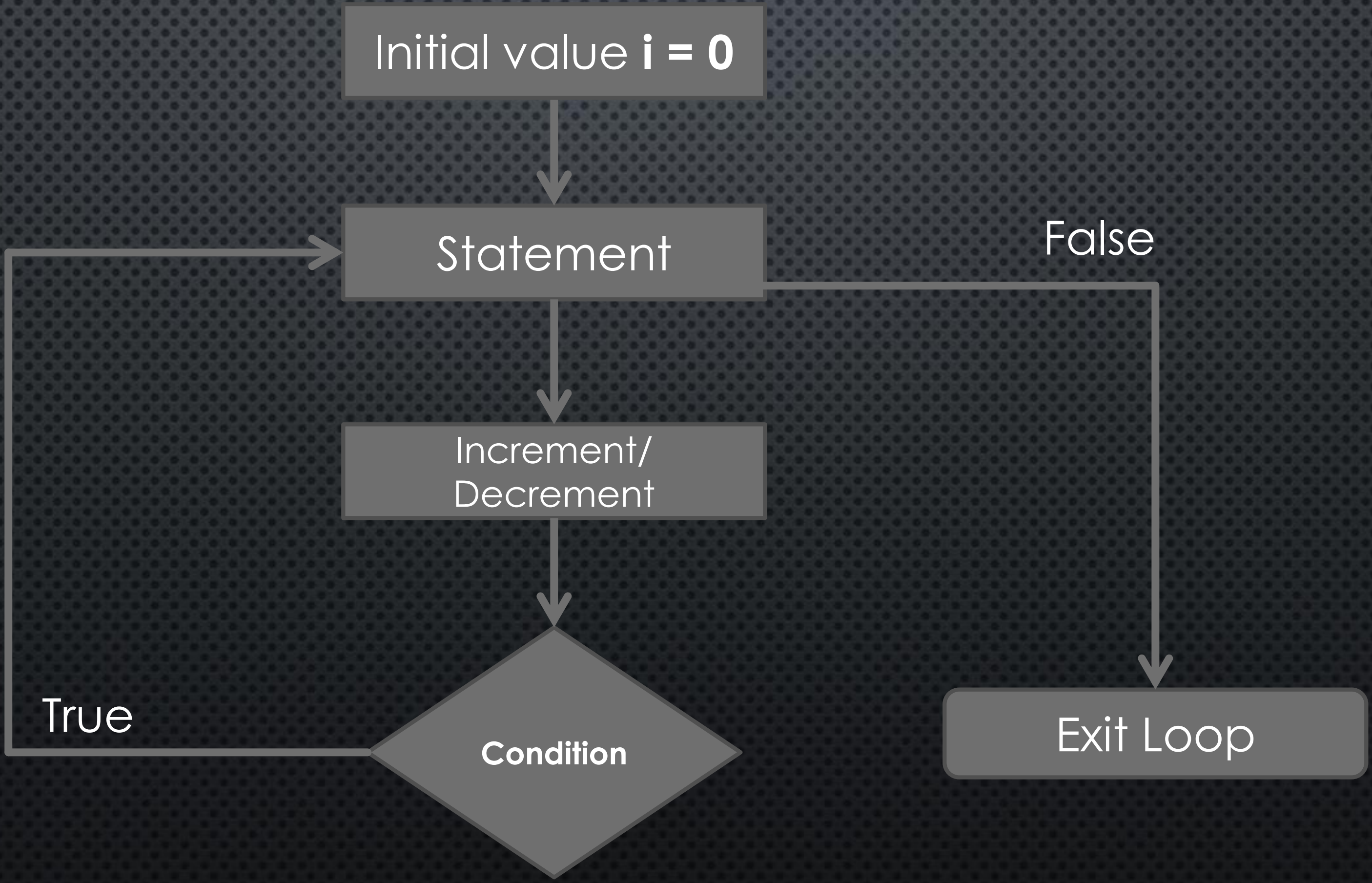
THE DO/WHILE LOOP

- The **Do/While Loop** is the variant of the while loop. The loop will be execute the code block once, before checking if the condition is **TRUE,** then it will repeat the loop as long as the condition is true.
- The **Do/While Loop** has the following syntax:



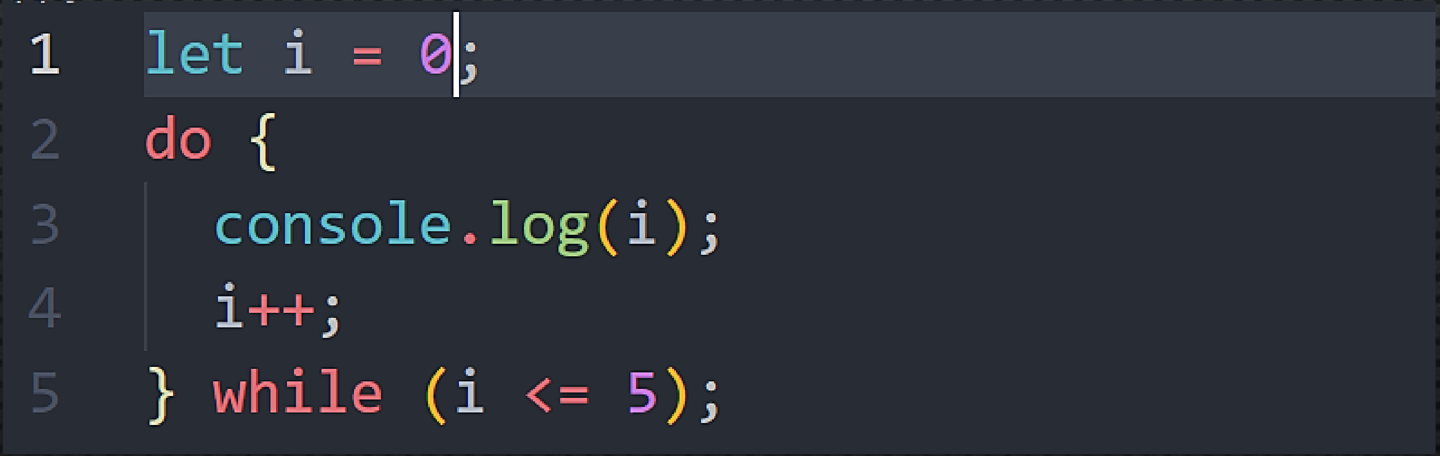
```
do {
  // code block to be executed
}
while (condition);
```

THE DO/WHILE LOOP



THE DO/WHILE LOOP

Code Example



```
let i = 0;
do {
  console.log(i);
  i++;
} while (i <= 5);
```

