JavaScript Fundamentals

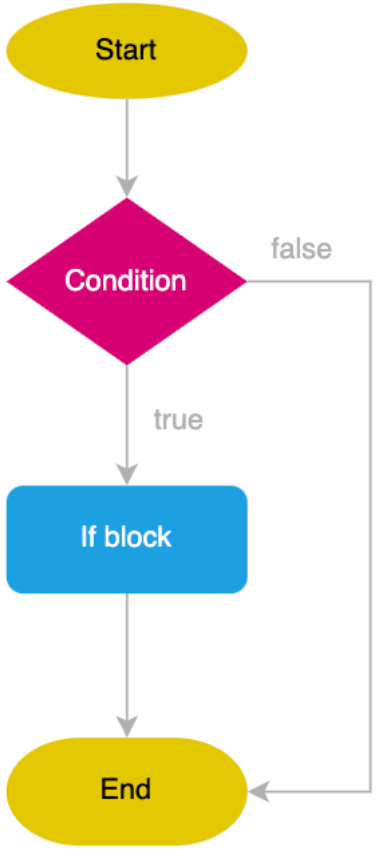
Instructor: Bui Binh Giang
buibinhgiang@vanlanguni.vn

CONTROL FLOW STATEMENTS

JavaScript if statement

The **if** statement executes block if a condition is true

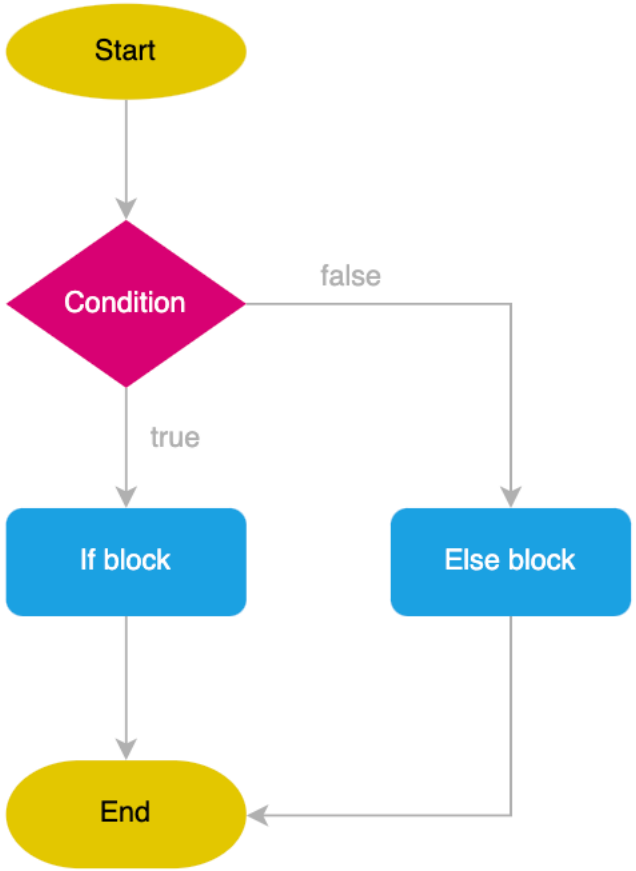
```
if (condition) {
  // statements to execute
}
```



JavaScript if...else statement

Use the JavaScript **if...else** statement to execute a block if a condition is true and another block otherwise

