Frontend Development

JavaScript

Green Academy
06/03/2022

Lập trình Java Script

GIỚI THIỆU JAVASCRIPT

GIỚI THIỆU JAVASCRIPT

- JS là ngôn ngữ script ở client, dùng để xử lý và tương tác với các thành phần HTML.
- Ngôn ngữ thông dịch, mã lệnh được thông dịch trực tiếp ngay khi thực thi.
- Cung cấp cho người thiết kế HTML công cụ lập trình
- Cho phép đặt đoạn văn bản động vào trang web
- Có thể tác động các sự kiện trong trang HTML
- Có thể đọc/ghi các thành phần của HTML
- Dùng để check dữ liệu từ người dùng
- Có thể check phiên bản trình duyệt
- Có thể thao tác cookie của trang web.

GIỚI THIỆU JAVASCRIPT(tt)

- **Các bước thực thi của JS**
  1. Trình duyệt tải trang web về
  2. Trình duyệt kiểm tra xem có mã JS trong web hay không
  3. Nếu có, trình duyệt sẽ chuyển mã JS cho bộ thông dịch
  4. Bộ thông dịch xử lý và thực thi các mã lệnh JS
  5. Các mã lệnh có thể tác động đến các thành phần của trang web.
  6. Trình duyệt hiển thị toàn bộ nội dung web.

GIỚI THIỆU JAVASCRIPT(tt)

- Cách đặt mã lệnh JS vào trang web
  - Internal: đặt trong head hay body

```
<head>
    <script type="text/javascript">
            .....
    </script>
</head>
```

```
<body>
    <script type="text/javascript">
            .....
    </script>
</body>
```

  - External: tạo tập tin bên ngoài và liên kết tập tin đó trong phần head.

```
<head>
        <script src="YourJsFile.js"></script>
</head>
```

Lập trình Java Script

KHAI BÁO BIẾN- CÁC TOÁN TỬ

KHAI BÁO BIẾN

- **Cách đặt tên biến**
  - Dùng các ký tự a..z, A..Z, 1..9, dấu gạch dưới '_', dấu '$'
  - Tên biến không trùng với từ khóa JS
  - Tên biến không bắt đầu bởi con số
  - Tên biến không có ký tự khoảng trắng
  - Tên biến là case sensitive.

Cú pháp	Ví dụ
var <tên biến>;	**var** n; n = 1;
var <tên biến> = <giá trị>;	**var** s = "JavaScript string";
<tên biến> = <giá trị>;	PI = 3.14;

KHAI BÁO BIẾN

- **Kiểu dữ liệu của biến**
  - JS không quy định kiểu biến khi khai báo biến, kiểu của biến sẽ được tự động xác định khi gán dữ liệu cho biến
  - Các kiểu dữ liệu của JS
    - Kiểu số (number): số nguyên, số thực
    - Kiểu chuỗi (string)
    - Kiểu luận lý (boolean): true/false
    - Kiểu đối tượng (object)
    - Kiểu hàm (function)

KHAI BÁO BIẾN

- **Xác định kiểu của biến**
  - Sử dụng toán tử typeof sẽ về các giá trị sau:
    - string
    - number
    - boolean
    - object
    - function
    - undefine

CÁC TOÁN TỬ - TOÁN HỌC

Toán tử	Mô tả	Ví dụ	Kết quả
+	Cộng (addition)	x=y+2	x=7
-	Trừ (subtraction)	x=y-2	x=3
*	Nhân (multiplication)	x=y*2	x=10
/	Chia (division)	x=y/2	x=2.5
%	Chia lấy dư (modulus)	x=y%2	x=1
++	Tăng (increment)	x=++y	x=6
--	Giảm (decrement)	x=--y	x=4

CÁC TOÁN TỬ - TOÁN TỬ GÁN

Toán tử	Ví dụ	Tương tự với
=	x=y	
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

CÁC TOÁN TỬ - SO SÁNH

Toán tử	Mô tả	Ví dụ
==	bằng	x==8
===	bằng (so sánh vừa giá trị vừa kiểu biến)	x===5
!=	không bằng	x!=8
>	lớn hơn	x>8
<	nhỏ hơn	x<8
>=	lớn hơn hay bằng	x>=8
<=	nhỏ hơn hay bằng	x<=8

