Front-end Essentials

JavaScript

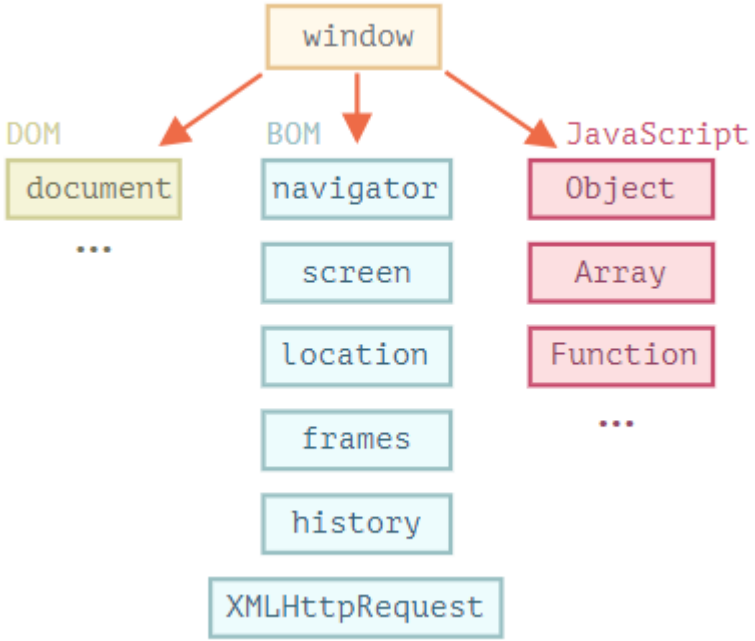
Event and DOM

Mục tiêu

- Hiểu và nắm được mục tiêu, ý nghĩa và ngữ cảnh sử dụng BOM, DOM để giải quyết vấn đề;
- Vận dụng các đối tượng BOM để giải quyết các vấn đề liên quan;
- Vận dụng các đối tượng DOM để giải quyết các vấn đề liên quan;
- Hiểu sự kiện (Event) trong JS và sử dụng, xử lý sự kiện với các phần tử DOM trong JS

Môi trường trình duyệt (Browser environment)

- JS ban đầu được xây dựng cho các trình duyệt Web;
- Nhưng sau đó nó được phát triển và trở thành một ngôn ngữ và đa nền tảng;
- Bức tranh toàn cảnh về Js chạy trên trình duyệt được cấu trúc hóa như hình bên cạnh:
  - ✓ Đối tượng gốc của trình duyệt là đối tượng window;
  - ✓ Đối tượng thứ cấp dưới Window là DOM, BOM và các đối tượng cơ sở khác (Object, Array, Function);



JavaScript - Window

Đối tượng Window (root)

- Đối tượng gốc (root) của trình duyệt là Window. Nó có 02 vai trò:
  - ✓ Nó là đối tượng toàn cục trong mã code JS
  - ✓ Nó đại diện cho cửa sổ trình duyệt và cung cấp các phương thức để kiểm soát trình duyệt;
  - ✓ Nó quản lý các cửa sổ trình duyệt và các tương tác giữa các cửa sổ trình duyệt;

```
<script>
    //Phương thức/Hàm toàn cục
    function sayHi() {
        alert("Hello");
    }
    //Đối tượng Window gọi hàm/phương thức toàn cục
    window.sayHi();
    //Đối tượng Window gọi phương thức có sẵn của nó
    alert(window.innerHeight); //Lấy độ cao của trình duyệt
</script>
```

Các phương thức của đối tượng Window

Phương thức	Mô tả	Phương thức	Mô tả
open()	Mở cửa sổ trình duyệt mới	setInterval()	Gọi một fuction hoặc đánh giá một biểu thức tại một khoảng thời gian xác định (tính bằng mili giây)
close()	Đóng cửa sổ trình duyệt hiện tại	setTimeout()	Gọi một fuction hoặc đánh giá một biểu thức sau một khoảng thời gian xác định (tính bằng mili giây)
moveTo()	Di chuyển cửa sổ đến vị trí cụ thể	prompt()	Hiển thị một hộp thoại yêu cầu người dùng nhập giá trị vào
resizeTo()	Thiết lập width và height của cửa sổ trình duyệt	scrollTo()	Cuộn cửa số xuống tọa độ xác định
alert()	Hiển thị hộp thoại với một thông điệp gì đó và nút OK	scrollBy()	Cuộn cửa sổ xuống theo giá trị pixel nào đó
confirm()	Hiển thị hộp thoại với một thông điệp gì đó và các nút OK, Cancel	focus()	Thiết lập trạng thái focus cho cửa sổ đang đứng
Tham khảo thêm: https://www.w3schools.com/jsref/obj_window.asp hoặc: http://www.javascriptkit.com/jsref/window.shtml			

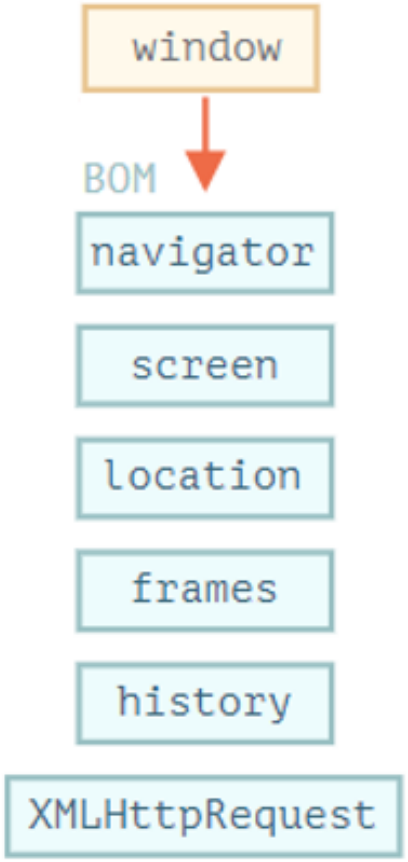
Các thuộc tính của đối tượng Window

Thuộc tính	Mô tả
closed	Có giá trị true khi cửa sổ bị đóng.
defaultStatus	Dùng để thiết lập dòng chữ mặc định trên thanh trạng thái của trình duyệt.
name	Tên của cửa sổ khi nó mới mở ra lần đầu.
opener	Tham chiếu đến cửa sổ tạo ra nó.
parent	Thường dùng để chỉ frame/window cha chứa/sinh ra frame/window hiện tại.
status	Thường dùng để thiết lập đoạn văn bản sẽ hiển thị trên thanh trạng thái khi người dùng di chuột qua một phần tử, ví dụ như đường liên kết.
top	Chỉ cửa sổ cha ở trên cùng.
Tham khảo thêm: https://www.w3schools.com/jsref/obj_window.asp	

