1. Lập trình là quá trình tạo ra các chỉ dẫn để ra lệnh cho máy tính hoàn thành 1 công việc nào đó
2. Pseudo code ( mã giả ) là cách mô tả thuật toán bằng cách tự nhiên
3. Flowchart mô tả giải thuật bằng các sơ đồ khối

|  |  |  |
| --- | --- | --- |
|  | Pseudo code | Flowchart |
| Ưu điểm: | Đơn giản, không cần kiến thức về cách biểu diễn (lưu đồ, ngôn ngữ lập trình | -Ngắn gọn, dễ hiểu.  -Dễ dàng diễn đạt được thuật toán |
| Nhược điểm: | -Dài dòng, không cấu trúc  -Đôi lúc khó hiểu, không diễn đạt được thuật toán | -Cần kiến thức về biểu diễn lưu đồ |

1. Ưu nhược điểm
2. Mục đích của form: cho phép người dùng nhập dữ liệu và gửi về sever
3. Khác nhau giữa post và get:

|  |  |
| --- | --- |
| Get | Post |
| * Dữ liệu gửi đi sẽ hiển thị trên thanh địa chỉ * Dung để gửi những dữ liệu nhỏ * Không bảo mật | * Dữ liệu gửi đi không hiển thị trên thanh địa chỉ * Không hạn chế, dùng để gửi dữ liệu lớn đến sever * Bảo mật cao |

1. Bảng được sử dụng để thể hiện dữ liệu dưới dạng hàng và cột để hiển thị một danh sách các đối tượng với nhiều thuộc tính khác nhau.

Các thẻ tạo bảng: <table>, <tr>, <th>, <td>

1. JavaScript là một ngôn ngữ script được dùng để tạo các script ở máy client và máy server. Các script ở máy client có thể thực thi được trong trình duyệt trong khi các script ở máy server lại thực hiện trên server.
2. Các cách nhúng JS:

* Sử dụng thẻ <script> bên trong mã HTML
* Sử dụng file javascript riêng biệt
* Viết mã Javascript ngay trong thẻ các thẻ HTML

1. Quy ước:

* Tên phải bắt đầu bằng ký tự chữ cái (từ a-z hoặc A-Z), ký tự gạch dưới (\_), hoặc dấu đô-la ($). Không được bắt đầu tên bằng một ký tự nào khác
* Tên chỉ có thể chứa chữ cái (viết hoa và viết thường), chữ số, dấu gạch dưới và dấu đô-la. Không được phép sử dụng các ký tự đặc biệt (chẳng hạn như dấu # hoặc ! hoặc ?...) trong tên
* Javascript phân biệt chữ hoa và chữ thường. Có nghĩa là một biến có tên là myName sẽ khác với một biến có tên là myname.

1. Trong Javascript, chúng ta có 2 cách để viết chú thích.

* Viết chú thích trên một dòng. Sử dụng 2 dấu gạch chéo: //
* Viết chú thích trên nhiều dòng. Sử dụng dấu /\* và \*/:

1. Một số phương pháp debug:

* Đọc mã nguồn (hand-trace)
* Chèn các lệnh in ra các giá trị trong từng đoạn của chương trình để kiểm tra các giá trị và việc thực thi các câu lệnh
* Sử dụng debugger – một chương trình cho phép quan sát quá trình thực thi của một ứng dụng
* Các IDE thông thường cũng tích hợp sẵn debugger
* Các công cụ debug của trình duyệt

1. Biến là một tên gọi được gắn cho 1 vùng nhớ chứa dữ liệu
2. Cú pháp: let username; hoặc var username; hoặc const username;
3. Quy tắc đặt tên biến:

* Tên biến phải bắt đầu bằng một ký tự alphabet (a-zA-z\_)
* Theo sau ký tự đầu có thể là các ký tự chữ, số …
* Nên tránh đặt tên biến trùng tên các từ khoá
* Tên biến nên mô tả được ý nghĩa của nó
* Tránh dùng các ký tự gây lầm lẫn
* Tên biến có phân biệt chữ hoa và chữ thường
* Nên áp dụng các quy ước đặt tên biến chuẩn khi lập trình

1. Kiểu dữ liệu là một cách phân loại dữ liệu cho trình biên dịch hoặc thông dịch hiểu các lập trình viên muốn sử dụng dữ liệu.
2. Các kiểu dữ liệu khác nhau được lưu trữ trong biến là:

* Số (numbers)

+ Số nguyên: 10 hay 83839

+ Số thực: 15.33 hay 23.6677

+ Số dương: 3, 4

+ Số âm: -6, -7

* Chuỗi: "Hello"
* Ký tự: 'A'
* Logic: true, false

1. Có nhiều loại toán tử khác nhau:

* Toán tử toán học
* Toán tử gán
* Toán tử cộng chuỗi
* Toán tử so sánh
* Toán tử logic
* Toán tử typeof

