**FORM**

**Ý nghĩa biểu mẫu form?:**biểu mẫu là cơ chế cho phép người dùng nhập dữ liệu và gởi về sver:form đăng nhập,khảo sát,tìm kiếm…

**1 sốThuộc tính của form:**

<form> :thẻ form được sử dụng để tạo một form

Thuộc tính acion có giá trị là một URL,qui định nơi dữ liệu được gởi đến.

Thuộc tính name qui định tên của form sau đó được dùng để truy suất các giá trị trong form.

Thẻ<fieldset> và<legend> được dùng để nhóm các trường dữ liệu.

**Phương thức POST và GET**: thuộc tính method được dùng để quy định cách thức dữ liệu được gởi lên sever.

<form action=”/action\_page.php method=”post”

|  |  |
| --- | --- |
| **GET** | **POST** |
| * Dữ liệu đươc gởi đi sẽ hiển thị trên thanh địa chỉ của trình duyệt * Không nên sử dụng để gởi các dữ liệu nhạy cảm * Thường để gởi những dữ liệu nhỏ | * Dữ liệu đươc gởi đi sẽ ko hiển thị trên thanh địa chỉ của trình duyệt * Được sử dụng. * Không hạn chế dữ liệu gởi đi.thường để sử dụng gởi/tải dữ liệu lớn lên sever |

**TABLE (bảng):**  được sử dụng để hiển thị dữ liệu dưới dạng bảng và cột

|  |  |
| --- | --- |
| Sử dụng bảng khi muốn hiển thị một danh sách các đối tượng với nhiều thuộc tính khác nhau  • Cách thành phần của bảng:   1. Cột (column) 2. Dòng (row) 3. Đường viền (border) 4. Tiêu đề (header)   • Các thẻ được dùng đẻ tạo bảng:   * <table>:hiển thị một bảng * <tr>:khai báo một dòng * <td>:khai báo một ô tiêu đề * <th>:khai báo một ô dữ liệu |  |

**SS7 \_Hàm built-in (GỌI THÔNG BÁO)**

Console – Alert – Prompt – Confirm – setTimeout (chạy một đoạn code 1 lần trong sau một khoảng thời gian)- setInterval(chạy 1 đoạn code liên tục sau một khoảng thời gian)

**TOÁN TỬ**

**Toán tử số học:** var=a **+ -\* /** b ; % (chia lấy dư); **\*\***(lũy thừa); **++** (tăng 1 giá trị số); **--** (giảm một giá trị số).

**Toán tử gán:** var fullNam= “tranquocdat”

Ví dụ: x +=y tương đương x=x+y

x-=y tương đương x=x-y. Tương tự vs các phép toán x / \*\*

* x++ tăng giá trị biến lên 1 và kết quả trả về giá trị **trước** khi tăng
* ++x tăng giá trị biến lên 1 và kết quả trả về giá trị **sau** khi tăng
* x-- giảm giá trị biến xuống 1 và kết quả trả về giá trị **trước** khi giảm
* --x giảm giá trị biến xuống 1 và kết quảtrả về giá trị **sau** khi giảm

**Toán tử so sánh:** var a=1; var b=2; if{(a**> <=**b) thì thực hiện code}

==; >= ; <= ; != (khác) ; Chỉ cho ra giá trị boolean (true;false)

**Toán thử logic:** var a=1; var b=2; if{ (a>0 **&&** b>0 thì thực hiện code}

&&: và ; **!**: Not ; or

**Toán tử gán :(string operator)**

**KIỂU DỮ LIỆU**

**1.Kiểu dữ liệu nguyên thủy:-primitive data**

Number – string – Boolean -Undifined *(định nghĩa ra 1 biến nhưng không gán giá trị cho nó)* – Nul*(không có gì…hiển thị thẳng ra keyword “null”)* – Symbol

**2.Kiểu dữ liệu phức tạp:** Function – Object

*Lưu ý: khi muốn kiểm tra kiểu dữ liệu ta dùng lệnh* **console.log(typeof tên\_biến)**

**SS8\_CÂU LỆNH ĐIỀU KHIỂN**

Một chương trình phần mềm thực thi các câu lệnh theo trật tự từ  
trên xuống dưới  
• Có thể thay đổi luồng thực thi của một chương trình bằng cách sử  
dụng các câu lệnh điều khiển luồng (control flow statement)  
• Các câu lệnh điều khiển của JavaScript:  
• Câu lệnh điều kiện (conditional statement)  
• Câu lệnh lặp (Loop statement)  
• Câu lệnh nhảy (jump statement)