CÁC TOÁN TỬ - LOGIC

Toán tử	Mô tả	Ví dụ
&&	and	(x < 10 && y > 1) → true
||	or	(x==5 || y==5) → false
!	not	!(x==y) → true

CÁC TOÁN TỬ - TOÁN TỬ ĐIỀU KIỆN

- Cú pháp:

  variablename = (*condition*) ? *value1*:*value2*

  – Điều kiện đúng *variablename chứa* value1, ngược lại chứa value2

  Ví du:

  let voteable = (age < 18) ? "Too young":"Old enough";

Lập trình Java Script

CẤU TRÚC ĐIỀU KHIỂN, VÒNG LẶP, HÀM, MẢNG, CHUỖI

CẤU TRÚC ĐIỀU KHIỂN – IF

- Cú pháp:

```
if (biểu_thức_điều kiện_1)
{
    //khối lệnh được thực hiện nếu biểu thức 1 đúng;
}
else if (biểu_thức_điều_kiện_2)
{
    //khối lệnh được thực hiện nếu biểu thức 2 đúng;
}
else
{
    //khối lệnh được thực hiện nếu cả hai biểu thức trên đều không đúng;
}
```

CẤU TRÚC ĐIỀU KHIỂN - SWITCH

- Cú pháp:

```
switch (biểu_thức_điều_kiện)
{
   case kết_quả_1 :
      //khối lệnh cần thực hiện nếu biểu_thức_điều_kiện bằng kết_quả_1;
      break;
   case kết_quả_2 :
      //khối lệnh cần thực hiện nếu biểu_thức_điều_kiện bằng kết_quả_2;
      break;
   default :
      //khối lệnh cần thực hiện nếu biểu_thức_điều_kiện cho ra một kết quả khác;
}
```

CẤU TRÚC ĐIỀU KHIỂN - FOR

```
for (bt_khởi_tạo; bt_điều_kiện; bt_thay_đổi_giá_trị) {
  // khối lệnh cần lặp

}
```

Ví dụ:

```
var cars = [‘BMW’,’Honda’];

for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

```
for (biến in đối tượng) {
  // khối lệnh cần lặp

}
```

Ví dụ:

```
var cars = [‘BMW’,’Honda’];

for (let car in cars) {
  text += car + "<br>";
}
```

CẤU TRÚC ĐIỀU KHIỂN - WHILE

- Cú pháp:

```
while (biểu_thức_điều_kiện)
{
        //khối lệnh lặp cần thực hiện nếu
        //biểu_thức_điều_kiện trả về true;
}
```

- Ví dụ:

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

- Cú pháp:

```
do
{
      //khối lệnh lặp cần thực hiện
} while (biểu_thức_điều_kiện);
```

- Ví du:

```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

Hàm (Function)

- Cú pháp:

```
function tên_hàm(đối_số_1, đối_số_2)
{
    //các câu lệnh cần thực hiện mỗi khi hàm được gọi;
    return giá_trị_cần_trả_về;
}
```

Ví du:

```
function myFunction(a, b) {
  return a * b;
}
```

Mảng (Array)

- Cách khai báo mảng 1 chiều

Cách 1: regular array

```
var myfriends=new Array();
myfriends[0]="John";      myfriends[1]="Bob";   myfriends[2]="Sue";
```

Cách 2: condensed array

```
var myfriends=new Array("John", "Bob", "Sue") ;
```

Cách 3: literal array

```
var myfriends=["John", "Bob", "Sue"] ;
```

Mảng (Array)

- Khai báo mảng 2 chiều
  - Mảng 2 chiều được xem như mảng 1 chiều với mỗi phần tử của mảng 1 chiều này là một mảng 1 chiều khác
  - Ví dụ khai báo mảng 2 chiều 3X3
  - Cách 1

```
var b = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
```