Độ ưu tiên của các toán tử: Trong một biểu thức có nhiều phép toán thì chúng sẽ lần lượt được đánh giá dựa vào độ ưu tiên. Có thể sử dụng dấu ngoặc “()” để thay đổi độ ưu tiên của các toán tử. Các toán tử có cùng độ ưu tiên thì sẽ thực hiện từ trái sang phải

1. - Toán tử **==** so sánh trừu tượng (abstract equality), tức là nó thực hiện các chuyển đổi loại cần thiết trước khi so sánh đẳng thức. Cú pháp so sánh: **a == b**

- Còn toán tử **===** so sánh cân bằng nghiêm ngặt (strict equality), nghĩa là nó sẽ không thực hiện chuyển đổi loại. Do đó nếu hai giá trị không cùng loại, thì khi so sánh, kết quả sẽ trả về **false.**Cú pháp so sánh: **a === b**.

1. Nhập vào điểm của học viên thi cuối module. Nếu điểm >=75 thì in (“pass module”) :

let diem = prompt (‘ nhap diem: ’);

if ( diem >= 75 ) console.log (“ pass module ”)

1. let a = prompt (‘nhap ngay trong tuan: ’);

if (a == 1) console.log (“chủ nhật”)

else if (a == 2) console.log (“thứ hai”)

else if (a == 3) console.log (“thứ ba”)

else if (a == 4) console.log (“thứ tư”)

else if (a == 5) console.log (“thứ năm”)

else if (a == 6) console.log (“thứ sáu”)

else if (a == 7) console.log (“thứ bảy”)

else console (“khác”)

Cách khác:

let day;

switch () {

case 1:

day = "chủ nhật";

break;

case 2:

day = "thứ hai";

break;

case 3:

day = "thứ ba";

break;

case 4:

day = "thứ tư";

break;

case 5:

day = "thứ năm";

break;

case 6:

day = "thứ sáu";

break;

case 7:

day = "thứ bảy";

break;

case 8:

day = "khác";

}

1. Vòng lặp cho phép tự động thực hiện một khối lệnh lặp đi lặp lại nhiều lần dựa vào một điều kiện cho trước. Vòng lặp giúp cho lập trình viên viết được các mã nguồn ngắn gọn hơn so với việc phải viết lặp lại những dòng mã tương tự nhau
2. Các vòng lặp hỗ trợ bởi JavaScript: for, while và do-while.

Cú pháp: for (initial-action; loop-continuation-condition; action-after-each-iteration) { statement(s); }

Trong đó:

• initial-action: là các câu lệnh được thực thi một lần duy nhất khi vòng lặp bắt đầu chạy

• loop-continuation-condition: là biểu thức điều kiện để xác định xem vòng lặp có được tiếp tục hay không

• statement(s): là khối lệnh sẽ được thực thi trong mỗi lần lặp

• action-after-each-iteration: là các câu lệnh được thực thi sau mỗi lần lặp

Luồng thực thi của for:

1. Các câu lệnh initial-action được thực thi

2. Biểu thức điều kiện được đánh giá. Nếu điều kiện đúng thì khối lệnh bên trong thân vòng lặp được thực thi. Nếu điều kiện sai thì vòng lặp kết thúc

3. Khối lệnh bên trong thân vòng lặp được thực thi

4. Các câu lệnh action-after-each-iteration được thực thi

5. Bắt đầu vòng lặp mới từ bước 2 – 3 – 4

1. - Vòng lặp while thực thi lặp lại một khối lệnh nếu biểu thức điều kiện trả về giá trị đúng

- do….while tương tự như while nhưng thực thi luôn ở lần đầu tiên không cần kiểm tra điều kiện.

A white paper with black text

Description automatically generated

1. -Break: vòng lặp sẽ kết thúc khi gặp break

-Continue: những câu lệnh của lần lặp hiện tại sẽ bị bỏ qua

1. Cú pháp của hàm: function name (parameter1, parameter2, parameter3) {

Code to be excuted

}

Các thành phần quan trọng của hàm :

* name: Tên hàm
* parameter1, parameter2, parameter 3: Tham số
* code to be executed: phần thân hàm (các lệnh thực thi hàm)

1. • Tham số (còn được gọi đầy đủ là tham số hình thức – formal parameter) là các biến được khai báo trong phần header

• Khi gọi hàm thì giá trị của các biến này sẽ được truyền vào. Các giá trị này được gọi là tham số thực (actual parameter) hoặc đối số (argument)

1. Phạm vi (scope) của biến là các vị trí trong chương trình mà một biến có thể được sử dụng

• Một biến được khai báo trong một phương thức thì được gọi là biến địa phương (local variable)

• Phạm vi của biến địa phương bắt đầu từ vị trí nó được khai báo cho đến điểm kết thúc của khối lệnh chứa nó

• Một biến địa phương cần được khai báo và gán giá trị trước khi sử dụng

• Tham số của hàm cũng là các biến địa phương

• Phạm vi của các tham số là trong toàn bộ hàm đó