JavaScript - BOM

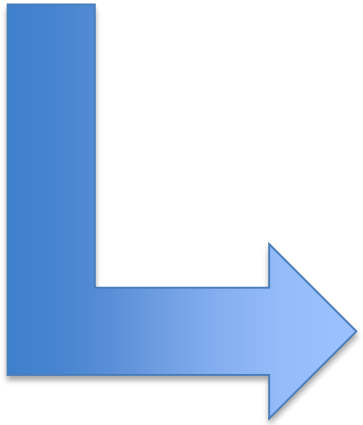
BOM (Mô hình đối tượng trình duyệt)

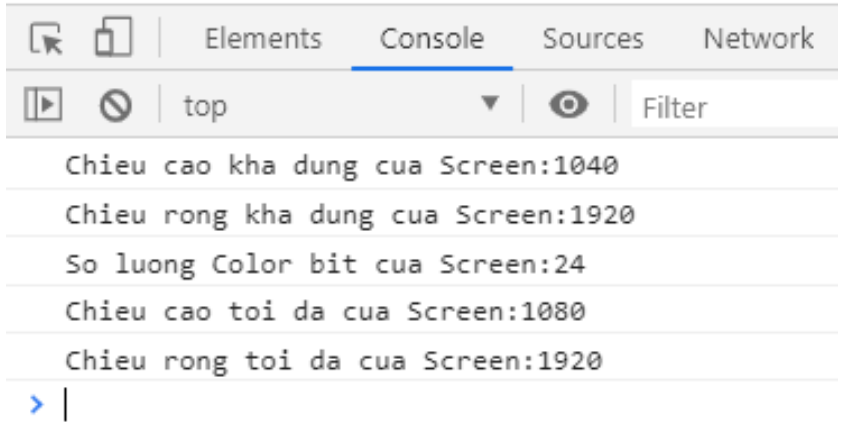
- BOM là lõi của JS trên trình duyệt;
- BOM cung cấp các đối tượng để chúng ta có được và khám phá trình duyệt mà người dùng đang dùng;
  - ✓ Location: Chứa các thông tin về trình duyệt đang được sử dụng bởi người dùng;
  - ✓ History: Lưu trữ tất cả nhưng trang Web mà người dùng đã truy cập/ghé thăm;
  - ✓ Navigator: Chứa thông tin về browser và hệ điều hành đang sử dụng browser đó;
  - ✓ Screen: chứa thông tin về màn hình hiển thị của máy người dùng hiện tại;



Window.screen

```
<script>
    //truy xuất thông tin Screen
    console.log("Chieu cao kha dung cua Screen:"+window.screen.availHeight);
    console.log("Chieu rong kha dung cua Screen:"+window.screen.availWidth)
    console.log("So luong Color bit cua Screen:"+window.screen.colorDepth);
    console.log("Chieu cao toi da cua Screen:"+window.screen.height);
    console.log("Chieu rong toi da cua Screen:"+window.screen.width);
</script>
```

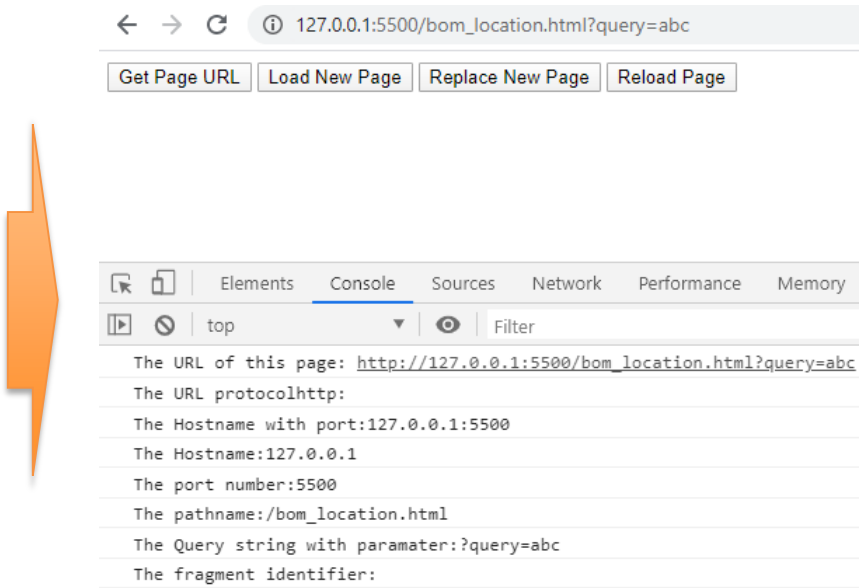




Window.location

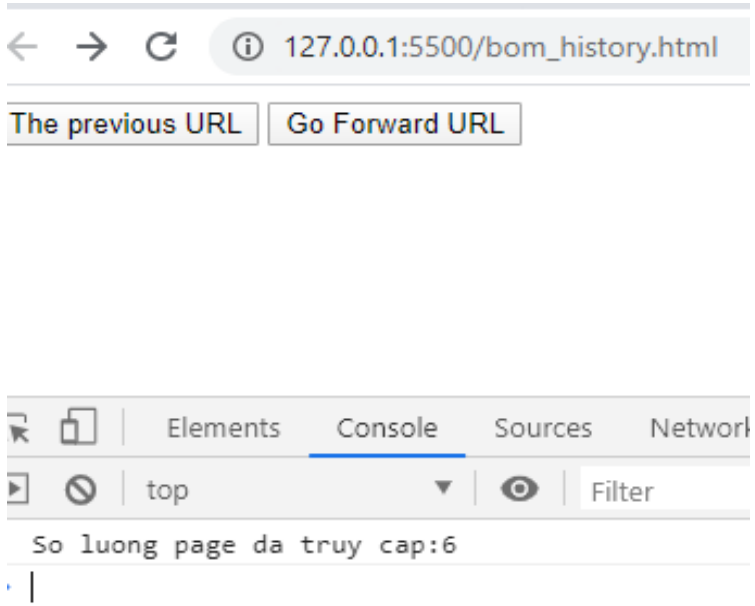
- Thướng sử dụng để điều hướng và xử lý URL trong Javascript

```
<button type="button" onclick="getURL();">Get Page URL</button>
<button type="button" onclick="loadNewPage();">Load New Page</button>
<button type="button" onclick="replaceNewPage();">Replace New Page</button>
<button type="button" onclick="forceReload();">Reload Page</button>
<script>
     function getURL() {
         console.log("The URL of this page: "+window.location.href);
        console.log("The URL protocol"+window.location.protocol);
         console.log("The Hostname with port:"+window.location.host);
        console.log("The Hostname:"+window.location.hostname);
        console.log("The port number:"+window.location.port);
         console.log("The pathname:"+window.location.pathname);
        console.log("The Query string with paramater:"+window.location.search);
        console.log("The fragment identifier:"+window.location.hash);
    }
    function loadNewPage(){
        window.location.assign("https://www.w3schools.com/js/js_window_location.asp");
    }
    function replaceNewPage(){
        window.location.replace("https://developer.mozilla.org/en-US/docs/Web/API/Window/location");
    }
    function forceReload() {
       window.location.reload(true);
    }
</script>
```