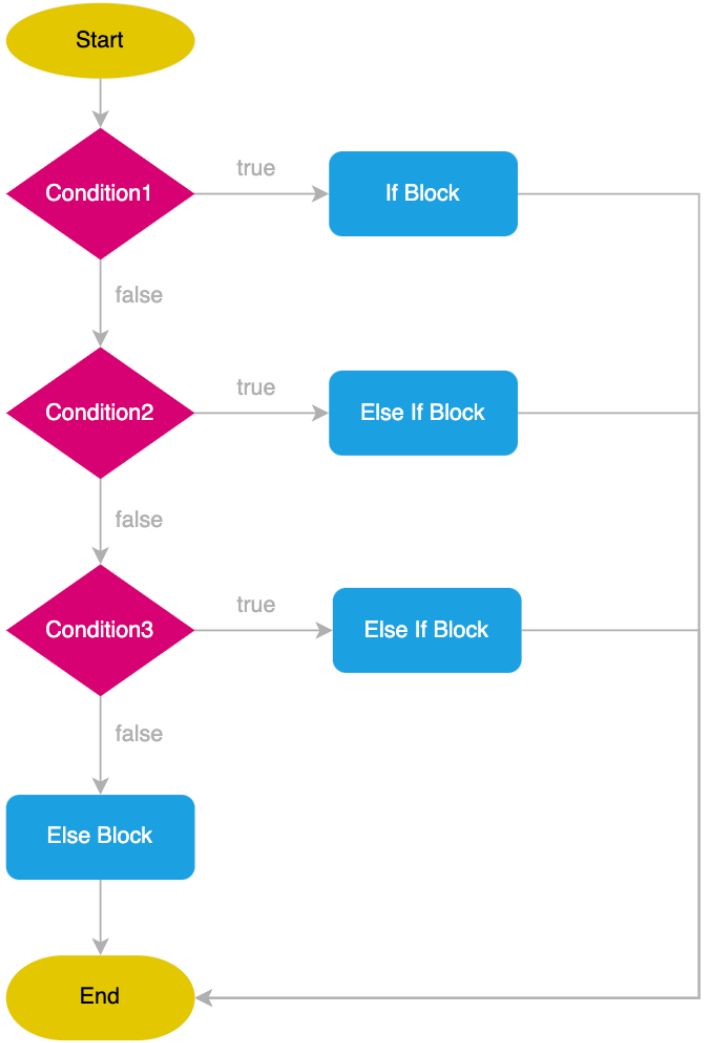
```
if( condition ) {
  // ...
} else {
  // ...
}
```



JavaScript if...else if... statement

Use the JavaScript **if...else...if** statement to check multiple conditions and execute the corresponding block if a condition is true

```
if (condition1) {
  // ...
} else if (condition2) {
  // ...
} else if (condition3) {
  //...
} else {
  //...
}
```



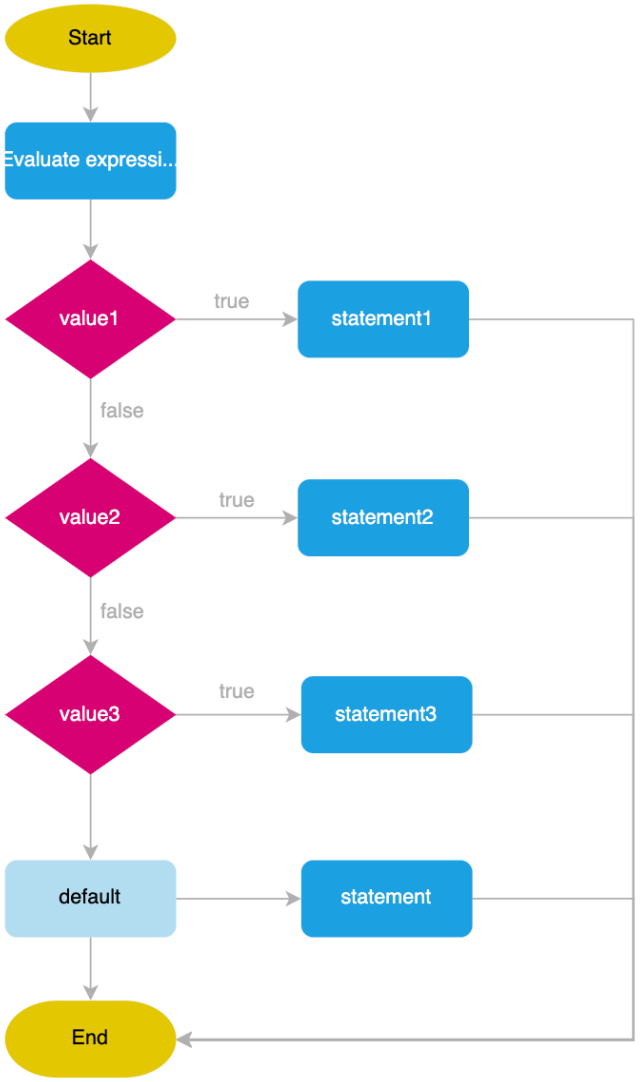
JavaScript Ternary Operator

```
let variableName = condition ? expressionIfTrue : expressionIfFalse;
```