CONDITIONAL STATEMENTS:

- **Conditional Statements** are used to perform different **actions based on different conditions.**
- In **Conditional Statements**, you want to perform different **actions for different decisions.**

CONDITIONAL STATEMENTS:

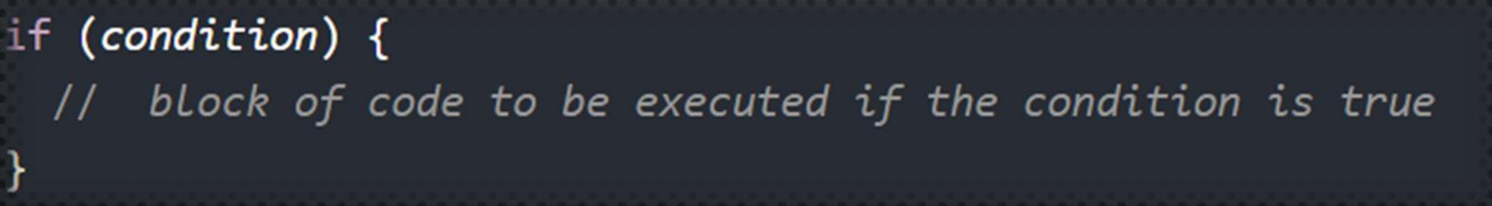
We have the following conditional statements:

- **"IF" Statement**
- **"ELSE" Statement**
- **"ELSE IF" Statement**
- **"SWITCH" Statement**

IF STATEMENT

"IF" STATEMENT: USE **IF** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **TRUE.**

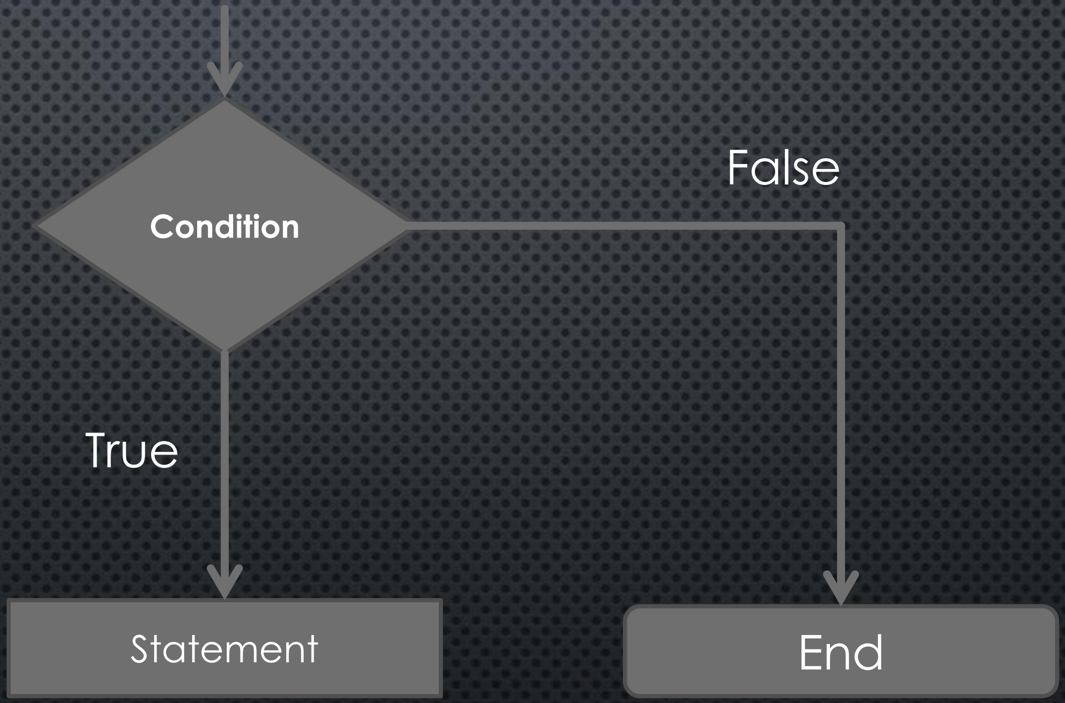
SYNTAX:



```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

Note that **if** is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

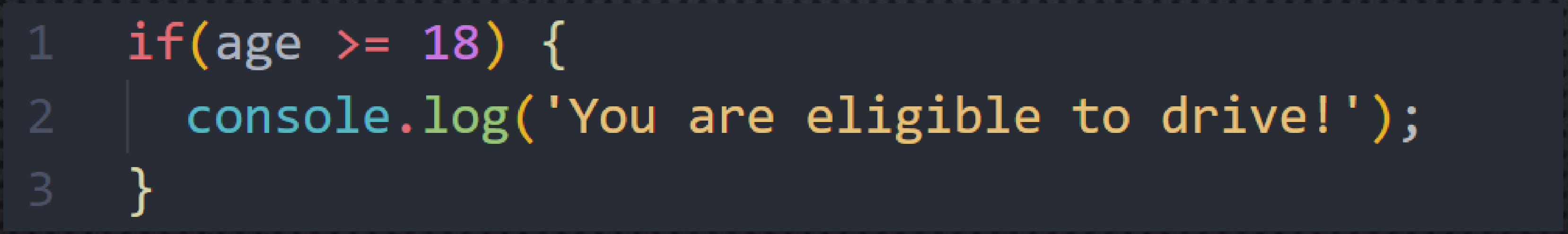
IF STATEMENT GRAPHICAL PRESENTATION



IF STATEMENT

"IF" STATEMENT: USE **IF** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **TRUE.**

CODE EXAMPLE:

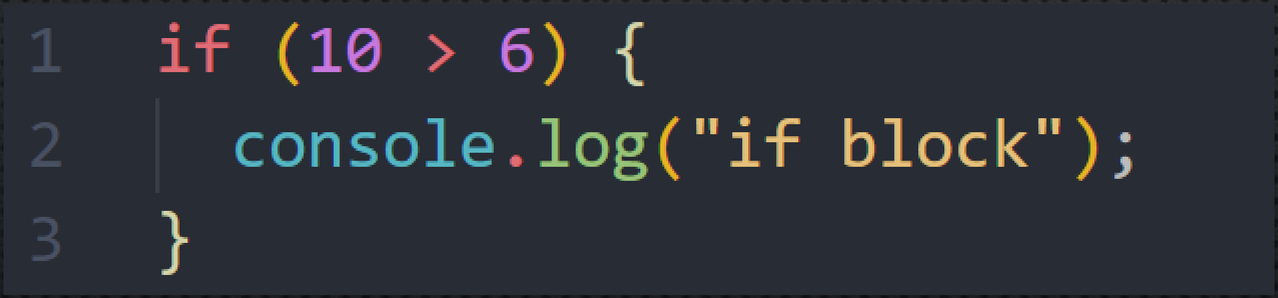