  - Cách 2

```
var c = new Array(3);
dem = 0;
for (i=0; i < 3; i++)
{
    c[i] = new Array(3);
    for (j=0; j < 3; j++)
        c[i][j] = ++dem;
}
```

Chú ý: Các phần tử của mảng không nhất thiết phải cùng kiểu với nhau

Ví dụ:

```
var c = [1, 2.2, "chuỗi"];
```

Mảng (Array)

- Các thuộc tính của đối tượng mảng

Thuộc tính	Mô tả
length	Là thuộc tính đọc/ghi, cho biết số lượng phần tử của mảng. Có thể gán số lượng phần tử cho thuộc tính này để thay đổi kích thước của mảng (mảng động)
prototype	Sử dụng thuộc tính này để thêm vào các thuộc tính mới hay phương thức mới cho đối tượng mảng

- **Ví du:**

```
var myfriends=["John", "Bob", "Sue"];

document.write(myfirends.length);
```

Mảng (Array)

- Các phương thức của đối tượng mảng

Phương thức	Mô tả
concat(value1,...)	Nối các giá trị value1, value2... vào mảng hiện tại. Kết quả trả về một mảng mới chứa tất cả phần tử
every(testfunction [thisobj]) (JS1.6, FireFox)	Duyệt qua các phần tử của mảng, kiểm tra tất cả các phần tử có thoả điều kiện được hàm testfunction hay không ? Nếu thoả trả về true, ngược lại trả về false
filter(testfunction [thisobj]) (JS1.6, FireFox)	Lọc ra các phần tử thoả điều kiện được quy định trong hàm testfunction testfunction(element, index, array){....}
join([separator])	Chuyển tất cả các phần tử của mảng ra kiểu chuỗi và kết chúng lại thành một chuỗi dài chứa tất cả các phần tử

Mảng (Array)

- Các phương thức của đối tượng mảng

Phương thức	Mô tả
indexOf(target, [startIndex])	Tìm phần tử target trong mảng từ vị trí startIndex đến vị trí cuối mảng (JS1.6, FireFox)
lastIndexOf(target, [startIndex])	Tìm phần tử target trong mảng từ vị trí startIndex trở về vị trí đầu mảng (JS1.6, FireFox)
map(testfunction [thisobj])	Trả về một mảng mới với các phần tử được tính toán lại từ hàm testfunction (JS1.6, FireFox)
pop()	Trả về phần tử nằm cuối mảng và xoá phần tử đó ra khỏi mảng
push(value1, ...)	Thêm các phần tử vào cuối mảng, trả về số lượng phần tử của mảng sau khi thêm vào

Mảng (Array)

- Các phương thức của đối tượng mảng

Phương thức	Mô tả
reverse()	Đảo ngược các phần tử trong mảng
shift()	Xoá và trả về phần tử đầu tiên của mảng
slice(start, [end])	Trả về một mảng các phần tử từ vị trí start đến vị trí end của mảng, nếu không xác định ví end xem như lấy đến cuối mảng (vị trí end có thể là số âm)
splice(startIndex, [how_many], [value1, ...])	Xoá [how_many] phần tử tại vị trí startIndex và thay thế bằng các giá trị [value1, value2....] (chèn các giá trị mới vào vị trí đã xoá)
some(testfunction [thisobj]) (JS1.6, FireFox)	Duyệt qua các phần tử của mảng, nếu tồn tại bất kỳ phần tử nào thoả điều kiện trong hàm testfunction thì trả về giá trị true, ngược lại trả về giá trị false

Mảng (Array)

- Các phương thức của đối tượng mảng

Phương thức	Mô tả
sort([sortFunction])	Sắp xếp các phần tử của mảng tăng/giảm theo điều kiện của hàm sortFunction(a, b). Hàm sortFunction trả về 3 giá trị: < 0: sắp vị trí của a trước b = 0: không cần sắp xếp (a == b) > 0: sắp vị trí của b trước a
toString()	Tạo chuỗi biểu diễn mảng và các phần tử của nó
unshift(value1, ...)	Chèn phần tử vào vị trí đầu mảng, trả về số lượng phần tử của mảng sau khi chèn
valueOf()	Trả về các giá trị của mảng ở dạng chuỗi