Window.history

- Khám phá lịch sử truy cập mà người dùng đã sử dụng trình duyệt




```
<button type="button" onclick="getBack();">The previous URL</button>
<button type="button" onclick="goForward();">Go Forward URL</button>
<script>
     console.log("So luong page da truy cap:"+window.history.length);
     function getBack(){
          window.history.back();
     }
     function goForward(){
          window.history.forward();
     }
</script>
```

Window.navigator

```
<script>
      console.log("User Agent:"+window.navigator.userAgent);
        console.log("User vendor:"+window.navigator.vendor);
        console.log("User Language:"+window.navigator.language);
        console.log("App Name: " + window.navigator.appName);
        console.log("App Version: " + window.navigator.appVersion);
        console.log("App Code Name: " + window.navigator.appCodeName);
        console.log("Platform: " + window.navigator.platform);
        console.log("Java Enabled or Not:"+window.navigator.javaEnabled());
        console.log("Cookies Are Enabled or Not:"+window.navigator.cookieEnabled);
        console.log("The Battery Status:");
        navigator.getBattery().then(function(battery) {
            battery.onchargingchange = chargingChange();
            function chargingChange() {
                console.log("IsCharging",  battery.charging);
                console.log("Percentage",battery.level);
            }
        });
        console.log("The Plugins installed:" + window.navigator.plugins.length);
        for(var i = 0; i < window.navigator.plugins.length; i++) {
            console.log(navigator.plugins[i].name + ' - ' +navigator.plugins[i].filename + ' - ' +  navigator.plugins[i].description);
        }
</script>
```

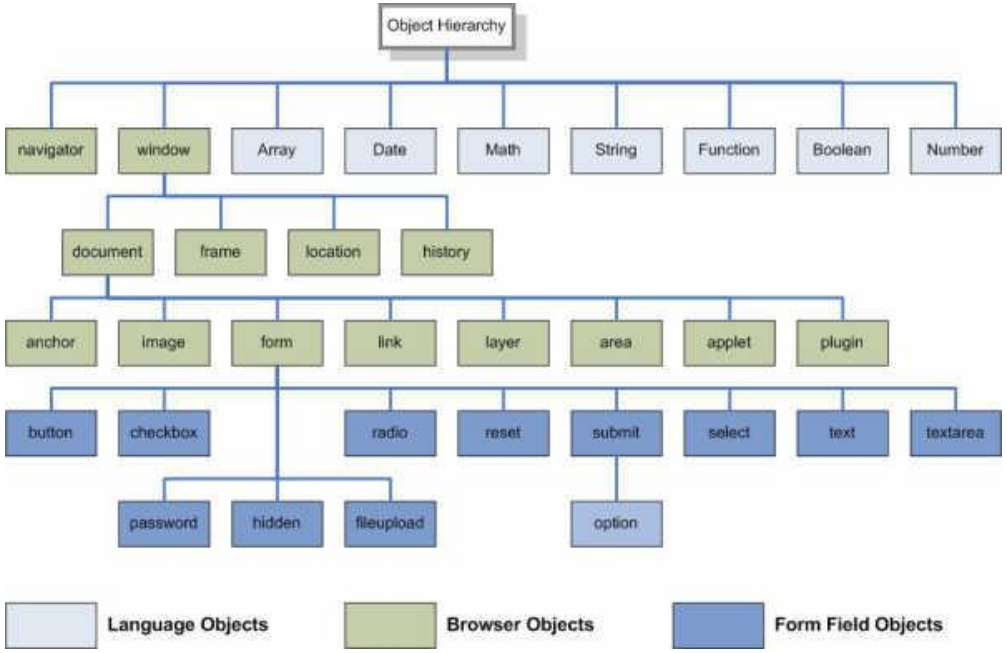
```
User Agent:Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36
User vendor:Google Inc.
User Language:en-US
App Name: Netscape
App Version: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36
App Code Name: Mozilla
Platform: Win32
Java Enabled or Not:false
Cookies Are Enabled or Not:true
The Battery Status:
The Plugins installed:3
Chrome PDF Plugin - internal-pdf-viewer - Portable Document Format
Chrome PDF Viewer - mhjfbmdgcfjbbpaeojofohoefgiehjai -
Native Client - internal-nacl-plugin -
IsCharging true
Percentage 1
charging Time 0
DisCharging Time Infinity
```

Tham khảo thêm: https://www.w3schools.com/js/js_window_navigator.asp

JavaScript - DOM

HTML DOM

- DOM = (**D**ocument **O**bject **M**odel
- DOM – Mô hình Đối tượng Tài liệu
- Khi trang Web được tải lên, trình duyệt sẽ tạo ra một mô hình đối tượng tài liệu của trang.
- Mô hình DOM HTML được xây dựng như một cây của đối tượng gọi là DOM Tree



HTML DOM

- HTML DOM là một mô hình đối tượng cho HTML, nó xác định:
  - ✓ Phần tử HTML như là một đối tượng
  - ✓ Thuộc tính cho tất cả các phần tử HTML
  - ✓ Phương thức cho tất cả các phần tử HTML
  - ✓ Sự kiện cho tất cả các phần tử HTML

- HTML DOM là một API cho JavaScript:
  - ✓ Có thể thêm/thay đổi/xóa các phần tử HTML
  - ✓ Có thể thêm/thay đổi/xóa thuộc tính HTML
  - ✓ Có thể thêm/thay đổi/xóa định dạng CSS
  - ✓ Có thể phản ứng với các sự kiện HTML
  - ✓ Có thể thêm/thay đổi/xóa sự kiện HTML

Tìm các phần tử HTML

- Các cách để tìm phần tử trong HTML
  - ✓ Tìm phần tử HTML theo id
    document.getElementById(id);
  - ✓ Tìm các phần tử HTML theo tên thẻ
    document.getElementsByTagName(tagname);
  - ✓ Tìm các phần tử HTML theo tên lớp
    document.getElementsByClassName(classname);
  - ✓ Tìm các phần tử HTML bằng bộ chọn CSS
    document.querySelectorAll(htmlselector);
  - ✓ Tìm phần tử HTML thông qua tập hợp các đối tượng HTML:
    anchors, forms, images, links và scripts

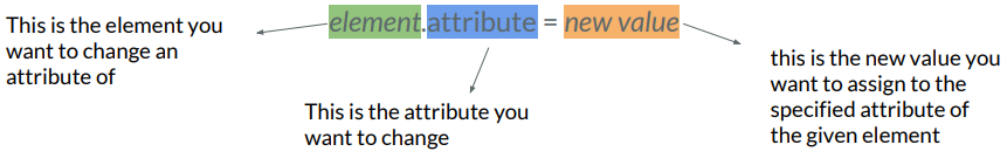
Thay đổi phần tử HTML

Thuộc tính	Mô tả
element.innerHTML = “giá trị mới cho phần tử”	Thay đổi giá trị của phân tử HTML
element.attribute = “giá trị thuộc tính mới”	Thay đổi giá trị thuộc tính của phần tử HTML
element.style.*property = “Thuộc tính CSS mới”*	Thay đổi giá trị CSS của một phần tử HTML
Phương thức	**Mô tả**
element.setAttribute*(attribute, value)*	Thay đổi giá trị thuộc tính của phần tử HTML