```
if(age >= 18) {
  console.log('You are eligible to drive!');
}
```

IF STATEMENT

"IF" STATEMENT: USE **IF** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **TRUE.**

CODE EXAMPLE:

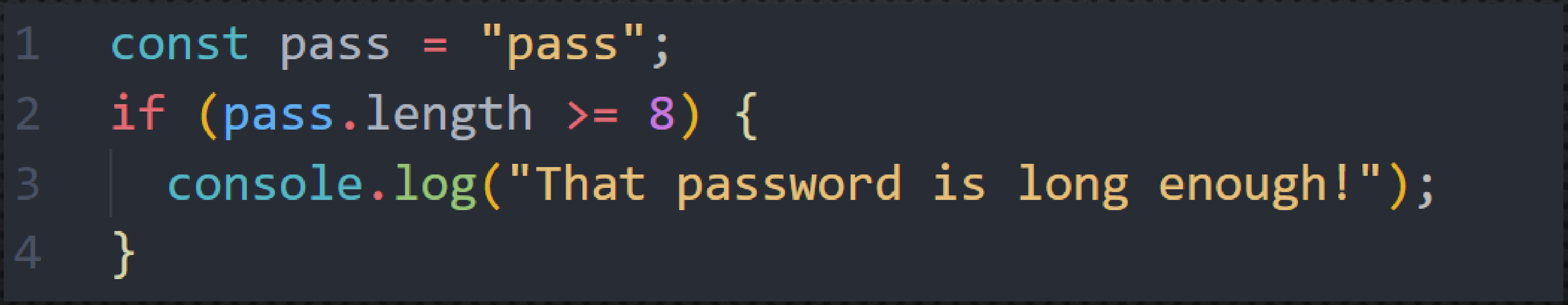


```
if (10 > 6) {
  console.log("if block");
}
```

IF STATEMENT

"IF" STATEMENT: USE **IF** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **TRUE.**

CODE EXAMPLE:



```
const pass = "pass";
if (pass.length >= 8) {
  console.log("That password is long enough!");
}
```

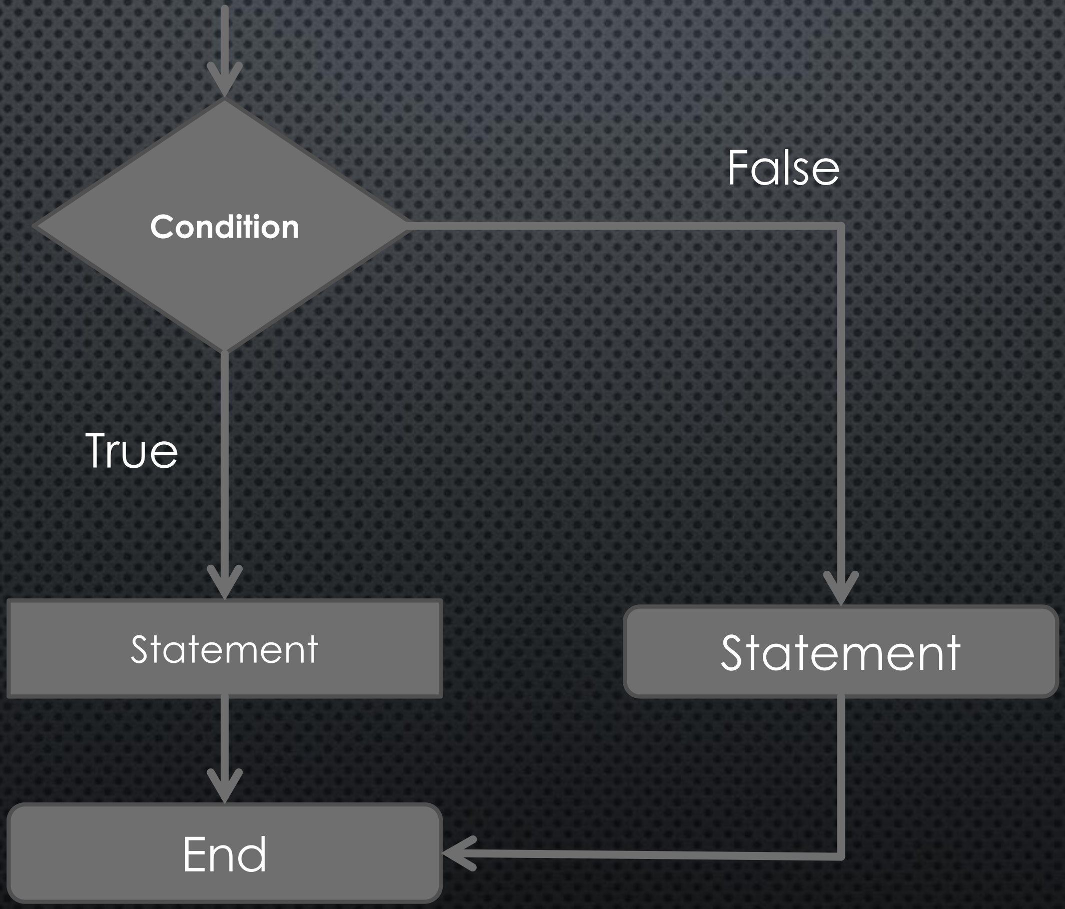
ELSE STATEMENT

"ELSE" STATEMENT: USE **ELSE** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **FALSE**.

SYNTAX:

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

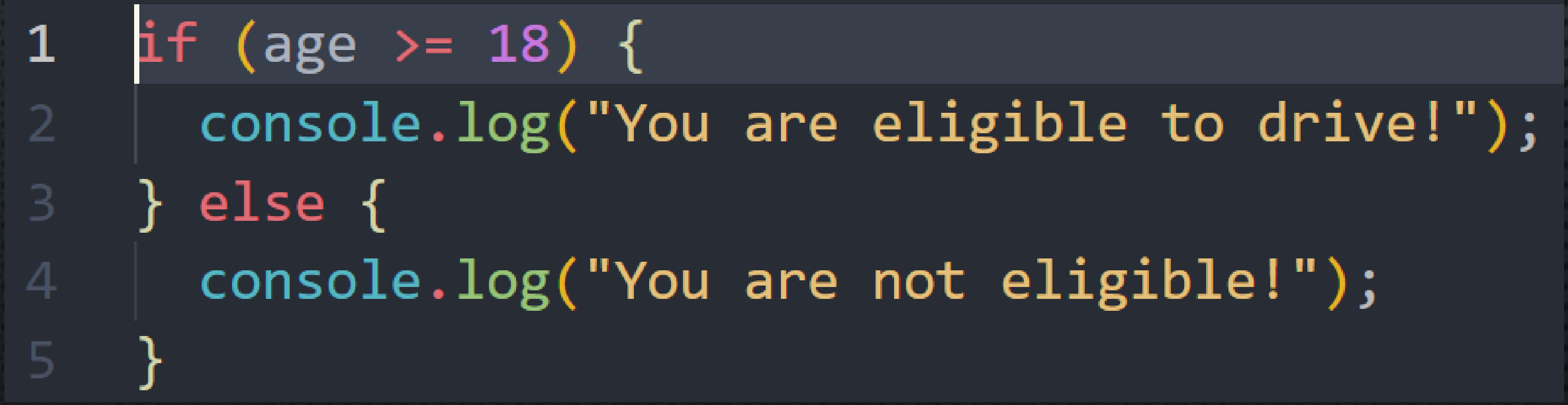
ELSE STATEMENT GRAPHICAL PRESENTATION



ELSE STATEMENT

“ELSE” STATEMENT: USE **ELSE** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **FALSE**.

CODE EXAMPLE:

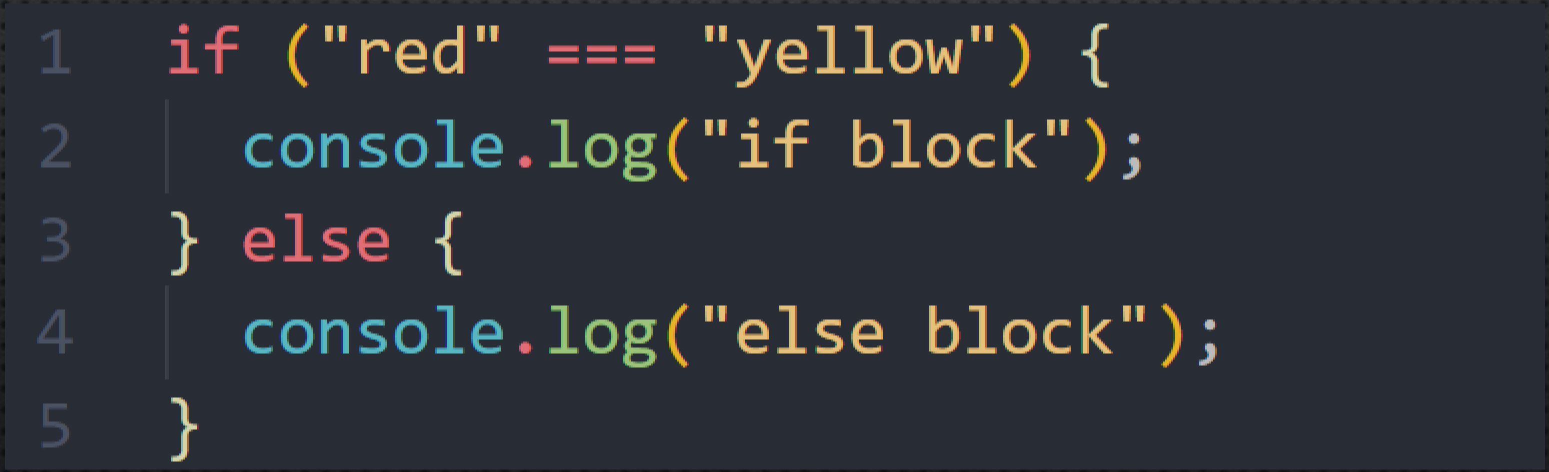