Chuỗi (String)

- Các thuộc tính của đối tượng chuỗi

Thuộc tính	Mô tả
length	Trả về chiều dài của chuỗi (tổng số ký tự trong chuỗi)
prototype	Sử dụng thuộc tính này để thêm vào các thuộc tính mới hay phương thức mới cho đối tượng chuỗi

Chuỗi (String)

- **Các phương thức của đối tượng chuỗi**

Phương thức	Mô tả
anchor(name)	Trả về chuỗi với thẻ <a name="name"> bao quanh nó
big()	Trả về chuỗi với thẻ <big> bao quanh nó
blink()	Trả về chuỗi với thẻ <blink> bao quanh nó
bold()	Trả về chuỗi với thẻ <b> bao quanh nó
fixed()	Trả về chuỗi với thẻ <tt> bao quanh nó
fontcolor(color)	Trả về chuỗi với thẻ <font color="color"> bao quanh nó
fontsize(size)	Trả về chuỗi với thẻ <font size=" size "> bao quanh nó
italics()	Trả về chuỗi với thẻ <i> bao quanh nó
link(url)	Trả về chuỗi với thẻ <a href="url"> bao quanh nó
small()	Trả về chuỗi với thẻ <small> bao quanh nó

Chuỗi (String)

- **Các phương thức của đối tượng chuỗi**

Phương thức	Mô tả
strike()	Trả về chuỗi với thẻ <strike> bao quanh nó
sub()	Trả về chuỗi với thẻ <sub> bao quanh nó
sup()	Trả về chuỗi với thẻ <sup> bao quanh nó
charAt(x)	Trả về ký tự tại vị trí "x" của chuỗi
charCodeAt(x)	Trả về giá trị Unicode của ký tự tại vị trí "x" của chuỗi
concat(v1, v2,...)	Kết hợp một hay nhiều chuỗi (tham số v1, v2, ...) vào một chuỗi đã tồn tại và trả về chuỗi đã được kết hợp (không làm thay đổi chuỗi nguồn)
fromCharCode (c1, c2,...)	Trả về chuỗi được tạo bởi một chuỗi các giá trị Unicode (tham số c1, c2...). Đây là phương thức tĩnh của lớp String, do đó không cần phải tạo đối tượng cụ thể để sử dụng (vd: String.fromCharCode(....))

Chuỗi (String)

- **Các phương thức của đối tượng chuỗi**

Phương thức	Mô tả
indexOf(substr, [start])	Tìm chuỗi con substr bên trong chuỗi, nếu tìm thấy trả về ký tự được tìm hay chuỗi con của chuỗi, nếu không tìm thấy trả về giá trị -1
lastIndexOf(substr, [start])	[start] là tham số tuỳ chọn xác định vị trí bắt đầu tìm (mặc định là 0)
match(regexp)	Tương tự phương thức indexOf() nhưng tìm bắt đầu từ cuối chuỗi trở về đầu chuỗi ([start] mặc định là string.length-1)
replace(regexp, replacetext)	Tìm một giá trị xác định trong chuỗi. Trả về mảng các thông tin hay trả về giá trị null nếu không tìm thấy
search(regexp)	Tìm và thay thế một số ký tự trong chuỗi (nếu thoả biểu thức regular expression – biểu thức chính quy)

Chuỗi (String)

- **Các phương thức của đối tượng chuỗi**

Phương thức	Mô tả
slice(start, [end])	Trả về chuỗi con của chuỗi dựa trên tham số start và tham số end. (mặc định vị trí end là cuối chuỗi, giá trị của tham số end có thể là số âm)
split(delimiter, [limit])	Trả về mảng chứa các chuỗi con được phân cách bởi delimiter của chuỗi. Tham số [limit] là tham số tuỳ chọn, xác định số lượng thành phần tối đa được trả về
substr(start, [length])	Trả về chuỗi con của chuỗi bắt đầu tại vị trí start và có chiều dài là length. Tham số [length] là tuỳ chọn, nếu không xác định, mặc định lấy chiều dài từ start đến cuối chuỗi
substring(from, [to])	Trả về các ký tự của chuỗi giữa vị trí from và to. Nếu không xác định giá trị [to], mặc định lấy đến cuối chuỗi
toLowerCase()	Chuyển các ký tự trong chuỗi ra chữ thường
toUpperCase()	Chuyển các ký tự trong chuỗi ra chữ hoa

