**SESSION 2 : BIẾN**

1. Biến

* Lưu trữ giá trị thay đổi
* Cách khai báo:
  + let + (Tên biến) = (Giá trị) : Khai báo giá trị có thể thay đổi
  + const + (Tên biến) = (Giá trị) : : Khai báo giá trị không đổi
  + var + (Tên biến) = (Giá trị) : Khai báo giá trị có thể thay đổi
* Quy tắc đặt tên biến
  + Không được bắt đầu bằng số
  + Chỉ được chứa các ký từ a-z, A-Z, số và “\_” “$”
  + Không được chứa dấu cách
  + Không trùng với từ khóa của ngôn ngữ
* Đặc điểm: Tên biến có phân biệt chữ hoa và thường
* Gợi ý về cách đặt tên: Đặt tên theo cái mà nó mô tả về
* Giá trị của biến nếu là chữ phải đặt trong “ ” hoặc ‘ ’
* Dấu “=” trong lập trình có hướng từ phải sáng trái

X = 5 không có nghĩa là 5 = X;

* Phạm vi biến:
  + let : chỉ có tác dụng trong block code { }
  + var : có tác dụng trong toàn phạm vi
* Cơ chế Hoisting: Kéo lên, Toàn bộ các hàm var được kéo lên trên đầu
* Tài liệu: w3shool , stackoverflow

ưerwerwer

**SESSION 3: KIỂU DỮ LIỆU (DATATYPES)**

* Number
  + Integer: số nguyên
  + Float: số thực
  + Các kiểu tính toán :
    - + , - , \* , / , % (lấy phần dư)
    - ++ , --
      * X++ , X-- tăng, giảm X lên 1 đơn vị sau khi thực hiện hành động;
      * ++X, X-- tăng, giảm X lên 1 đơn vị trước khi thực hiện hành động;
* Chuỗi: Khai báo trong **“ ”** hoặc **‘ ’** hoặc **` `**
  + VD: “Giá trị của z là: ” + z
  + ⬄ `Giá trị của z là ${z}`
* Null
* Undefined
* Object
* Array

**Câu lệnh Prompt**

**Bài tập:**

**Bài 1: Hỏi người dùng năm sinh – In ra tuổi người dùng**

**Bài 2: Hỏi tên người dùng**

**Hỏi năm sinh người dùng**

**In ra: Chào bạn [tên người dùng], bạn [Tuổi]**

**Bài về nhà: Hỏi tên và tuổi người dùng , cách nhau bởi dấu phẩy. In ra Tên bạn là ?, bạn ? tuổi (Tham khảo tại w3shool)**

**Hỏi tên và năm sinh người dùng , cách nhau bởi dấu phẩy. In ra Tên bạn là ?, bạn ? tuổi (Tham khảo tại w3shool)**

**SESSION 4 : VÒNG LẶP**

1. Vấn đề: 1 việc lặp lại nhiều lần
2. Cú pháp:

* **Biết rõ số lần lặp:**

**for (let index = 0 ; index <5 ; index++) {**

**}**

* **Không biết rõ số lần lặp lại:**

**While (điều kiện) {**

**Hành động lặp lại**

**}**

**Vòng 1 :**

**+ Kiểm tra điều kiện .**

**+ Nếu đúng, thực hiện hành động**

**Vòng 2 :**

**+ Kiểm tra điều kiện .**

**+ Nếu đúng, thực hiện hành động**

**.**

**.**

**.**

**Vòng N : Điều kiện sai, dừng**

a = a + 5 ⬄ a += 5 (Tương tự cho các phép tính khác)

Hàm Strng.includes() (Có bao gồm ký tự ? trong chuỗi. Kết quả trả về true nếu có, false nếu không có )