```
if (age >= 18) {
  console.log("You are eligible to drive!");
} else {
  console.log("You are not eligible!");
}
```

ELSE STATEMENT

"ELSE" STATEMENT: USE **ELSE** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **FALSE**.

CODE EXAMPLE:

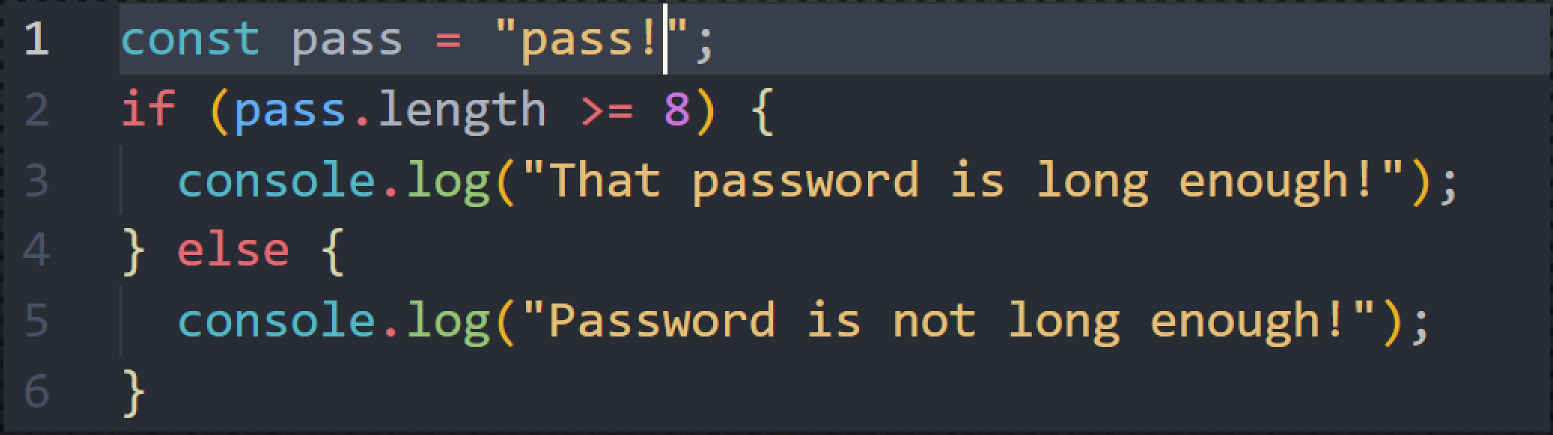


```
if ("red" === "yellow") {
  console.log("if block");
} else {
  console.log("else block");
}
```

ELSE STATEMENT

"ELSE" STATEMENT: USE **ELSE** TO SPECIFY A BLOCK OF CODE TO BE EXECUTED, IF A CONDITION IS **FALSE**.

CODE EXAMPLE:



```javascript
const pass = "pass!";
if (pass.length >= 8) {
  console.log("That password is long enough!");
} else {
  console.log("Password is not long enough!");
}
```

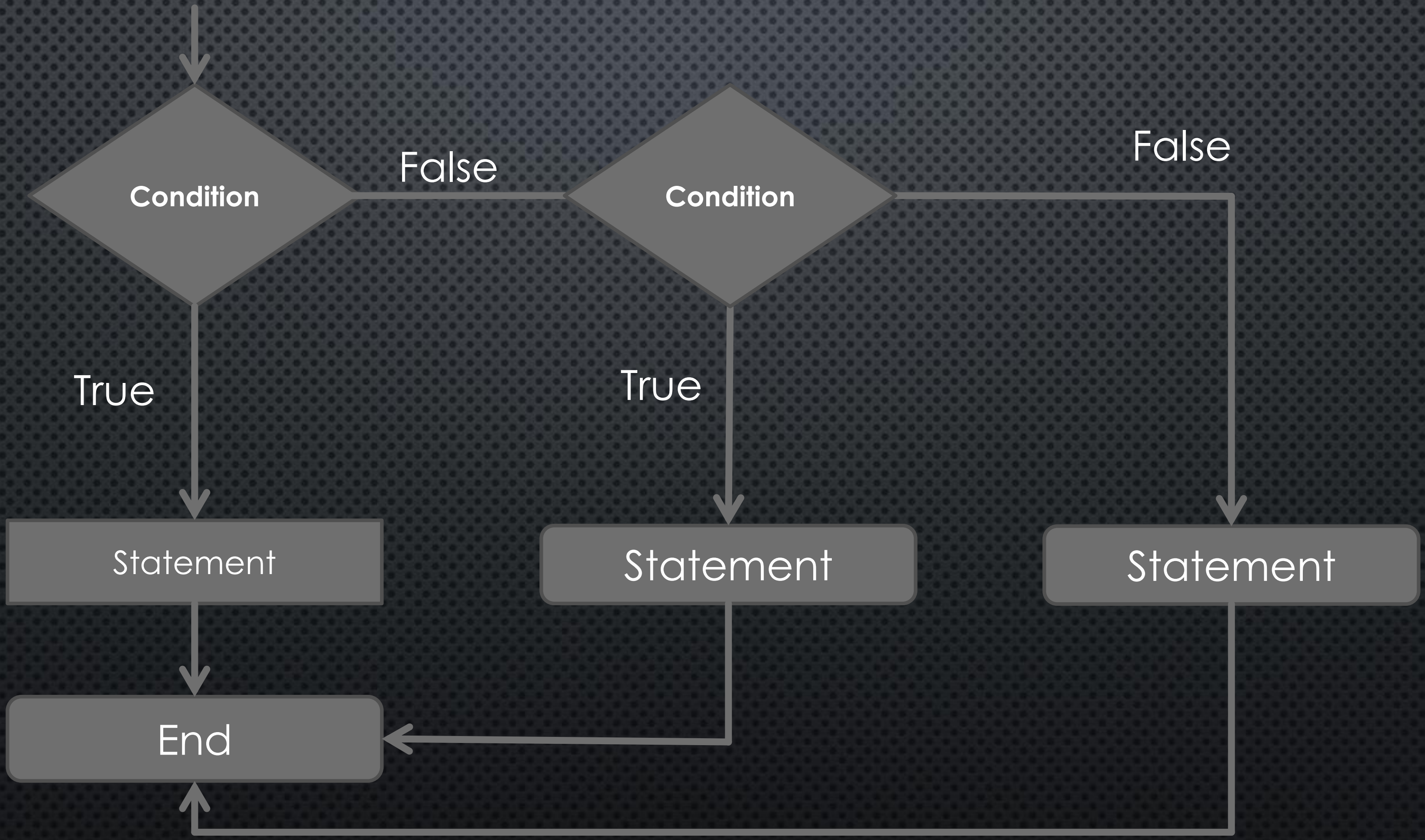
ELSE IF STATEMENT

"ELSE IF" STATEMENT: USE **ELSE IF** TO SPECIFY A NEW CONDITION TO TEST, **IF A FIRST CONDITION IS FALSE.**

SYNTAX:

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

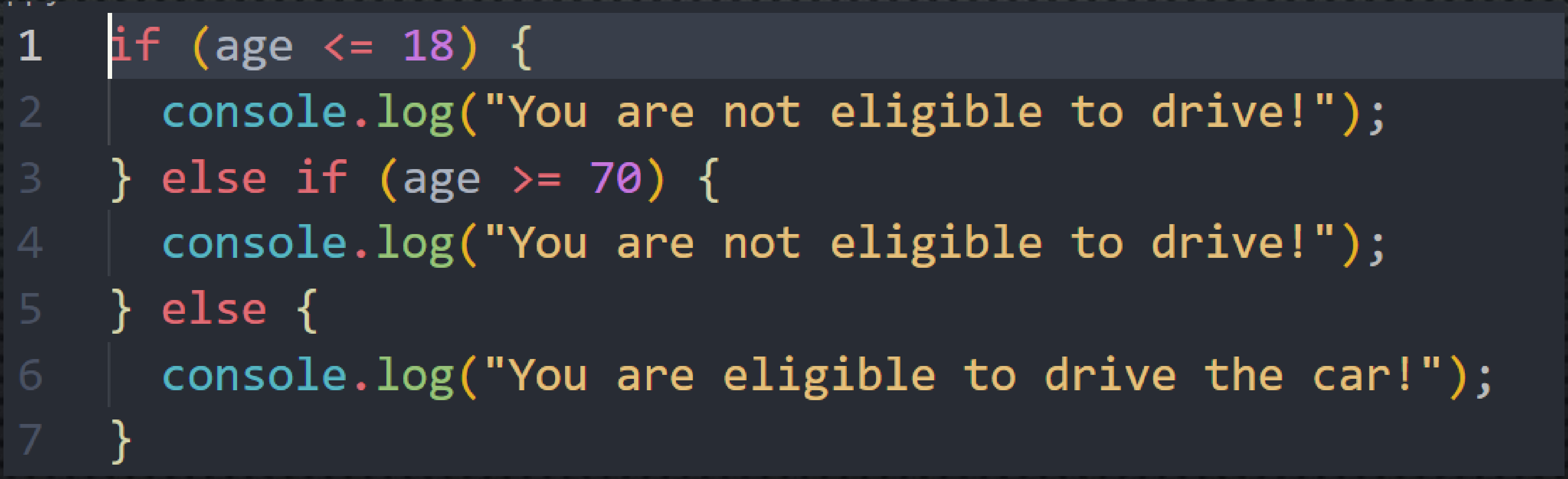
ELSE IF STATEMENT GRAPHICAL PRESENTATION



ELSE IF STATEMENT

"ELSE IF" STATEMENT: USE **ELSE IF** TO SPECIFY A NEW CONDITION TO TEST, **IF A FIRST CONDITION IS FALSE.**

CODE EXAMPLE:

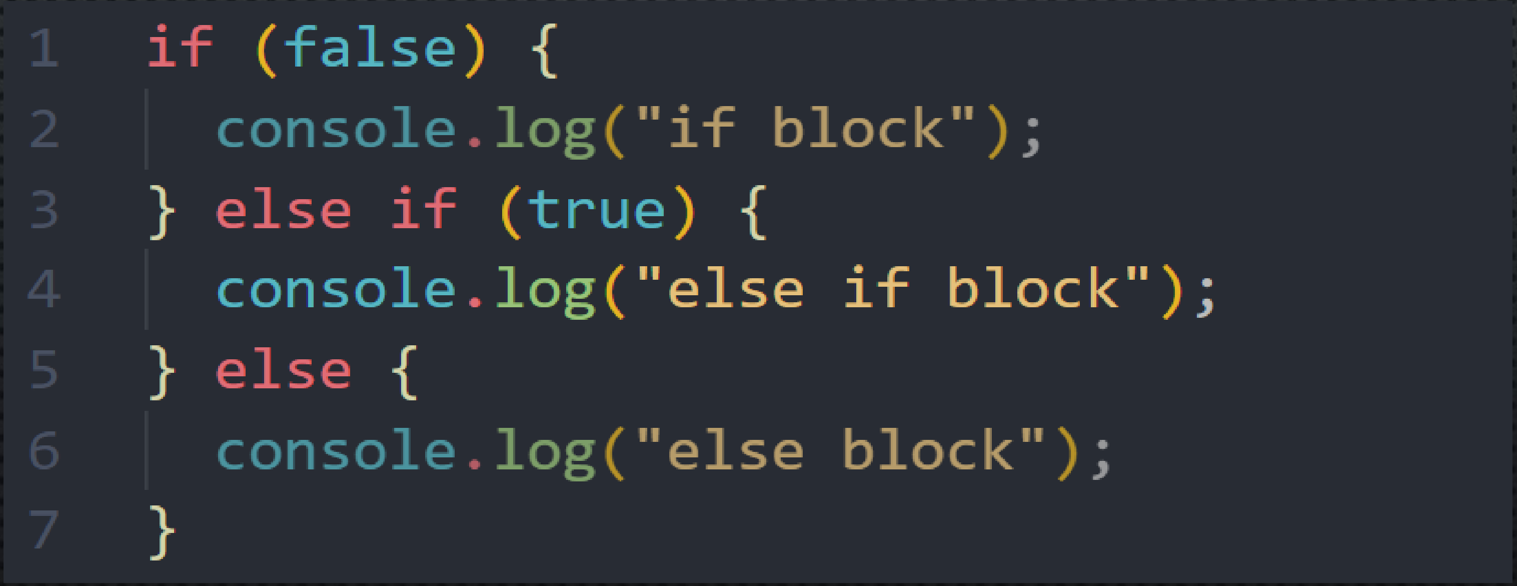