Code ví dụ - Thay đổi giá trị thuộc tính

HTML

```
<body>
    <h2>Wellcome to HTML DOM</h2>
    <p id="intro">Lorem ipsum dolor sit amet co
nsectetur adipisicing elit. Veritatis error quo
 nam expedita aspernatur, deleniti, doloremque
alias architecto maiores reprehenderit animi ex
plicabo sit ex? Excepturi quam quisquam maiores
 aperiam voluptates.</p>
    <h2>Result</h2>
    <p class="result"></p>
    <img id="ima" src="images/color-
fruit.jpeg" alt="" width="200" height="150">
</body>
```



HTML + JS

```
<script>
    //Lay gia tri tu phan tu chua id
    var intro = document.getElementById("intro");
    //Gan gia tri cho phan tu chua class
    document.getElementsByClassName("result")[0].innerHT
ML = "<b>Update values for Element By Class Name from El
ement Id:</b>"+intro.innerHTML;
    //Thay doi gia tri thuoc tinh src cho phan tu image
    document.getElementById('ima').src = "images/pom-
laptop.png";
    //hoac su sung phuong thuc
    var img = document.getElementById('ima');
    img.setAttribute('src',"images/pom-laptop.png");
</script>
```

Ví dụ

Trước

127.0.0.1:5500/dom.html

Wellcome to HTML DOM

Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis error quo nam expedita aspernatur, deleniti, doloremque alias architecto maiores reprehenderit animi explicabo sit ex? Excepturi quam quisquam maiores aperiam voluptates.

Result

Sau

127.0.0.1:5500/dom.html

Wellcome to HTML DOM

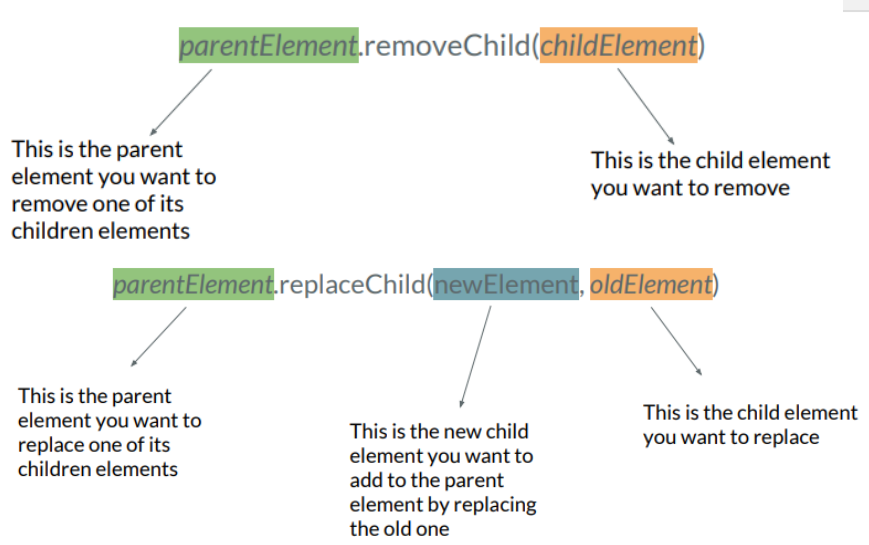
Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis error quo nam expedita aspernatur, deleniti, doloremque alias architecto maiores reprehenderit animi explicabo sit ex? Excepturi quam quisquam maiores aperiam voluptates.

Result

Update values for Element By Class Name from Element Id: Lorem ipsum dolor sit amet consectetur adipisicing elit. Veritatis error quo nam expedita aspernatur, deleniti, doloremque alias architecto maiores reprehenderit animi explicabo sit ex? Excepturi quam quisquam maiores aperiam voluptates.

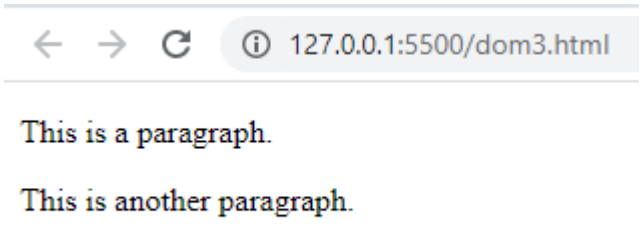
Thêm, thay thế và xóa các phần tử

Phương thức	Mô tả
document.createElement(*element*)	Tạo mới một phần tử
document.removeChild(*element*)	Xóa một phần tử
document.appendChild(*element*)	Thêm một phần tử con
document.replaceChild(*new, old*)	Thay thể một phần tử
document.write(*text*)	Xuất nội dung ra tài liệu HTML

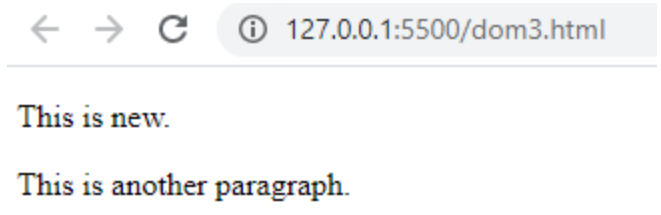


Ví dụ (Thêm và thay thế)

Kết quả ban đầu



Kết quả sau khi thêm mới và thay thế nội dung cho phần tử




```
<div id="div1">
   <p id="p1">This is a paragraph.</p>
   <p id="p2">This is another paragraph.</p>
</div>
```

```
<script>
   var parent = document.getElementById("div1");
   var child = document.getElementById("p1");
   var para = document.createElement("p");
   var node = document.createTextNode("This is new.");
   para.appendChild(node);
   parent.replaceChild(para,child);
</script>
```

Ví dụ: Tìm phần tử HTML

- Sử dụng phương thức querySelectorAll('selector') sẽ trả về danh sách các phần tử HTML khớp với CSS Query;

const pars = document.querySelectorAll("p.main");

Ví dụ (Xóa phần tử con)


```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
```

```
<script>
   var parent = document.getElementById("div1");
   var child = document.getElementById("p1");
   parent.removeChild(child);
</script>
```

Thêm phần tử mới (linh hoạt)


```
 <form enctype="multipart/form-
data" action="" method="post">
    <p>Upload file(s)</p>
    <div id="files">
        <p id="file-0">
<input type="file" name="uploaded_file[]" />
</p>
    </div>
    <input type="button" value="Add File" oncl
ick="addFile();" />
</form>
```