Lập trình Java Script

XỬ LÝ SỰ KIỆN

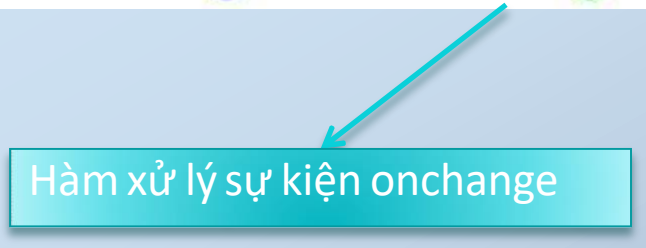
XỬ LÝ SỰ KIỆN

- Cú pháp:

```
<tagName  eventHandler = "JavaScript-Code or Function">
```

- VD: để kiểm tra khi có bất cứ sự thay đổi trên giá trị nhập liệu, ta có thể dùng sự kiện onchange() khai báo tới hàm xử lý.

```
<input type="text" name="age" onchange="checkage()">
```



XỬ LÝ SỰ KIỆN

- **Các sự kiện trong JS**

Thuộc tính	Sự kiện xảy ra khi
onabort	Tải một hình ảnh bị ngắt
onblur	Khi một thành phần bị mất focus
onchange	Người sử dụng thay đổi nội dung trên một field
onclick	Click chuột vào một đối tượng
ondblclick	Double click chuột vào một đối tượng
onerror	Một lỗi xảy ra khi tải một tài liệu hay hình ảnh
onfocus	Một thành phần có focus
onkeydown	Khi một phím trên bàn phím được nhấn xuống
onkeypress	Khi một phím trên bàn phím được nhấn
onkeyup	Khi một phím trên bàn phím được thả ra

XỬ LÝ SỰ KIỆN

- **Các sự kiện trong JS**

Thuộc tính	Sự kiện xảy ra khi
onload	Một trang hay hình ảnh tải hoàn tất
onmousedown	Khi nút chuột được nhấn
onmousemove	Khi chuột di chuyển
onmouseout	Khi chuột di chuyển ra ngoài một thành phần
onmouseover	Khi chuột di chuyển bên trên một thành phần
onmouseup	Khi nút chuộc được thả
onreset	Khi nút Reset được nhấn
onresize	Khi một cửa sổ hay frame thay đổi kích thước
onselect	Khi văn bản được chọn
onsubmit	Khi nhấn nút submit
onunload	Người sử dụng thoát khỏi trang web

XỬ LÝ SỰ KIỆN

- **Các sự kiện thường dùng của một số đối tượng**

Đối tượng	Các xử lý sự kiện của đối tượng
Selection list	onBlur, onChange, onFocus
Text	onBlur, onChange, onFocus, onSelect
Textarea	onBlur, onChange, onFocus, onSelect
Button	onClick
Checkbox	onClick
Radio button	onClick
Hypertext link	onClick, onMouseOver, onMouseOut
Clickable Imagemap area	onMouseOver, onMouseOut
Reset button	onClick