```
if (age <= 18) {
  console.log("You are not eligible to drive!");
} else if (age >= 70) {
  console.log("You are not eligible to drive!");
} else {
  console.log("You are eligible to drive the car!");
}
```

ELSE IF STATEMENT

"ELSE IF" STATEMENT: USE **ELSE IF** TO SPECIFY A NEW CONDITION TO TEST, **IF A FIRST CONDITION IS FALSE.**

CODE EXAMPLE:

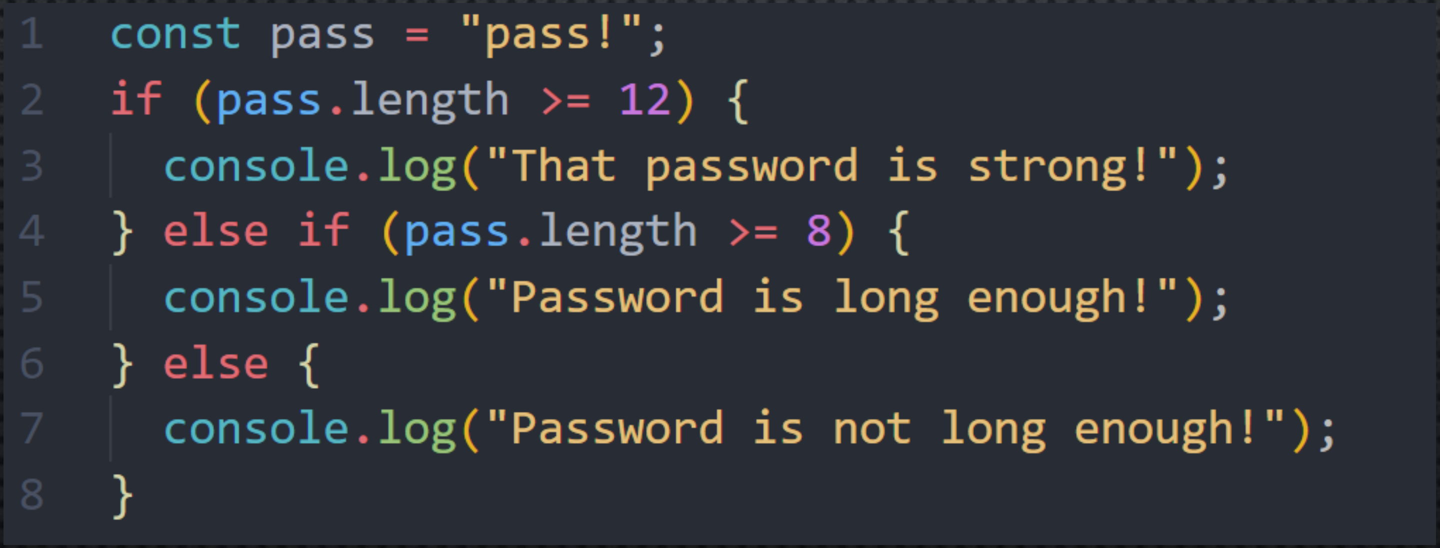


