Control Statements

Ba Nguyễn

Control Structure

Cấu trúc điều khiển là *đặc trưng và là yếu tố quan trọng* trong lập trình. Các cấu trúc điều khiển *xác định logic cho chương trình*, ví dụ:

- Chạy các câu lệnh *theo một thứ tự* nào đó
- *Bỏ qua* một vài câu lệnh, chạy các câu lệnh khác
- Thực thi một *nhóm các câu lệnh* dựa *theo một điều kiện* nào đó
- *Lặp đi lặp lại* một hành động (nhóm các câu lệnh)

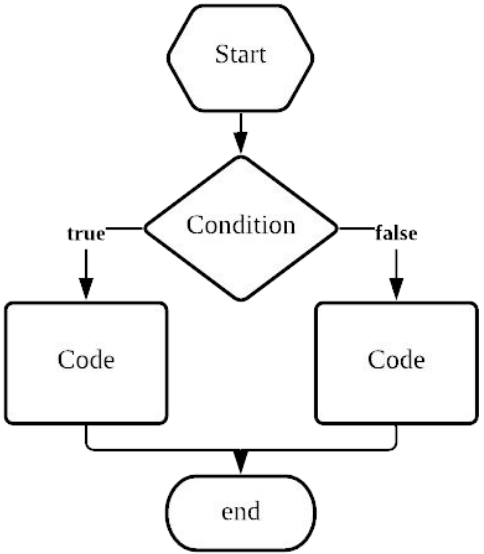
Các cấu trúc điều khiển trong JavaScript:

- **if else**
- **for loop**
- **while loop**
- ...

If Else

Thực thi các câu lệnh theo điều kiện chỉ định, điều kiện có thể là một giá trị, biểu thức, hàm, ... và được tự động chuyển đổi về kiểu **boolean**. Mệnh đề **if** chỉ định điều kiện và mã bên trong chỉ được thực thi nếu điều kiện đúng, ngược lại, mệnh đề **else** sẽ được thực thi

```
if (condition) {
    // Nếu condition là true
    // Thực thi khối mã này
} else {
    // Nếu condition là false
    // Thực thi khối mã này
}
```



If Else

```
let user = {
    name: "...",
    isAdmin: true,
}

if (user.isAdmin) {
    console.log("Hello, " + user.name + "!");
    console.log("Rendering data");
} else {
    console.log("Forbiden");
}
```

If Else

Mệnh đề **if** có thể sử dụng độc lập mà không có mệnh đề **else**, hoặc kết hợp kiểm tra nhiều điều kiện với mệnh đề **else if**

```
if (condition) {
    // Nếu condition là true thì thực thi đoạn mã
    // Nếu condition là false thì bỏ qua
}

if (condition) {
    // Nếu condition là true thì thực thi đoạn mã này
    // Nếu condition là false thì kiểm tra điều kiện phía sau
} else if (otherCondition) {
    // ...
} else { }
```

If Else

Khi sử dụng **if else** kiểm tra nhiều điều kiện, lưu ý các điều kiện trùng nhau, điều kiện ít khả năng xảy ra nhất nên kiểm tra trước

```
if (points > 10000) console.log("Diamon");
else if (points > 5000) console.log("Gold");
else if (points > 3000) console.log("Silver");
else if (points > 1000) console.log("Member");
else console.log("Cheap");
```

💡Khi chỉ có một câu lệnh cho mỗi khối lệnh **if else** có thể bỏ qua phần **{ }**

For loops

Thực hiện lặp lại một hành động (khối mã) cho tới khi điều kiện trở thành **false**