XỬ LÝ SỰ KIỆN

- **Các sự kiện thường dùng của một số đối tượng**

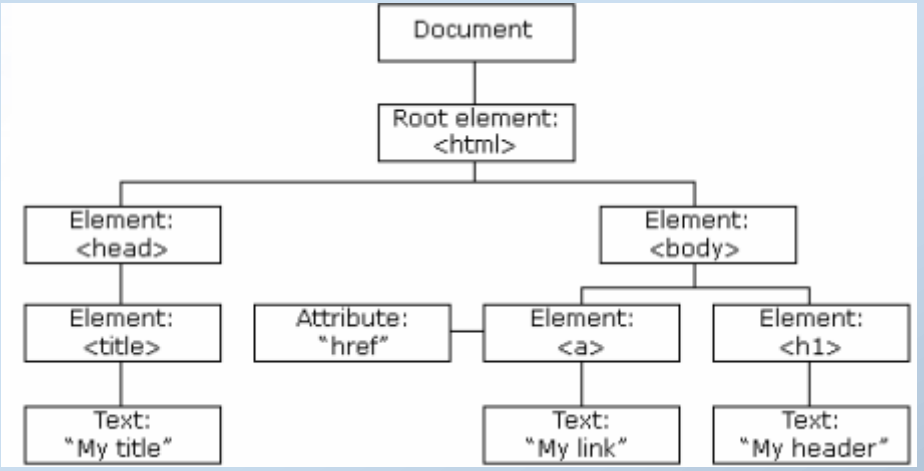
Đối tượng	Các xử lý sự kiện của đối tượng
Submit button	onClick
Document	onLoad, onUnload, onError
Window	onLoad, onUnload, onBlur, onFocus
Framesets	onBlur, onFocus
Form	onSubmit, onReset
Image	onLoad, onError, onAbort

Lập trình Java Script

HTML DOM & BOM

HTML DOM (Document Object Model)

- Khi một trang web được truy cập, trình duyệt sẽ tạo DOM của trang web đó.
- HTML DOM: mô hình đối tượng tài liệu HTML
  - Định nghĩa một chuẩn để truy cập và thao tác trên các tài liệu HTML
- DOM biểu diễn một tài liệu HTML bằng một cấu trúc cây (node tree), với các phần tử, thuộc tính và văn bản



HTML DOM (Document Object Model)

- Với mô hình DOM, JavaScript có tất cả sức mạnh cần thiết để tạo HTML động:
  - JavaScript có thể thay đổi tất cả các phần tử HTML trong trang
  - JavaScript có thể thay đổi tất cả các thuộc tính HTML trong trang
  - JavaScript có thể thay đổi tất cả các kiểu CSS trong trang
  - JavaScript có thể xóa các phần tử và thuộc tính HTML hiện có
  - JavaScript có thể thêm các phần tử và thuộc tính HTML mới
  - JavaScript có thể phản ứng với tất cả các sự kiện HTML hiện có trong trang
  - JavaScript có thể tạo các sự kiện HTML mới trong trang