```
if (false) {
  console.log("if block");
} else if (true) {
  console.log("else if block");
} else {
  console.log("else block");
}
```

ELSE IF STATEMENT

"ELSE IF" STATEMENT: USE **ELSE IF** TO SPECIFY A NEW CONDITION TO TEST, **IF A FIRST CONDITION IS FALSE**.

CODE EXAMPLE:

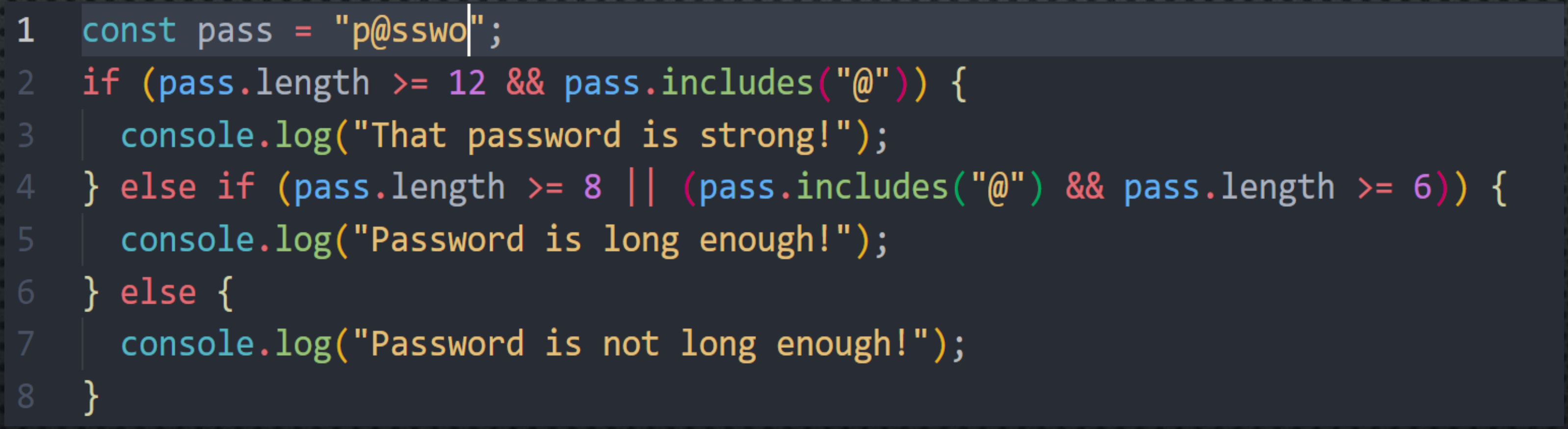


```
const pass = "pass!";
if (pass.length >= 12) {
  console.log("That password is strong!");
} else if (pass.length >= 8) {
  console.log("Password is long enough!");
} else {
  console.log("Password is not long enough!");
}
```

ELSE IF STATEMENT

LOGICAL OPERTORS OR || AND AND &&

CODE EXAMPLE:

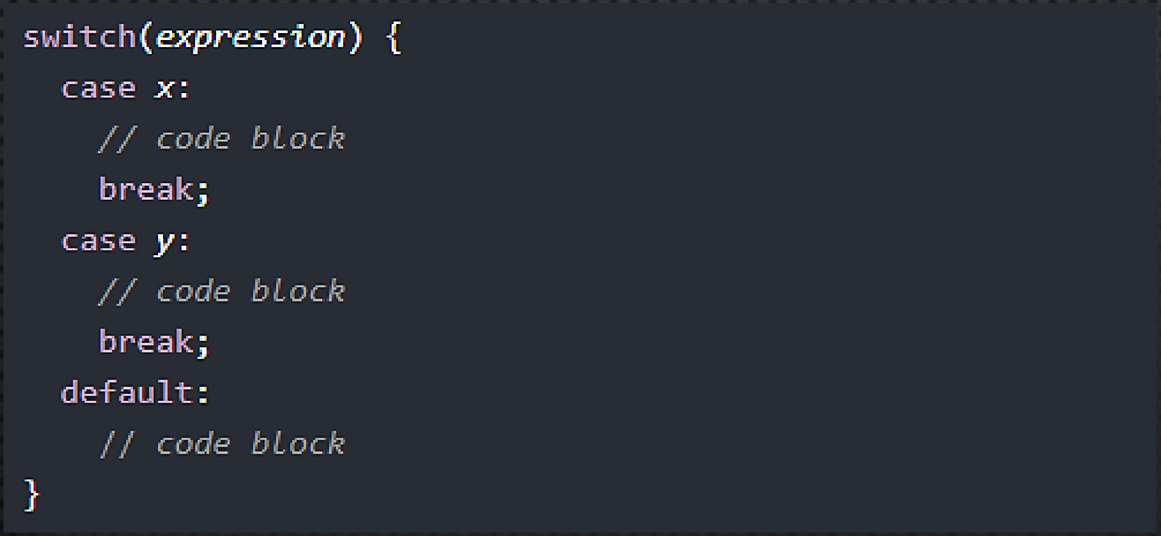