JavaScript switch case statement

- The **switch** statement evaluates an expression, compare its result with **case** values, and execute the statement associated with the matching case.
- Use the **switch** statement to rather than a complex **if...else...if** statement to make the code more raedable.
- The **switch** statement uses the strict comparison (===) to compare the expression with the **case** values.

```
switch (expression) {
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    case value3:
        statement3;
        break;
    default:
        statement;
}
```



Exercise - 201

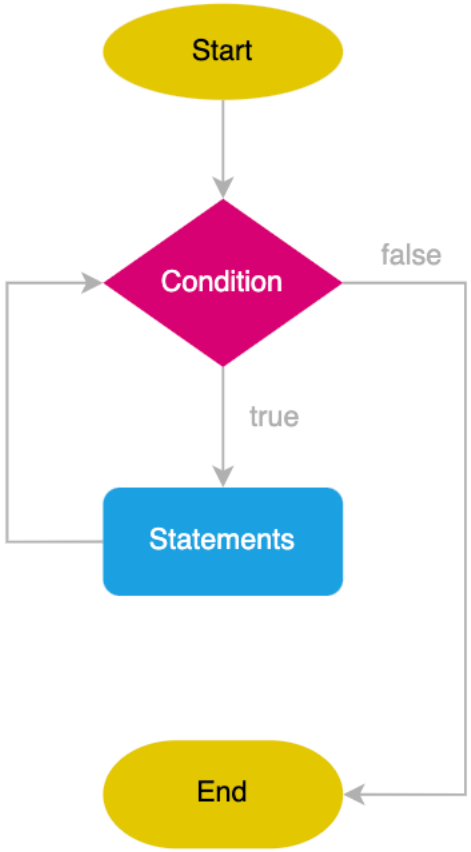
Requirement:

- Create a new object named “date” with 3 props: “dayName”, “day”, “month”.
- Give them random values:
  - “dayName”: “Sunday” → “Saturday”
  - “month”: “January” → “December”
  - “day”: 1 → 28/31
- Output the to the console a string with format: “dayName day month”. Ex: Monday 26 December
- **Hint**: *let randomNum = Math.floor(Math.random() * (MAX - MIN + 1)) + MIN;*

JavaScript while loop statement

Use a **while** loop statement to create a loop that executes a block as long as a condition evaluates to **true**.

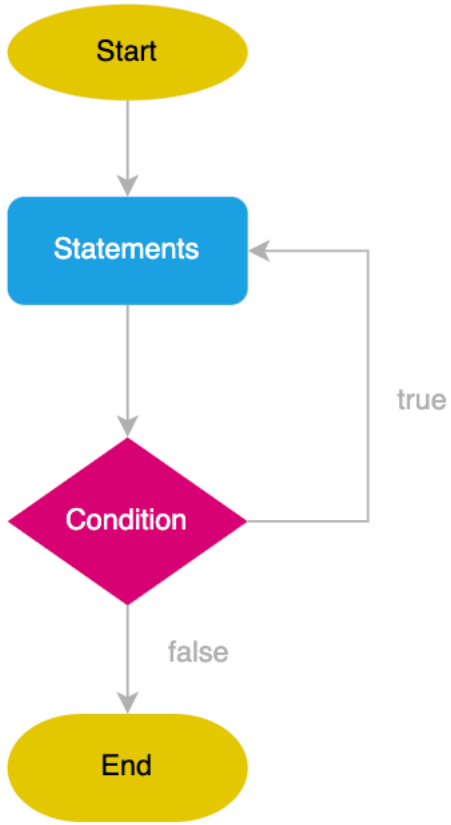
```
while (expression) {
    // statement
}
```



JavaScript do…while statement

Use the **do…while** statement to create a loop that executes a code block until a condition is false.

```
do {
  statement;
} while(expression)
```