HTML DOM - Methods

- Các phương thức HTML DOM là các hành động (action) bạn có thể thực hiện (trên Phần tử HTML).
- Thuộc tính HTML DOM là các giá trị (của Phần tử HTML) mà bạn có thể đặt hoặc thay đổi.
- Ví du:

```
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

getElementById: là phương thức DOM

innerHTML: là thuộc tính DOM

HTML DOM - Document

- DOM document là nút gốc của tất cả các elements trong trang web.
- Các phương thức để tìm kiếm HTML elements

Phương thức	Mô tả
document.getElementById(*id*)	Tìm theo element id
document.getElementsByTagName(*name*)	Tìm theo tag name
document.getElementsByClassName(*name*)	Tìm theo class name

HTML DOM - Document

- Thay đổi HTML element

Thuộc tính	Mô tả
element.innerHTML = *new html content*	Thay đổi thuộc tính HTML bên trong một element
element.attribute = new value	Thay đổi giá trị thuộc tính của một HTML element.
element.style.*property = new style*	Thay đổi style của một HTML element
Phương thức	Mô tả
element.setAttribute*(attribute, value)*	Thay đổi giá trị thuộc tính của một HTML element.

HTML DOM - Document

- Thêm hoặc bớt HTML element

Phương thức	Mô tả
document.createElement(*element*)	Tạo mới một HTML element
document.removeChild(*element*)	Xóa một HTML element
document.appendChild(*element*)	Thêm một HTML element
document.replaceChild(*new*, *old*)	Thay thế một HTML element
document.write(*text*)	Ghi vào HTML output stream

HTML DOM - Document

- Thêm sự kiện vào HTML element

Phương thức	Mô tả
document.getElementById(*id*).onclick = function(){*code*}	Thêm một đoạn code vào sự kiện click của một elmement

BOM (Browser Object Model)

- BOM cho phép javascript tương tác với trình duyệt thông qua các đối tượng sau:
  - **Window**: đối tượng ở mức cao nhất, có các thuộc tính thực hiện áp dụng trên toàn cửa sổ
  - **Navigator**: đối tượng lưu các thông tin về trình duyệt của client
  - **Screen**: đối tượng lưu các thông tin về màn hình client
  - **History**: đối tượng lưu các URL đã viếng thăm của cửa sổ trình duyệt
  - **Location**: đối tượng lưu thông tin về URL hiện hành
  - **Cookies**: cho phép lưu trữ thông tin người dung
  - **Popup**: cho phép hiển thị các hộp thoại thông báo cho người dung
  - **Timing Events**: cho phép chạy các hàm javascript theo thời gian chỉ định.

Lập trình Java Script

REGULAR EXPRESSION (REGEXP)

REGULAR EXPRESSION (REGEXP)

- Một Regular Expression có thể được định nghĩa với RegExp () constructor như sau:

  var pattern = /pattern/modifiers;

  Trong đó:

  - *pattern* : Một chuỗi mà xác định pattern của Regular Expression hoặc Regular Expression khác.
  - *modifiers*: Một chuỗi tùy chọn chứa bất kỳ giá trị sau:
    - g: Thực hiện đối sánh toàn cục (tìm tất cả các kết quả phù hợp thay vì dừng lại sau trận đấu đầu tiên)
    - i: không phân biệt hoa thường (case-insensitive),
    - m : thực hiện so khớp nhiều dòng (multiline matches)

REGULAR EXPRESSION- Patterns

- Dấu ngoặc vuông (bracket) được sử dụng để tìm phạm vi (range) các ký tự:

Expression	Miêu tả
[...]	Bất kỳ một ký tự nào trong dấu ngoặc vuông
[^...]	Bất kỳ một ký tự nào không trong dấu ngoặc vuông
[0-9]	Nó so khớp bất kỳ số thập phân nào từ 0 đến 9
[a-z]	Nó so khớp bất kỳ ký tự chữ thường nào từ a đến z.
[A-Z]	Nó so khớp bất kỳ ký tự chữ hoa nào từ A đến Z.
[a-Z]	Nó so khớp bất kỳ ký tự nào từ chữ thường a đến chữ hoa Z.

REGULAR EXPRESSION- Patterns

- **Phép lượng hóa**

Expression	Miêu tả
p+	Nó so khớp bất kỳ chuỗi nào chứa ít nhất một p.
p*	Nó so khớp bất kỳ chuỗi nào chứa 0 hoặc nhiều p.
p?	Nó so khớp bất kỳ chuỗi nào chứa 1 hoặc nhiều p
p{N}	Nó so khớp bất kỳ chuỗi nào chứa một dãy có {N} p
p{2,3}	Nó so khớp bất kỳ chuỗi nào chứa một dãy có 2 hoặc 3 p
p{2, }	Nó so khớp bất kỳ chuỗi nào chứa một dãy có ít nhất 2 p
p$	Nó so khớp bất kỳ chuỗi nào kết thúc với p
^p	Nó so khớp bất kỳ chuỗi nào bắt đầu bằng p

REGULAR EXPRESSION- Patterns

- **Metacharacter**

Ký tự	Miêu tả
.	Một ký tự đơn
\s	Một ký tự khoảng trống trắng (space, tab, dòng mới)
\S	Không phải ký tự khoảng trống trắng
\d	Một ký số (0-9)
\D	Không là ký số
\w	Một ký tự từ (a-z, A-Z, 0-9, _)
\W	Không là một ký tự từ
[\b]	Một literal backspace (trường hợp đặc biệt)

REGULAR EXPRESSION- Patterns

- **Các phương thức của RegExp**

Phương thức	Miêu tả
exec()	Thực thi một tìm kiếm cho một so khớp trong tham số chuỗi của nó.
test()	Kiểm tra một so khớp trong tham số chuỗi của nó.

Ví dụ : : hàm exec() tìm kiếm ký tự "e“ trong chuỗi