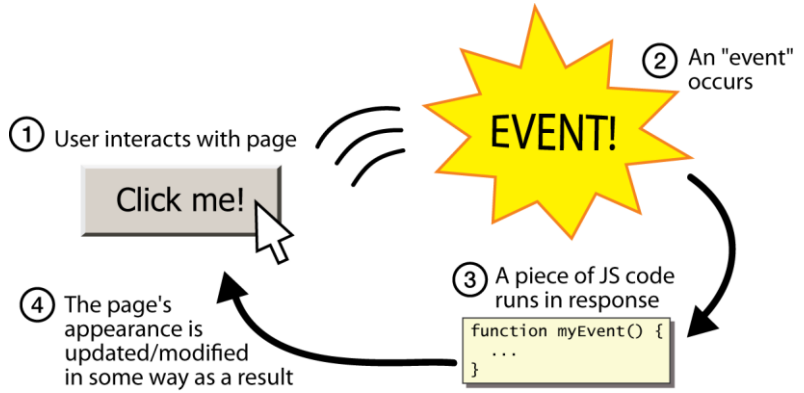
```
//Hàm Add một phần tử con vào phần tử cha
function addElement(parentId, elementTag, elementId, html) {
      var p = document.getElementById(parentId);
      var newElement = document.createElement(elementTag);
      newElement.setAttribute('id', elementId);
      newElement.innerHTML = html;
      p.appendChild(newElement);
}
//Hàm Xóa phần tử con (Dựa vào Id phần tử con)
function removeElement(elementId) {
      var element = document.getElementById(elementId);
      element.parentNode.removeChild(element);
}
var fileId = 0;
//Add File - Hàm lắng nghe sự kiện
function addFile() {
      fileId++; // Tăng ID của phần tử lên một cách tự động để đảm báo tính duy nhấ
t của ID cho phần tử mới
      var elementId = 'file-'+fileId;
      var removeLink = '<a href="" onclick="removeElement'+elementId+');">Remove Fi
le</a>';
      var html = '<input type="file" name="uploaded_files[]"/>'+removeLink;
      console.log(elementId);
      addElement('files', 'p', elementId, html);
}
```

JavaScript

Sự kiện - Event

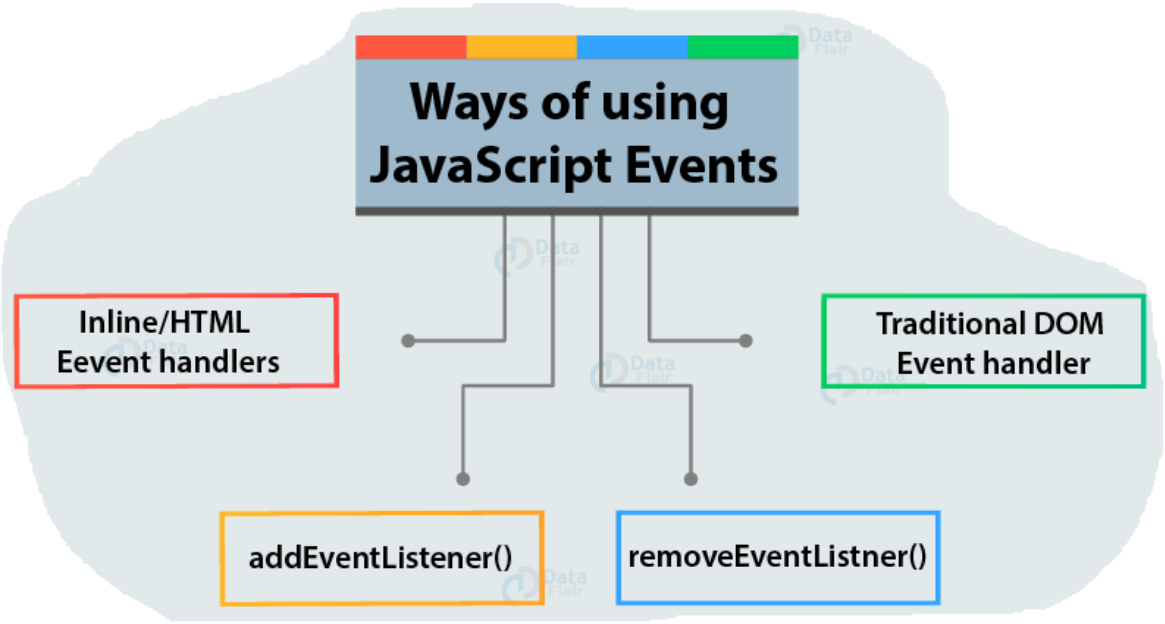
JS Event

- Sự kiện là "Những gì" được thực hiện hoặc có hành động nào đó xẩy ra trên trình duyệt;
- Một số sự kiện trên trang:
  - ✓ Trang web được tải xong
  - ✓ Nhấn vào một Nút nào đó, một phần tử nào đó
  - ✓ Các phần tử form thay đổi giá trị
  - ✓ ….



Các cách lắng nghe sự kiện trong JS

- Lắng nghe trong phần tử HTML (Inline/HTML Event handlers)
- Lắng nghe sự kiện sử dụng hàm addEventListener()
- Bỏ lắng nghe sự kiện với hàm removeEventListeners()



Inline/HTML Event handlers

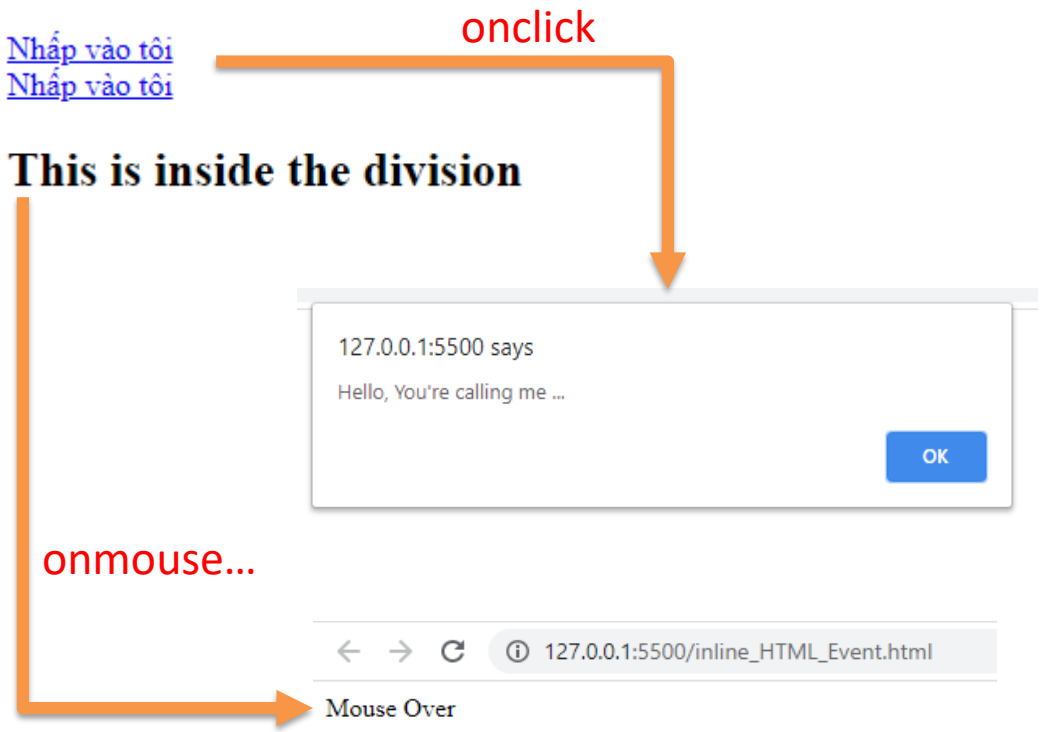
- Sự kiện được kích hoạt thông qua liên kết giữa phần tử HTML bằng tên thuộc tính của nó
- Tên sự kiện bắt đầu bằng tiền tố on
- Nhưng không phải tất cả các phần tử đều có thể gọi các kiểu sự kiện này được. Ví dụ: Onchange thường sử dụng cho input Text Field
  - ✓ onchange: Một phần tử HTML đã được thay đổi
  - ✓ onclick: Người dùng nhấp vào một phần tử HTML
  - ✓ onmouseover: Người dùng di chuột qua phần tử HTML
  - ✓ onmouseout: Người dùng di chuyển chuột ra khỏi phần tử HTML
  - ✓ onkeydown: Người dùng nhấn phím trên bàn phím
  - ✓ onload: Trình duyệt đã tải xong trang