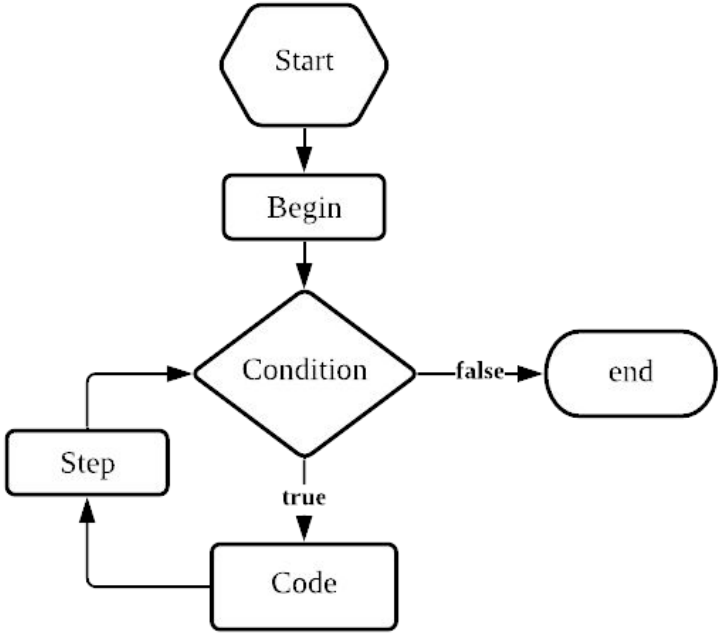
```
for (begin; condition; step) {
    // Thực thi khối mã lặp đi lặp lại
    // Cho tới khi condition là false
}

// Begin: Khởi tạo (các) giá trị ban đầu cho vòng lặp
// Condition: (Các) Điều kiện để tiếp tục chạy vòng lặp
// Step: Cập nhật (các) giá trị sau mỗi lần lặp
```

💡 Cần chú ý điều kiện dừng, tránh vòng lặp vô hạn

For loops

Thực hiện lặp lại một hành động (khối mã) cho tới khi điều kiện trở thành **false**



For loops

```
console.log("Before loop");       // Before loop

for (let i = 0; i < 10; i++) {    // 0
    console.log(i);               // 1
}                                 // ...
                                  // 9

console.log("After loop");        // After loop
```

For loops

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

// 1. Khởi tạo giá trị i = 0 (1 lần duy nhất)
// 2. Kiểm tra điều kiện i < 10 (true)
// 3. In ra console i (0)
// 4. Tăng i lên 1 (i = 2)
// 5. Quay lại bước 2, kiểm tra điều kiện i < 10 (true)
// ...
// N. Quay lại bước 2, kiểm tra điều kiện i < 10 (false)
// Dừng vòng lặp
```

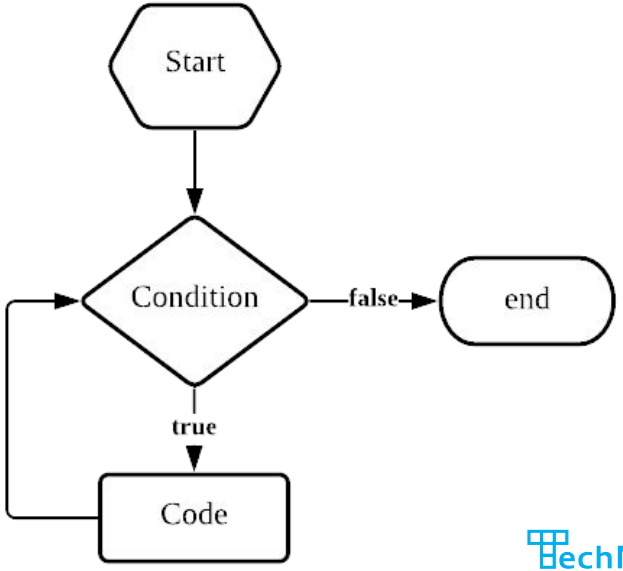
While loops

Tương tự **for**, thực hiện lặp lại một hành động (khối mã) cho tới khi điều kiện trở thành **false**, tuy nhiên khác với **for**, **while** không có bước khai báo và cập nhật giá trị với mỗi vòng lặp

```
// Các giá trị khởi tạo (nếu có)

while (condition) {
    // Thực thi khối mã lặp đi lặp lại
    // Cho tới khi condition là false