**Câu lệnh điều kiện** còn được gọi là câu lệnh ra quyết định (decision  
making)  
• Cho phép thay đổi luồng thực thi của chương trình  
• Lựa chọn thực thi một khối lệnh dựa trên việc đánh giá một điều kiện  
cho trước  
• JavaScript hỗ trợ các câu lệnh điều kiện:  
• if  
• switch-case

**Câu lệnh if:**Đánh giá một điều kiện và đưa ra lựa chọn thực  
thi một khối lệnh dựa trên điều kiện đó  
• Nếu điều kiện có giá trị đúng (true) thì khối lệnh bên trong if sẽ được thực thi  
• Nếu điều kiện có giá trị sai (false) thì khối lệnh bên trong if sẽ được bỏ qua. Luồng thực thi sẽ chuyển xuống ngay sau khối lệnh if.

Cú pháp: ***if (condition) { one or more statements}***  
• condition: là biểu thức trả về giá trị kiểu boolean  
• statements: Các câu lệnh sẽ được thực thi nếu điều kiện trả về **true**

**Câu lệnh if-else** lựa chọn thực thi 1 trong 2 khối lệnh thông qua việc đánh giá một điều kiện

Cú pháp: ***if (condition) { one or more statements***

***} else { one or more statements}***

• condition: điều kiện để đánh giá. Nếu condition trả về **true** thì khối lệnh bên trong **if**được thực thi. Nếu condition trả về **false** thì khối lệnh trong **else** được thực thi.

**Câu lệnh if lồng trong if :** Một câu lệnh if có thể được đặt trong câu lệnh if khác:

**Cú pháp:** *if(condition1) {  
if(condition2) { true-block statement(s);}  
else { false-block statement(s); } }  
else { false-block statement(s);}*

**Câu lệnh if bậc thang:** Có thể đặt các câu lệnh điều kiện if-else liên tiếp nhau

**Cú pháp** *if(condition) { one or more statements }  
else if (condition) { one or more statements }  
else { one or more statements }*

**Câu lệnh switch-case:**

switch-case là một cấu trúc điều kiện cho phép lựa chọn thực thi các khối lệnh khác nhau dựa trên kết quả của việc so sánh  
• switch-case so sánh giá trị của một biến với lần lượt từng giá trị một,nếu có giá trị phù hợp với biến thì khối lệnh tương ứng sẽ được thực thi  
• switch-case không thể thay thế if-else trong tất cả các trường hợp

**So sánh if vs switch-case**

|  |  |
| --- | --- |
| **if** | **switch-case** |
| Có thể sử dụng để so sánh lớn hơn, nhỏ hơn… | Chỉ có thể sử dụng để so sánh bằng hoặc khác nhau |
| Mỗi câu lệnh if có một biểu thức điều kiện, với giá trị trả về là **true** hoặc **false** | Tất cả các trường hợp (case) đều so sánh với giá trị của biểu thức điều kiện duy nhất |
| Biểu thức điều kiện cần trả về giá trị kiểu b**oolean** | Biểu thức điều kiện cần trả về giá trị là kiểu **byte**, **short**, **char**, **int**, hoặc **String** |
| Chỉ có một khối lệnh được thực thi nếu điều kiện đúng | Nếu điều kiện đúng mà không có câulệnh break thì tất cả các khối lệnh ở phía sau cũng được thực thi |

**CẤU TRÚC ĐIỀU KIỆN 2**

**Kiểu thay đổi vị trí thuộc tính:** *.style.position*

Trả lại thuộc tính vị trí: *object.style.position*

Đặt lại thuộc tính vị trí:

*object.style.position = “static || absolute|| fixed|| relative|| sticky|| initial|| inherit"*

**Đặt bên trái thuộc tính:** *.style.left*

Trả lại tài sản bên trái : *object.style.left*

Đặt thuộc tính bên trái : *object.style.left = "auto| length |%| initial | inherit"*

**Thẻ sự kiện : onload=” “**

**Câu lệnh lặp**

**Khái niệm về vòng lặp (loop) trong js?** 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  
•Vòng lặp bao gồm:  
• Một điều kiện để đánh giá (lúc nào thì lặp?)  
• Một khối lệnh để thực thi (lặp cái gì?)  
•Các vòng lặp hỗ trợ bởi JavaScript: for, *while và do-while*