```
var str = "The best things in life are free"; var patt = new
RegExp("e");
var res = patt.exec(str);
```

Kết quả *res* là: **e**

jQuery

GIỚI THIỆU

GIỚI THIỆU JQUERY

- jQuery là thư viện được viết từ JavaScript, jQuery giúp xây dựng các chức năng bằng Javascript dễ dàng, nhanh và giàu tính năng hơn
- ***Tương thích đa nền tảng***: Nó tự động sửa lỗi và chạy được trên mọi trình duyệt phổ biến như Chrome, Firefox, Safari, MS Edge, IE, Android và iOS.
- ***Xử lý nhanh nhạy thao tác DOM***: **jQuery** giúp lựa chọn các phần tử DOM để traverse (duyệt) một cách dễ dàng, và chỉnh sửa nội dung của chúng bằng cách sử dụng Selector mã nguồn mở, mà còn được gọi là Sizzle.
- ***Đơn giản hóa việc tạo hiệu ứng***: Giống với code snippet có hiệu ứng animation, nó phủ các dòng code và bạn chỉ việc thêm biến/nội dung vào thôi.
- ***Hỗ trợ tốt phương thức sự kiện HTML***: Xử lý sự kiện − **jQuery** xử lý các sự kiện đa dạng mà không làm cho HTML code trở nên lộn xộn với các Event Handler

CÀI ĐẶT jQUERY

- Cách 1:
  - Tải jQuery từ jQuery.com.
  - Link thư viện jquery trong thẻ head

```
<head>
<script src="jquery-3.6.0.min.js"></script>
</head>
```

- Cách 2:
  - Link trực tiếp từ jQuery CDN

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
</head>
```

CÚ PHÁP CỦA JQUERY

- Cú pháp JQuery

  $(selector).action();

- Trong đó

  $: là ký hiệu bắt đầu sử dụng JQuery

  selector: là bộ lựa chọn, tương tự như CSS, dùng để lựa chọn đúng phần từ (element) mong muốn trong tài liệu HTML.

  action(): là các hàm yêu cầu đối tượng thực hiện hành động.

Ví du:

$(this).hide() - ẩn html element hiện tại.
$("p").hide() - ẩn tất cả thẻ <p>.
$(".test").hide() - ẩn tất cả thể có class="test".
$("#test").hide() - ẩn tất cả thẻ có id="test".

jQuery Selectors

- jQuery selector cho phép chọn và xử lý các phần tử HTML.
- jQuery selector được sử dụng để "tìm" (hoặc chọn) các phần tử HTML dựa trên tên, id, lớp, loại, thuộc tính, giá trị của thuộc tính.
- Tất cả các selector trong jQuery đều bắt đầu bằng ký hiệu $ và dấu ngoặc đơn: $ ().

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

jQuery Selectors – #id Selector

- #id selector sử dụng thuộc tính id của thẻ HTML để tìm phần tử cụ thể.
- Một id phải là duy nhất trong một trang, vì vậy chỉ nên sử dụng #id selector khi bạn muốn tìm một phần tử duy nhất.
- Để tìm một phần tử có id cụ thể, hãy viết một ký tự #, theo sau là id của phần tử HTML

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

jQuery Selectors – .class Selector

- .class selector tìm các phần tử với một lớp cụ thể.
- Để tìm các phần tử với một lớp cụ thể, viết một ký tự dấu chấm, theo sau là tên của lớp:

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

SỰ KIỆN TRONG jQuery

- Tất cả các hành động của người dùng khi tương tác với những thành phần của một trang web được gọi là sự kiện.
- Một sự kiện đại diện cho thời điểm chính xác khi điều gì đó xảy ra.

Sự kiện chuột	**Sự kiện bàn phím**	**Sự kiên form**	**Sự kiên Document/Window**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

SỰ KIỆN TRONG jQuery (tt)

- Cú pháp để thêm một sự kiện

  [selector].[hàm xử lý sự kiện]

```
$("p").click(function(){
  // action goes here!!
});
```

Thank you