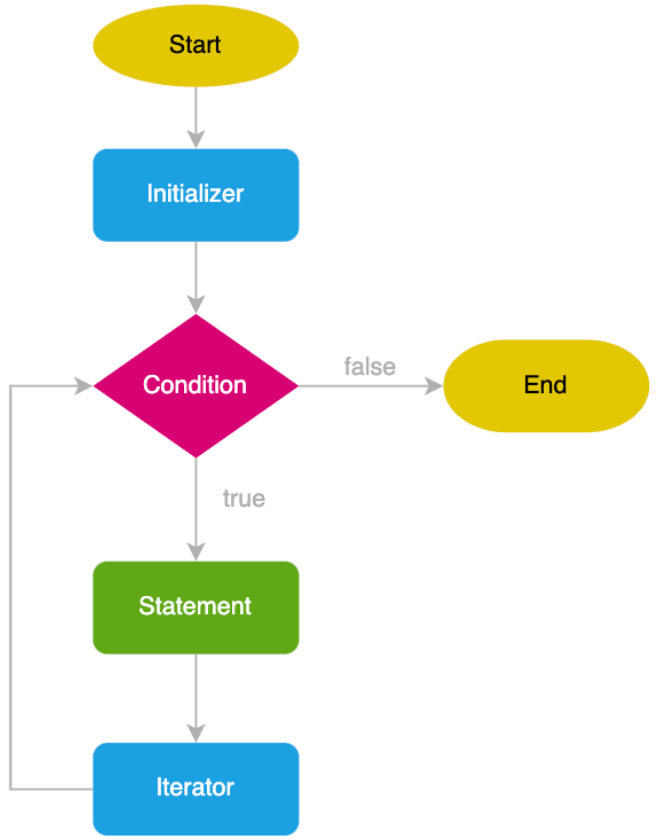


JavaScript for loop statement

The **for** loop statement creates a loop with three optional expressions:

- **Initializer**: The **for** statement executes the initializer only once the loop starts. Typically, you declare and initialize a local loop variable in the initializer
- **Condition**: The **for** statement evaluates the condition before each iteration. If the condition is true (or is not present), it executes the next iteration. Otherwise, it'll end the loop
- **Iterator**: The **for** statement executes the iterator after each iteration.

```
for (initializer; condition; iterator) {
    // statements
}
```



JavaScript break statement

- Use the **break** statement to terminate a loop including **for**, **while**, and **do...while** prematurely
- When used in a nested loop, the **break** statement terminates the enclosing loop. To terminate the nested loop, you use a **label** statement

```
for (let i = 0; i < 5; i++) {
  console.log(i);
  if (i == 2) {
    break;
  }
}
```

JavaScript continue statement

Use the JavaScript **continue** statement to skip the current iteration of a loop and continue the next one

```
for (let i = 0; i < 10; i++) {
  if (i % 2 === 0) {
    continue;
  }
  console.log(i);
}
```

Exercise - 202

Requirement:

- Generates a random integer between 1 and 10.
- Prompt user to guess a number until his number matches the random number .
- Output the number of guesses.

```
let input = prompt( message: `Please enter a number between ${MIN} and ${MAX}`);
```

```
alert(`You're correct after ${guesses} guess(es).`);
```

FUNCTIONS

JavaScript functions

- To avoid repeating the same code all over places, you can use a function to wrap that code and reuse it.
- Use the **function** keyword to declare a function.
- Use the **functionName**() to call a function.
- All functions implicitly return **undefined** if they don't explicitly return a value.
- Use the return statement to return a value from a function explicitly.
- The **arguments** variable is an array-like object inside a function, representing function arguments.
- The function **hoisting** allows you to call a function before declaring it.

JavaScript functions

Declare a function:

```
function functionName(parameters) {
    // function body
    // ...
}
```

Calling a function:

```
functionName(arguments);
```

The arguments object:

```
function add() {
    let sum = 0;
    for (let i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

JavaScript Anonymous Functions

Anonymous functions:

```
(function () {
    //...
});
```

Using anonymous functions as arguments:

```
setTimeout(function() {
    console.log('Execute later after 1 second')
}, 1000);
```

Immediately invoked function execution:

```
(function () {
    console.log('Immediately invoked function execution');
})();
```

JavaScript Recursive Function

- A recursive function is a function that calls itself until it doesn't.
- A recursive function always has a condition that stops the function from calling itself.

```
function recurse() {
    if(condition) {
        // stop calling itself
        //...
    } else {
        recurse();
    }
}
```

Exercise - 203

Requirement:

- Define an Array of 100 elements type Number with random value (range [1..100]).
- Output all prime values of Array.
- Special requirement: Write a recursive function to check prime number.