**Cấu trúc vòng lặp for**

***for*** *(initial-action; loop-continuation-condition; action-after-each-iteration) { statement(s); }*  
Trong đó:  
*•(1) initial-action*: là các câu lệnh được thực thi một lần duy nhất khi vòng lặp bắt đầu chạy  
*•(2) loop-continuation-condition*: là biểu thức điều kiện để xác định xem vòng lặp có được tiếp tục hay không.  
• *(3)statement(s)*: là khối lệnh sẽ được thực thi trong mỗi lần lặp  
*• (4)action-after-each-iteration*: là các câu lệnh đượcthực thi sau mỗi lần lặp

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)

**Các loại vòng lặp trong js?** for, *while và do-while*

**Vòng lặp while**  
•Vòng lặp *while* thực thi lặp lại một khối lệnh nếu biểu thức điều kiện trả về giá trị đúng  
•Cú pháp:

while (loop-continuation-condition) {statement(s);}

Trong đó:  
*•* loop-continuation-condition : là biểu thức điều kiện  
*•* statement(s): là các câu lệnh được thực thi trong mỗi lần lặp

***LƯU Ý***: **Lựa chọn while hay for?**•Các loại vòng lặp có thể được sử dụng thay thế cho nhau  
•Việc lựa chọn vòng lặp cho phù hợp với từng tình huống là quan trọng  
•Khác biệt thông thường nhất giữa for và while:  
• for thường được sử dụng khi biết trước số lần lặp  
• while thường được sử dụng khi không biết số lần lặp

Vòng lặp do-while  
•Cú pháp: do { statements(s);} while (loop-continuation-condition);  
Trong đó:  
• statement(s): Các câu lệnh được thực thi trong mỗi lần lặp  
• loop-continuation-condition: Biểu thức điều kiện. Nếu biểu thức điều kiện trả về giá trị true thì vòng lặp sẽ tiếp tục thực thi. Nếu biểu thức điều kiện trả về false thì vòng lặp kết thúc  
**•Điểm khác biệt giữa while và do-while:** Thân của vòng lặp do-while được thực hiện ít nhất là một lần. Kể cả khi biểu thức điều kiện trả về sai.

**Câu lệnh break**• Lệnh break dùng để thoát khỏi vòng lặp mà không đi tới điểm cuối  
• Khi gặp câu lệnh này trong vòng lặp, chương trình sẽ thoát ra khỏi vòng lặp và chỉ đến câu lệnh liền sau nó  
• Nếu nhiều vòng lặp thì break sẽ thoát ra khỏi vòng lặp gần nhất.  
• Ngoài ra, break còn được dùng trong cấu trúc lựa chọn switch

**Câu lệnh continue**•Lệnh continue để nhảy qua một phần của khối lệnh và bắt đầu dòng tiếp theo.  
•Khi gặp lệnh này trong các vòng lặp, chương trình sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện lần lặp tiếp theo.

**SS12\_MẢNG**

**1.Mảng là gì? Cú pháp khai báo?**

Trong các chương trình máy tính, có những trường hợp chúng ta phải lưu trữ rất nhiều dữ liệu  
•Chẳng hạn, cần lưu trữ danh sách tên của hằng trăm sinh viên, cùng điểm thi của từng người  
•Việc khai báo hằng trăm biến để lưu trữ các dữ liệu kiểu này rất mất thời gian và không thực tế  
•JavaScript và phần lớn các ngôn ngữ khác hỗ trợ một cấu trúc dữ liệu là mảng để giải quyết các tình huống này

**Mảng là một loại biến đặc biệt**, có thể lưu được nhiều giá trị thay vì chỉ một giá trị như các biến thông thường  
•Mỗi giá trị trong mảng được gọi là một phần tử  
•Các phần tử được lưu trữ ở các vị trí kế tiếp nhau trong bộ nhớ  
**Tên mảng:** Tuân thủ theo quy tắc đặt tên của biến  
•Phần tử: Các giá trị được lưu trữ trong mảng  
•Chỉ số: Vị trí của các phần tử trong mảng. Chỉ số bắt đầu từ 0.  
•Độ dài: Số lượng các phần tử của mảng

**Cú pháp cách 1:** var array\_name= [ iteam1,iteam2…]

**Cú pháp cách 2:** **:** var array\_name= new\_array( iteam1,iteam2…)

**2.Các cách duyệt mảng**

**Vòng lặp for each** cho phép thao tác vs từng phần tử.Nhận vào tham số là 1 hàm trong đó có các phần tử của mảng

**Vòng lặp for-in** (còn gọi là *enhanced for*) được sử dụng để duyệt qua các phần tử của một collection, chẳng hạn như mảng, danh sách…  
•Cú pháp:  **for** (var **in** collection) { }