    // Tự cập nhật các giá trị (nếu có)
    // bên trong vòng lặp
}
```



While loops

Tương tự **for**, thực hiện lặp lại một hành động (khối mã) cho tới khi điều kiện trở thành **false**, tuy nhiên khác với **for**, **while** không có bước khai báo và cập nhật giá trị với mỗi vòng lặp

```
let validCode = 123;
let answer = prompt("Enter passcode");

while (answer != validCode) {
    alert("Invalid passcode! Try again!");

    answer = prompt("Re-Enter passcode")
}
```

Break, Continue

Câu lệnh **break** và **continue** chỉ sử dụng trong vòng lặp

continue dừng lần lặp hiện tại, chuyển lớp lần lặp tiếp theo, các câu lệnh bên dưới cũng bị bỏ qua

```
for (let i = 0; i < 10; i++) {
    if (i % 2 == 0 || i % 3 == 0) continue;

    // Nếu điều kiện trong if là đúng
    // Các câu lệnh này sẽ bị bỏ qua
    console.log(i * i);
}

// Kết quả: 1 25 49
```

Break, Continue

Câu lệnh **break** và **continue** chỉ sử dụng trong vòng lặp

break được sử dụng để dừng vòng lặp, thay thế cho điều kiện **condition**, tương tự **continue**, các câu lệnh phía sau sẽ bị bỏ qua

```
let validCode = 123;

while (true) {
    let answer = prompt("Enter passcode");

    if (answer == validCode) break;
    // Nếu điều kiện trong if là đúng sẽ dừng vòng lặp
}
```

Return

Trong **function**, có thể sử dụng **return** thay thế cho **break** trong vòng lặp, **return** ngay lập tức dừng vòng lặp tương tự **break** và trả về kết quả

```
// Hàm kiểm tra n có phải số nguyên tố hay không
function isPrime(n) {
    for (let i = 2; i <= n / 2; i++) {
        // Nếu n chia hết cho 1 số trong khoảng 2 -> n / 2
        // thì n không phải số nguyên tố
        if (n % 2 === 0) return false;
    }

    return true;
}
```

Try Catch

Cú pháp **try catch** được sử dụng để xử lý trường hợp một lỗi có thể phát sinh trong khi chương trình thực thi (VD: biến không có giá trị, giá trị không phù hợp, ...).

Nếu một lỗi không được xử lý, toàn bộ chương trình có thể bị dừng ngay lập tức.

```
try {
    // Code có thể phát sinh lỗi
} catch (error) {
    // Code xử lý khi có lỗi phát sinh
}
```

Try Catch

```
try {
    console.lgo("'lgo' không phải function");
} catch (error) {
    console.log(error.name);
    console.log(error.message);
}

console.log("After handling the error");
console.log("App is still running");
```

Note

💡Tip:

- **for** thường dùng với các vòng lặp mà **xác định được** chính xác số lần lặp
- **while** thường dùng với các vòng lặp mà **không xác định** được số lần lặp chính xác

Ngoài các cấu trúc điều khiển trên, còn các cấu trúc điều khiển khác như **do while**, **switch case**

Ngoài ra, **for** có thể bỏ qua một (hoặc tất cả) biểu thức bên trong **( )**, cách hoạt động cũng tương tự **while**

```
for (; ;) {
    // sử dụng break để ngắt vòng lặp
}
```

Exercises

1. Viết chương trình lặp giá trị **i** từ **1** đến **10**, với mỗi giá trị **i**, kiểm tra xem nó là chẵn hay lẻ và in ra console
2. Viết chương trình lặp giá trị **i** từ **1** đến **10**, với mỗi giá trị **i**, kiểm tra xem:
   - Nếu **i** chia hết cho **3** thì in ra console **“Fizz”**
   - Nếu **i** chia hết cho **5** thì in ra console **“Buzz”**
   - Nếu **i** chia hết cho cả **3** và **5** thì in ra **“FizzBuzz”**
3. Viết chương trình kiểm tra xem một số nguyên dương **n** có phải số nguyên tố hay không
4. Viết chương trình in ra chữ số đầu và cuối của một số **n** (VD: 12345 -> 15)