<element attribute = "functionName()">

Inline/HTML Event handlers – Ví dụ

```
<a id="alink" href="javascript:doSomething(this);"> Nhấp vào tôi </a>
<a id = "alink" href = "" onclick = "doSomething(this);"> Nhấp vào tôi </a>
<div onmouseover = "over()" onmouseout = "out()">
     <h2> This is inside the division </h2>
</div>
```

Bad practice




```
<script>
        function doSomething(){
             alert("Hello, You're calling me ...");
        }

        function over() {
             document.write ("Mouse Over");
        }
        function out() {
             document.write ("Mouse Out");
        }
</script>
```

addEventListener() and removeEventListener()

- Thêm sự kiện cho phần tử thông qua addEventListene()

  element.**addEventListener**(“event”, functionName [,Boolean]);

- Xóa sự kiện của phần tử

  element.**removeEventListener**(“event”, functionName [,Boolean]);

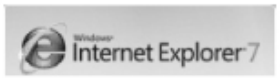
```
<button>Click me</button>
<p></p>

<script type="text/javascript">
     var btn = document.querySelector("button");
     function changeColor(){
            btn.style.backgroundColor = "blue"; //change background color
            btn.style.color = "white"; //change font color
            document.querySelector("p").innerHTML = "Great! The button changed its color." //add text
     }
     btn.addEventListener("click", changeColor); //adds event listener
     // btn.removeEventListener(“click, changeColor”); //To remove the event listener
</script>
```

Internet Explorer

```
currentBtn.attachEvent("onmouseover", showHint);
```

Use attachEvent() for Internet Explorer browsers.




```
currentBtn.addEventListener("mouseover", showHint, false);
```

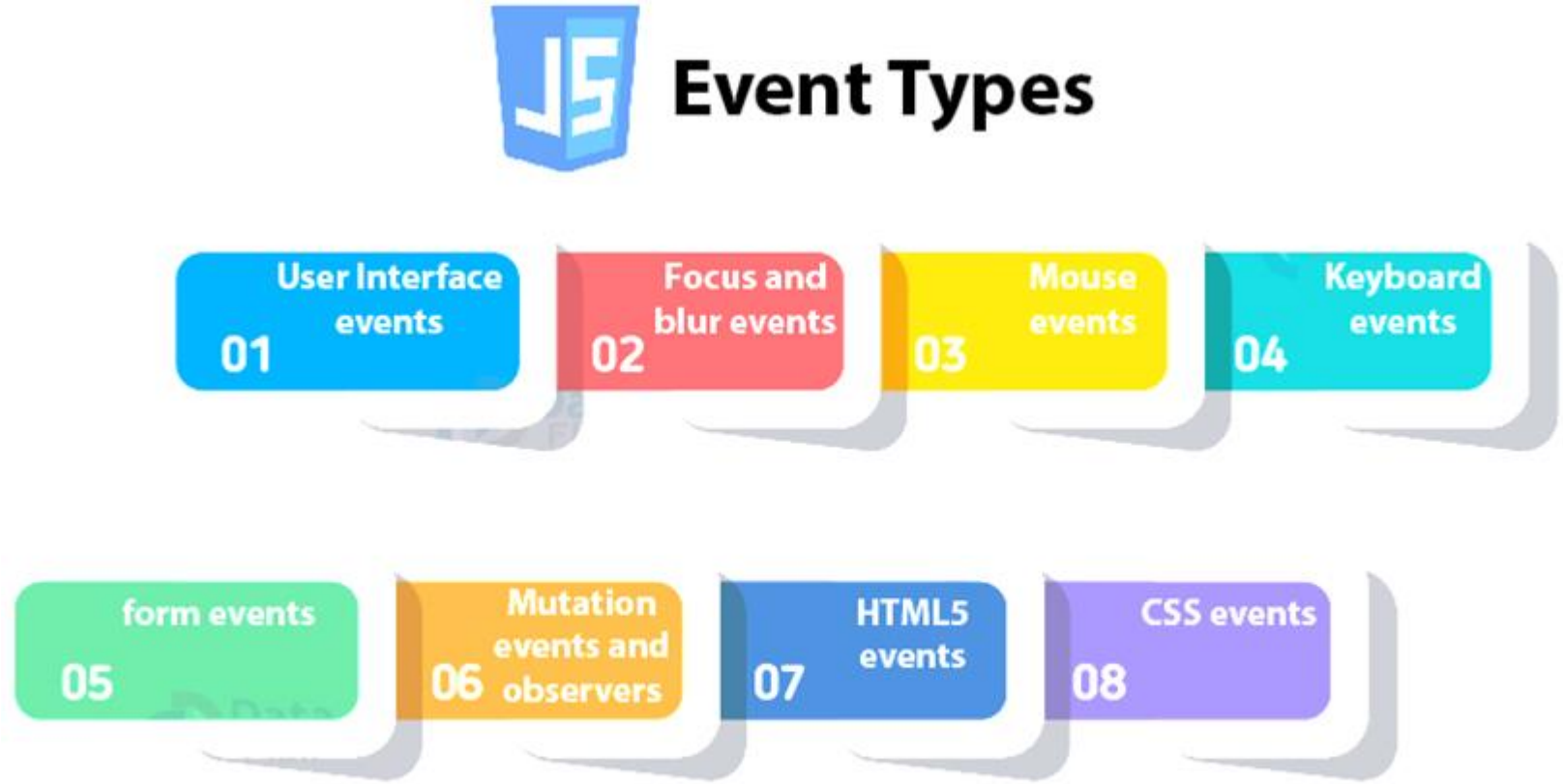
Use addEventListener() for Firefox...

...Opera...

...as well as Safari and most other modern browsers.

These are all DOM Level 2 browsers.