• var: Biến đại diện lần lượt cho từng phần tử của collection trong mỗi lần lặp  
• collection: đối tượng cần lặp

**Vòng lặp for :**Sử dụng vòng lặp for để duyệt qua tất cả các phần tử của mảng

**Khác biệt khi sử dụng for và for-in**•Có thể sử dụng index khi duyệt bằng for, không có index nếu sử dụng for-in  
•Có thể sử dụng for để duyệt theo các chiều khác nhau (từ đầu đến cuối, từ cuối đến đầu…)  
•For in chỉ duyệt từ đầu đến cuối mảng.

**3.Mảng đa chiều**•Mảng đa chiều là mảng có các phần tử là các mảng khác  
•Có thể có mảng 2 chiều, 3 chiều… hoặc nhiều hơn  
•Mảng càng nhiều chiều thì độ phức tạp khi xử lý càng cao

**.indexof():**

* Trả về vị trí của lần xuất hiện đầu tiên của một giá trị trong một chuỗi.
* Thức trả về -1 nếu giá trị không được tìm thấy.
* Phân biệt chữ hoa và chữ thường.

**Chuỗi JavaScript toUpperCase ():** Chuyển sang chữ hoa:

* Chuyển đổi một chuỗi thành chữ hoa.
* Không thay đổi chuỗi ban đầu.

**JavaScript toLowerCase ():**Chuyển sang chữ thường:

* Chuyển đổi một chuỗi thành chữ thường.
* Không thay đổi chuỗi ban đầu.

**SS13\_FUNCTIONS**

**1.Hàm là gì?**

Hàm (function) là một nhóm các câu lệnh thực hiện một nhiệm vụ nhất định

• Hàm là thuật ngữ được sử dụng phổ biến trong Lập trình hướng đối tượng. Trong nhiều trường hợp khác, các tên gọi được sử dụng là phương thức (method) và thủ tục (procedure)

• console.log(), Math.pow(), Math.random() là các hàm đã được định nghĩa sẵn cho chúng ta sử dụng

Hàm là chiếc hộp đen

• Có thể hình dung hàm như là những chiếc hộp đen có công dụng thực hiện các nhiệm vụ nhất định

• Tham số là đầu vào của chiếc hộp

• Giá trị trả về là đầu ra của chiếc hộp

• Người sử dụng hàm không quan tâm đến bên trong chiếc hộp, chỉ quan tâm đến đầu vào và đầu ra

**2.Cú pháp khai báo hàm:**

Cú pháp của hàm

Function name (paramerter1,parameter2){ code to be executed}

• 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)

**3. Cách gọi hàm?**

Gọi (call hoặc invoke) phương thức là cách để thực thi một hàm đã được định nghĩa trước đó

• Khi gọi hàm thì cần truyền đối số vào

• Ví dụ, gọi hàm không có giá trị trả về: console.log("Welcome to Java!");

• Ví dụ, gọi hàm có giá trị trả về: var larger = max(3, 4);

**4.Tham số và đối số?**

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)

**5.Phân biệt hàm có return và hàm không có return?**

Giá trị trả về

• Giá trị trả về là kết quả sẽ được trả về tại vị trí hàm được gọi

• Mỗi một hàm chỉ có một giá trị trả về. Các giá trị này có thể là một biến, một mảng hay một đối tượng, danh sách đối tượng.

Câu lệnh return được sử dụng để trả về giá trị của một phương thức. **return**được gọi trong một hàm, thì việc thực thi hàm này sẽ bị dừng lại.các câu lệnh sau return không được thực hiện.Phương thức trả về nhiều kiểu dữ liệu.

Hàm không có return giá trị trả về là undefine-không thể thay đổi nội dung thông báo mỗi khi sử dụng.

**5. Tham trị và tham chiếu?**

-Kiểu tham trị (Value Types) Kiểu này dùng cho các kiểu dữ liệu nguyên thuỷ: String, Number, Boolean, BigInt, Symbol, undefined, null.

Kiểu tham trị thì lưu lại giá trị trong ô nhớ, ghi gán cho biến khác và sửa thì sẽ không làm thay đổi giá trị trước khi gán.