Kiểu dữ liệu Boolean : True hoặc False

Dấu Khác : >< hoặc !=

**Bài về nhà :**

**B1: Nhập mật khẩu. Yêu cầu : Độ dài >8 và chưa ký tự đặc biệt $ , % , \_**

**B2: Yêu cầu người dùng nhập 2 số a, b.**

**In ra dấu \* thành hình chữ nhật , tam giác trong đó a, b là chiều dài cạnh**

**SESSION 5: CÂU LỆNH ĐIỀU KIỆN**

1. Điều kiện: (Condition):

* Kết quả là : True – False
* Các cú pháp về điều kiện:
  + Toán tử so sánh: Bằng == , khác != , > , < , >= , <= , === , !==
  + Toán tử LOGIC : and && , or || , not !

Ví dụ : Chuyển các phép toán đại số sang lập trình

* a != b , a >= b , a <=b
* a < b <=c , a < b , a >= c

1. Cú pháp

* Câu lệnh if ( condition) {

Code

} Nếu điều kiện là đúng thì code được chạy

else { Nếu điều kiện là sai thì code 2 được chạy

Code

}

* If nhiều điêu kiện:
* if ( condition) {

Code1

}

else if (condition) {

Code2

}

else if (condition) {

Code3

}

Bài tập:

Ax + b = 0 . Tìm nghiệm x

Ax2 + bx + c = 0

* switch (biến) {

case giá trị 1: = if

break;

case giá trị 2: = else if

break;

case giá trị 3:

break;

.

.

.

Default: = else

}

Switch case chỉ so sánh bằng ==

**SESSION 6: MẢNG**

1. Vấn đề

* Đa giác n cạnh . Tính chu vi. In ra độ dài các cạnh
* Cú pháp:

**Let tenbien = [Giá trị 1, 2, 3, 4, ....];**

Độ dài của mảng = tổng số giá trị mảng lưu trữ = **tenbien.length**

Truy cập từng giá trị:

**Tenbien[index] : Giá trị tại chỉ số index**

**Thêm giá trị:**

**Tenbien.push(giá trị) = thêm giá trị vào cuối mảng**

**Tenbien.unshift(giá trị) = thêm giá trị vào đầu mảng**

**Tenbien[index] = gia tri; thêm giá trị vào vị trí index**

**Delete tenbien[index];**

**Bài tập:**

**MENU**

1. **Danh sách sách trong thu viện.**
2. **Thêm sách**
3. **Đổi tên sách**
4. **Xóa sách**
5. **Thoát**

**SESSION 7: OBJECT**

1. **Vấn đề:**

Object:

Let person ={

Key1 : value1, // Cặp key - value

Key2 : value2,

.

.

}

1. Quy tắc:

* Key phải là duy nhất, nếu trùng thì nhận Key khai báo sau
* Value : Lưu trữ bất kỳ kiểu dữ liệu nào mà JS có: Number, string, array, object, function
* Cách khai báo:
  + Key : Nằm trong dấu “ ” , ‘ ’, không có nháy

1. Cách sử dụng:

* Array: array[index] => Lấy value tại index
* Object:
  + Object [key]
  + Object[“key”]
  + Object.key
* Object.key có kiểu dữ liệu khi khai báo hoặc khi bị thay đổi
* Thêm cặp key : value
  + Object[“key mới”] = value;
  + Object.keymoi = value;
* Xóa cặp key:value
  + Delete object[“key”];
  + Delete object.key;
* For in dùng cho Object
* for (let key in object) {
* if (object.hasOwnProperty(key)) {
* const element = object[key];
* console.log(element);
* }
* }

**SESSION 8: FUNCTION**

1. Vấn đề:
2. Định nghĩa: Là 1 khối code thực hiện 1 nhiệm vụ cụ thể nào đó, chỉ được thực thi khi được gọi đến.

Dấu ngoặc vuông: có thể có hoặc không

1. Cú pháp:
   1. ***Cách 1***

**Function [ten function] ([danh sách tham số]) {**

Code thực hiện nhiệm vụ

**return** - Kết quả trả về

**}**

* 1. ***Cách 2***

Let tenfunction = function([Danh sách tham số]){

Code

**return** - Kết quả trả về

};