```javascript
const pass = "p@sswo";
if (pass.length >= 12 && pass.includes("@")) {
  console.log("That password is strong!");
} else if (pass.length >= 8 || (pass.includes("@") && pass.length >= 6)) {
  console.log("Password is long enough!");
} else {
  console.log("Password is not long enough!");
}
```

SWITCH STATEMENT

"SWITCH" STATEMENT: USE **SWITCH** STATEMENT TO **SELECT ONE OF MANY CODE BLOCKS** TO BE EXECUTED.

SYNTAX:



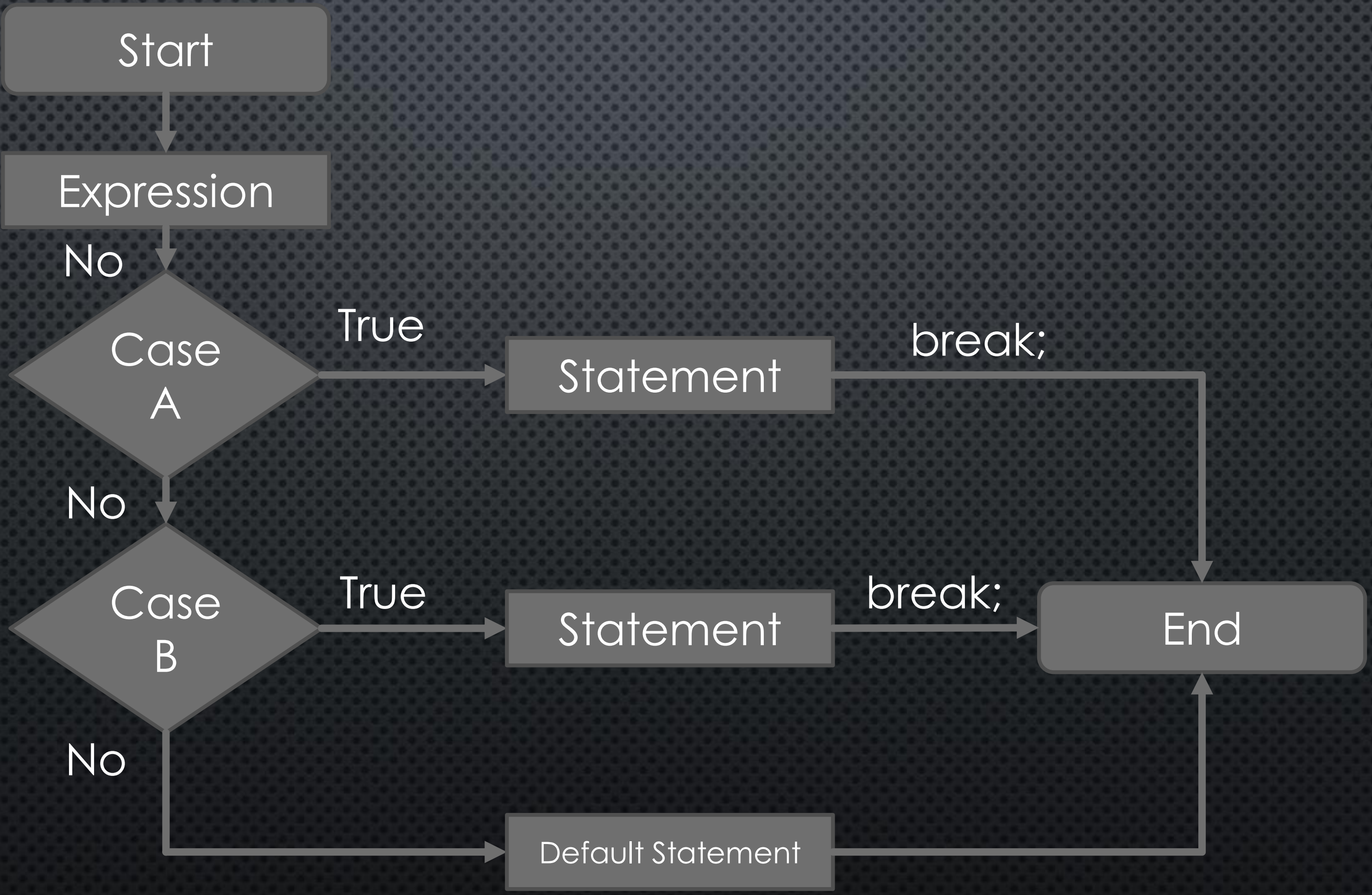
```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

SWITCH STATEMENT

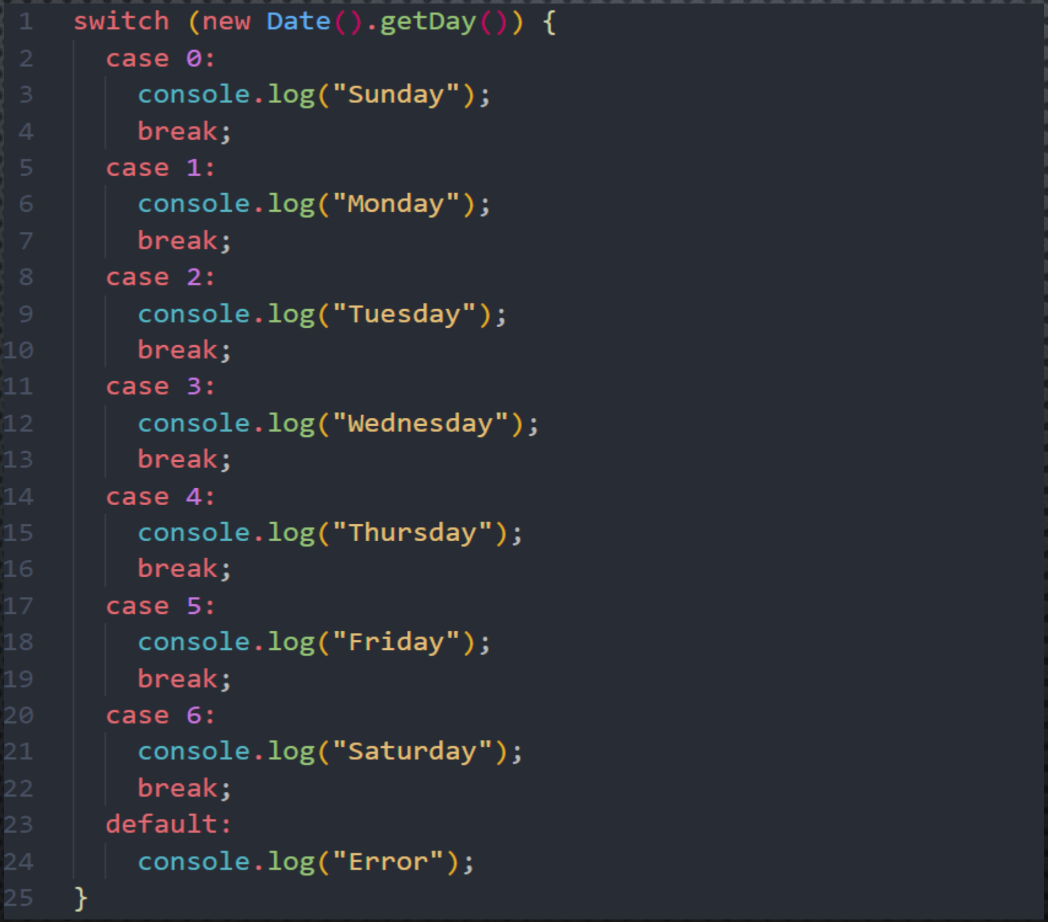
SWITCH STATEMENT, HOW IT WORKS:

- THE SWITCH EXPRESSION IS EVALUATED ONCE.
- THE VALUE OF THE EXPRESSION IS COMPARED WITH THE VALUES OF EACH CASE.
- IF THERE IS A MATCH, THE ASSOCIATED BLOCK OF CODE IS EXECUTED.
- IF THERE IS NO MATCH, THE DEFAULT CODE BLOCK IS EXECUTED.

SWITCH STATEMENT GRAPHICAL PRESENTATION



SWITCH STATEMENT CODE EXAMPLE



```javascript
switch (new Date().getDay()) {
  case 0:
    console.log("Sunday");
    break;
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  case 3:
    console.log("Wednesday");
    break;
  case 4:
    console.log("Thursday");
    break;
  case 5:
    console.log("Friday");
    break;
  case 6:
    console.log("Saturday");
    break;
  default:
    console.log("Error");
}
```

QUESTIONS