-Kiểu tham chiếu thì phức tạp hơn kiểu tham trị và được dùng cho các kiểu dữ liệu phức tạp: Object, Array, Function. Còn kiểu tham chiếu bản chất sẽ lưu lại địa chỉ ô nhớ chứa giá trị chứ không lưu giá trị, khi gán cho một biến khác thì sẽ gán địa chỉ này và khi sửa thì địa chỉ này không đổi nên các biến trỏ đến cùng địa chỉ này sẽ thay đổi khi có một cái nào đó thay đổi. Để tránh việc làm thay đổi giá trị kiểu tham chiếu thì ta clone ra một object mới rồi thay đổi hoặc chuyển object thành chuỗi rồi parse lại thành object để tạo ra object mới ở ô nhớ mới.

**SS14\_Lập Trình Hướng Đối Tượng**

**1.Ngôn ngữ Hướng đối tượng (OOP)**• Không tập trung vào cấu trúc như ngôn ngữ thủ tục, mà tập trung vào mô hình hoá dữ liệu (data modeling)  
• Lập trình viên sử dụng các Lớp (class) khi lập trình

• Mô phỏng các đối tượng trong thế giới thực vào trong các chương trình.  
• Class: khuôn mẫu của dữ liệu được mô hình hoá  
• Ví dụ: C++, C#, Java, PHP, Javascript…

OOP (Object Oriented Programming) là một triết lý thiết kế chương trình  
• Có nhiều ngôn ngữ lập trình hỗ trợ OOP  
• Tất cả mọi thứ trong OOP đều là “đối tượng”. Một chương trình phần mềm được coi như là

một thế giới bao gồm các đối tượng tương tác với nhau.  
• Mã lệnh và dữ liệu được kết hợp trong một thể thống nhất – đó là *đối tượng*.  
• Đối tượng bao gồm:  
• Thuộc tính: các dữ liệu, tính chất của đối tượng  
• Hành vi: các khả năng, hành động mà đối tượng có thể thực hiện  
• Các đối tượng có thể có quan hệ với nhau.

**2.Phân biệt class và object.**

**Class là bản thiết kế hoặc khuôn mẫu mà từ đó đối tượng được tạo ra**

* Là tập hộ những đối tượng có thuộc tính, hành vi những mối quan hệ thông thường giống nhau.
* 1 lớp chỉ được khai báo 1 lần
* Được khai báo vs từ khóa “class”
* Không được cấp phát bộ nhớ khi tạo
* Là 1 thực thể logic vd human

**Object là một thể thiện của lớp**

* Là một thực theerr trong thế giới thực
* có những **thuộc tính** hay những đặc điểm mô tả và  
  những **hành vi** riêng nhằm phân biệt nó với các đối tượng khác.
* 1 đối tượng được khai báo nhiều lần theo nhu cầu
* Được khai báo vs từ khóa “new” or tên lớp
* Được phân bộ nhớ mỗi khi khởi tạo
* Là một thực thể vật lý

**3.Contructor là gì**

Phương thức khởi tạo (Constructor) là một phương thức đặc biệt với các đặc điểm:

• Tên phương thức trùng với tên lớp

• Không có kiểu dữ liệu trả về

• Có các tham số đầu vào như phương thức thông thường khác

• Phương thức khởi tạo được sử dụng để:

• Tạo đối tượng

• Khởi tạo các giá trị ban đầu cho các thuộc tính của lớp

• Phương thức khởi tạo được gọi ngay khi một đối tượng được tạo ra bởi từ khoá new

**4.Cách tạo class trong js**

-use hàm: sử dụng bên trong – khai báo bằng thuộc tính prototype

-use object: var object name:{ } or var Object Name=new object ( );

-use Singleton object

**Ss15\_OBJECT AND CLASS**

**1.Tính đồng gói là gì ?**

Khả năng truy suất vào các thành phần của một đối tượng trong khi  
vẫn đảm bảo che giấu các đặc tính riêng tư bên trong đối tượng được  
gọi là tính bao gói.

**2.Tính kế thừa là gì?**

Tính kế thừa cho phép các đối tượng có thể chia sẻ hay mở rộng các  
thuộc tính hoặc phương thức mà không phải tiến hành định nghĩa lại.

**3.Tính trừu tượng là gì?**

Loại bỏ những thuộc tính và hành vi không quan trọng của đối tượng,  
chỉ giữ lại những thuộc tính và hành vi có liên quan đến vấn đề đang  
giải quyết.

**4.Tính đa hình là gì?**

Tính đa hình thể hiện khi với cùng một phương thức nhưng có thể có  
cách ứng xử khác nhau ở những lớp cùng giao diện