Các kiểu Sự kiện trong JS



User Interface events

- Sự kiện này xẩy ra khi có bất kỳ sự tương tác nào lên cửa sổ trình duyệt hơn là lên trang HTML;
- Sự kiện này chúng ta thường lắng nghe trên đối tượng Window, không phải trên đối tượng tài liệu (DOM);
- Một số sự kiện phổ biến:
  - ✓ Load → Đang tải trang Web
  - ✓ Unload → Trang Web chưa tải xong
  - ✓ Error → Có lỗi JavaScript xẩy ra trên trang
  - ✓ Resize → Resize lại kích thước cửa sổ trình duyệt
  - ✓ Scroll → Khi người dùng kéo thanh trượt lên/xuống trên cửa sổ trình duyệt.

Focus and blur events

- Sự kiện này sẽ được kích hoạt khi người dùng focus vào phần tử HTML;
- Loại sự kiện này thường hữu ích khi sử dụng trên các Form với các công việc:
  - ✓ Hiển thị tips hoặc feedback cho người dùng khi người dùng tương tác vào phần tử Form;
  - ✓ Kiểm tra tính hợp lệ của dữ liệu trên form khi người dùng di chuyển ra khỏi một phần tử HTML mà chưa cần nhấn nút Submit;
- Một số hàm sự kiện:
  - ✓ Focus → xẩy ra trên một nút DOM nào đó mà người dùng đang "ở trong" phần tử đó (focus)
  - ✓ Blur → xẩy ra trên một nút DOM khi người dùng không còn focus vào phần tử
  - ✓ Focusin → tương tự như sự kiện focus. Nhưng Firefox không hỗ trợ sự kiện focusin
  - ✓ Focusout → tương tự như sự kiện blur. Nhưng Firefox không hỗ trợ sự kiện này

Mouse Event

- onclick — The event occurs when the user clicks on an element
- oncontextmenu — User right-clicks on an element to open a context menu
- ondblclick — The user double-clicks on an element
- onmousedown — User presses a mouse button over an element
- onmouseenter — The pointer moves onto an element
- onmouseleave — Pointer moves out of an element
- onmousemove — The pointer is moving while it is over an element
- onmouseover — When the pointer is moved onto an element or one of its children
- onmouseout — User moves the mouse pointer out of an element or one of its children
- onmouseup — The user releases a mouse button while over an element

Window/Frame Event

- onabort — The loading of a media is aborted
- onbeforeunload — Event occurs before the document is about to be unloaded
- onerror — An error occurs while loading an external file
- onhashchange — There have been changes to the anchor part of a URL
- onload — When an object has loaded
- onpagehide — The user navigates away from a webpage
- onpageshow — When the user navigates to a webpage
- onresize — The document view is resized
- onscroll — An element's scrollbar is being scrolled
- onunload — Event occurs when a page has unloaded

Form Event

- onblur — When an element loses focus
- onchange — The content of a form element changes

(for <input>, <select> and <textarea>)

- onfocus — An element gets focus
- onfocusin — When an element is about to get focus
- onfocusout — The element is about to lose focus
- oninput — User input on an element
- oninvalid — An element is invalid
- onreset — A form is reset
- onsearch — The user writes something in a search field (for <input="search">)
- onselect — The user selects some text (for <input> and <textarea>)
- onsubmit — A form is submitted

Event handlers for Form Elements.

Table : Event handlers for Form Elements.

Object	Event Handler
button	*onClick, onBlur, onFocus*
checkbox	*onClick, onBlur, onFocus.*
FileUpLoad	*onClick, onBlur, onFocus*
hidden	*none*
password	*onBlur, onFocus, onSelect.*
radio	*onClick, onBlur, onFocus*
reset	*onReset.*
select	*onFocus, onBlur, onChange.*
submit	*onSubmit*
text	*onClick, onBlur, onFocus , onChange*
textarea	*onClick, onBlur, onFocus , onChange*

Drag/Drop Event

- ondrag — An element is dragged
- ondragend — The user has finished dragging the element
- ondragenter — The dragged element enters a drop target
- ondragleave — A dragged element leaves the drop target
- ondragover — The dragged element is on top of the drop target
- ondragstart — User starts to drag an element
- ondrop — Dragged element is dropped on the drop targe

Media Event

- onabort — Media loading is aborted
- oncanplay — The browser can start playing media (e.g. a file has buffered enough)
- oncanplaythrough — The browser can play through media without stopping
- ondurationchange — The duration of the media changes
- onended — The media has reached its end
- onerror — Happens when an error occurs while loading an external file
- onloadeddata — Media data is loaded
- onloadedmetadata — Metadata (like dimensions and duration) are loaded
- onloadstart — The browser starts looking for specified media
- onpause — Media is paused either by the user or automatically
- onplay — The media has been started or is no longer paused
- onplaying — Media is playing after having been paused or stopped for buffering
- onprogress — The browser is in the process of downloading the media

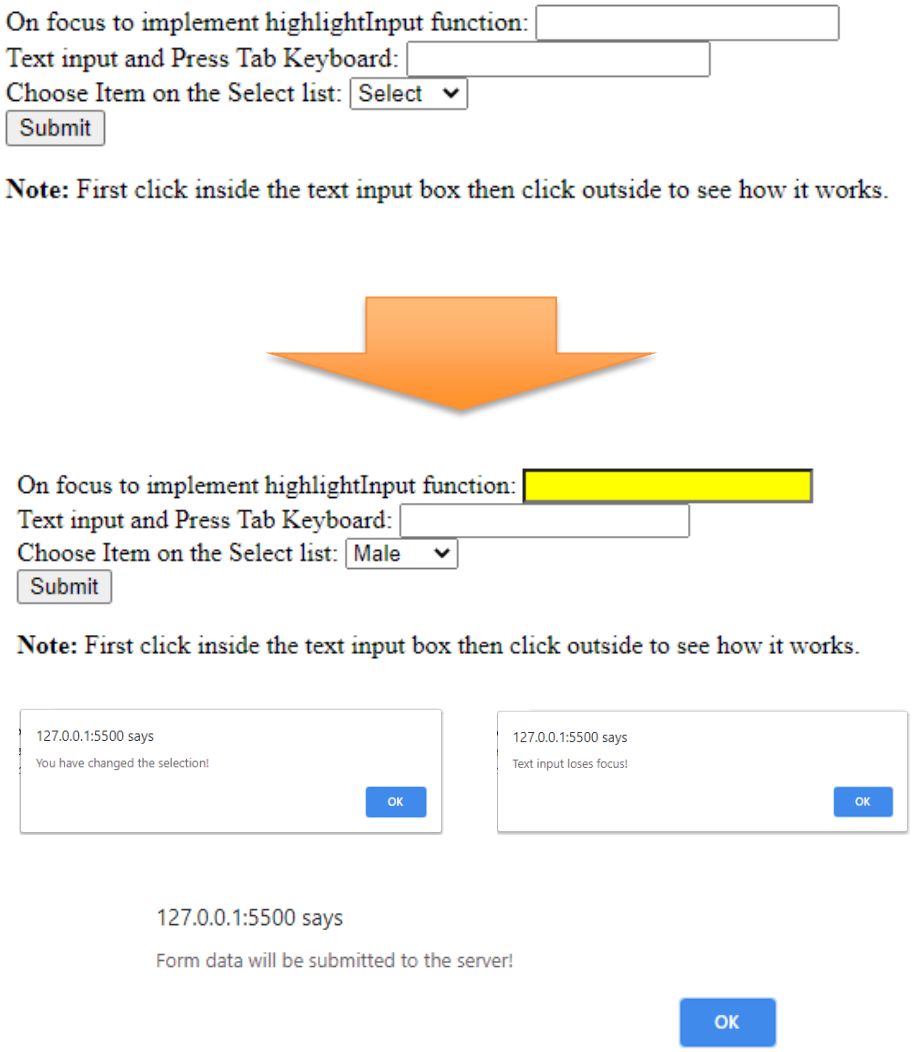
Media Event (tt)