1. Cách gọi: tên function([danh sách biến]);
2. Lưu ý: Function chỉ nên thực hiện 1 nhiệm vụ cụ thể, đủ nhỏ để đặt tên

* Number (int, float) , string : Kiểu dữ liệu nguyên thủy. Khi sử dụng vào function tham số nhận giá trị tham trị
* Array, object: Khi sử dụng vào function tham số nhận giá trị tham chiếu
* **Đệ quy:**

**SESSION 9 : HTML Hyper text markup language**

Để tạo website có 3 thành phần:

* HTML
* CSS
* Javascript

HTML bao gồm các thẻ, mỗi thẻ có chức năng nhất định.

* Thẻ <html> : Bao gồm toàn bộ code html bên trong.
* Thẻ head : bao gồm các setting cho trang html
* Thẻ body : Chứa toàn bộ nội dung trang web
* Các loại thẻ HTML (tag, element, component)
  + Thẻ tự đóng : <tên thẻ/> vd : <hr/> , <br/>…
  + Cặp thẻ đóng mở : <tên thẻ> Nội dung <tên thẻ/>
* Thuộc tính :

<tên thẻ \_\_ tên thuộc tính1 = “Giá trị1\_\_ tên thuộc tính2 = “Giá trị2”…>

</tên thẻ>

* Đặc điểm : Mỗi thuộc tính chỉ được xuất hiện 1 lần trong 1 cặp thẻ.
* Thuộc tính (attribute)
  + id : Định danh
  + class: lớp

Trong 1 trang web chỉ có 1 id duy nhât, nhưng có thể có nhiều class.

* JS + HTML
  + Document : Chứa toàn bộ nội dung của HTML
  + getElementById : Tìm thẻ có ID xác định
  + getElementsByClass (“tên class”) : Tìm danh sách các thẻ có tên class xác định
* Event : Sự kiện
  + Do tự thẻ html có
  + Do javascript định nghĩa
* Thẻ table
  + tr
  + td
  + th
  + thead
  + tbody
  + tfoot
* remove element html : remove (java)
* appendChild
* document.createElement

button – click –onlick – function

this event

**CSS**

3 cách để viết CSS

|  |  |  |
| --- | --- | --- |
| **TT ưu tiên** | **Cách viêt** | **Cú pháp** |
| 1 | Viết ngày trong thẻ html : thuộc tính style = “code css” . | tên thuộc tính: giá trị; |
| 2 | viết trong thẻ head, body…. : Đặt code trong cặp thẻ <style> code CSS </style> | selector {  tên thuộc tính : giá trị;  tên thuộc tính : giá trị;  .  .  .  }   1. tên thẻ   h1 {  tên thuộc tính : giá trị;  }   1. id   #tên ID {  tên thuộc tính : giá trị;  }   1. class   .tên class {  tên thuộc tính : giá trị;  } |
| 3 | đặt ra ngoài thành file .css riêng : dùng thẻ link để nhúng file vào html  <link href = “file css” rel = “stylesheet” /> |

sử dụng dấu “cách” áp dụng cho tất cả, sử dụng dấu > áp dụng cho con trực tiếp

p1 id = thẻ 1 -> class thẻ 2 thẻ 3

p2 id = thẻ 2

**Tìm hiểu về Ajax JS – JSON – Ajax**

**JSON:** là 1 chuỗi có cấu trúc

* JSON.stringify(biến) : Trả về 1 chuối JSON của biến
* JSON.parse(chuối json): Trả về 1 JS Object

**AJAX**: XML HTTP request

* Phương thức (Method)
  + get : gửi và nhận dữ liệu
  + post: gửi và nhận dữ liệu
  + put: update dữ liệu

CODE game : tìm hiểu thẻ canvas của html

getbootstrap.com giao diện