- onratechange — The playing speed of the media changes
- onseeked — User is finished moving/skipping to a new position in the media
- onseeking — The user starts moving/skipping
- onstalled — The browser is trying to load the media but it is not available
- onsuspend — The browser is intentionally not loading media
- ontimeupdate — The playing position has changed (e.g. because of fast forward)
- onvolumechange — Media volume has changed (including mute)
- onwaiting — Media paused but expected to resume (for example, buffering)

Ví dụ:

```
<form onsubmit="alert('Form data will be submitted to the server!');">
        <label>On focus to implement highlightInput function:</label>
        <input type="text" onfocus="highlightInput(this)">
        <br>
        <label>Text input and Press Tab Keyboard:</label>
        <input type="text" onblur="alert('Text input loses focus!')">

        <br>
        <label>Choose Item on the Select list:</label>
        <select onchange="alert('You have changed the selection!');">
            <option>Select</option>
            <option>Male</option>
            <option>Female</option>
        </select>
        <br>
        <input type="submit" value="Submit">
        <p><strong>Note:</strong> First click inside the text input box
then click outside to see how it works.</p>
</form>
<script>
     function highlightInput(elm){
         elm.style.background = "yellow";
     }
</script>
```



Mouse Event – ví dụ

```
<body onmousemove  = "show_coords(event)">
        <p>Moving in the document to get the
x (horizontal) and y (vertical) coordinates o
f the mouse pointer </p>
        <p id="demo"></p>
</body>
```

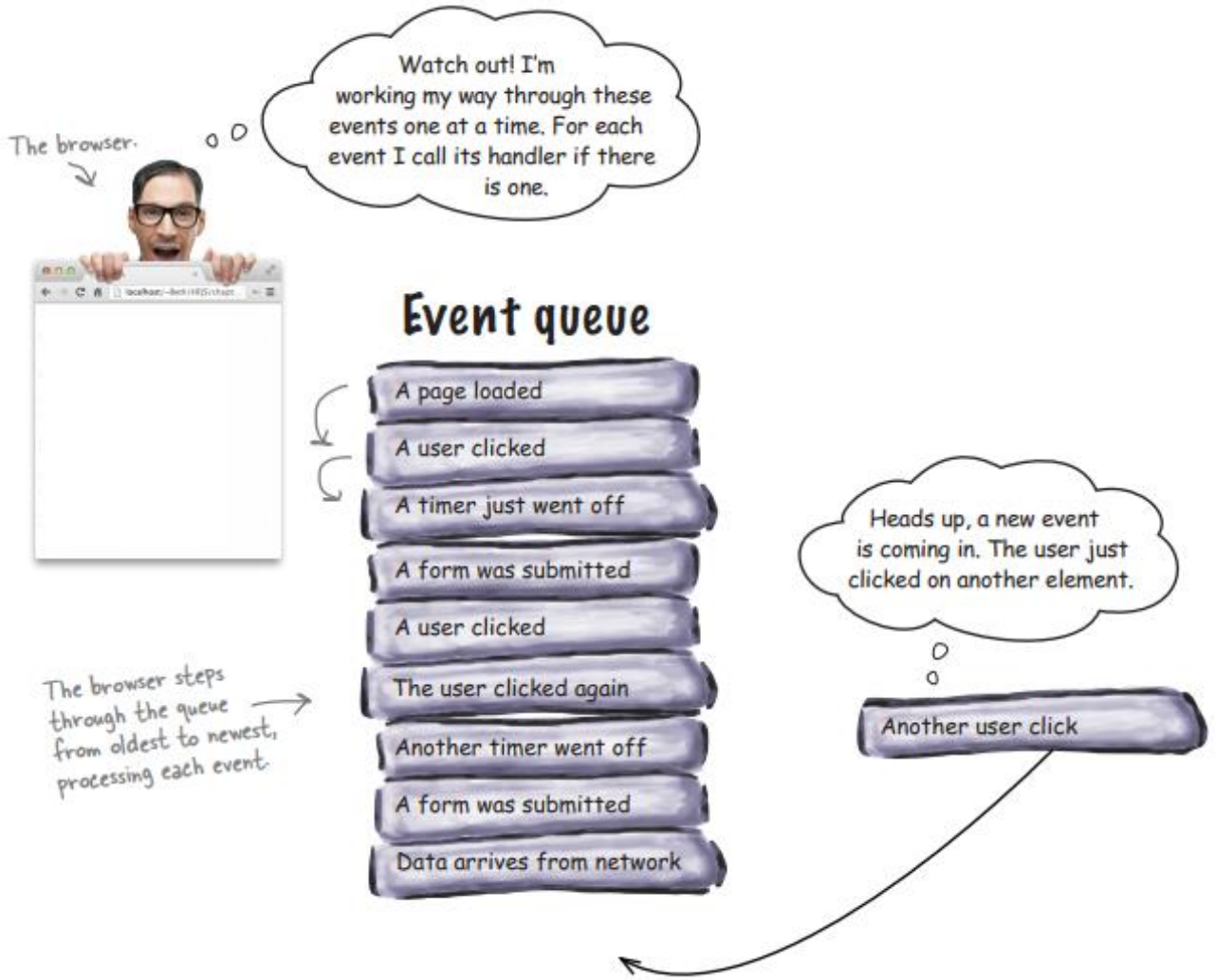
```
<script type="text/javascript">
    function show_coords(event) {
        var x = event.clientX;
        var y = event.clientY;
        var coords = "X coords: " + x + ", Y c
oords: " + y;
        document.getElementById("demo").innerH
TML = coords;
    }
</script>
```

Moving in the document to get the x (horizontal) and y (vertical) coordinates of the mouse pointer

Moving in the document to get the x (horizontal) and y (vertical) coordinates of the mouse pointer

X coords: 60, Y coords: 32

Events and queues



Time-based events

- with time-based events, rather than assigning a handler to a property, you call a function, setTimeout, instead and pass it your handler

First we write an event handler. This is the handler that will be called when the time event has occurred.

```
function timerHandler() {
    alert("Hey what are you doing just sitting there staring at a blank screen?");
}
```

All we're doing in this event handler is showing an alert.

```
setTimeout(timerHandler, 5000);
```

And here, we call setTimeout, which takes two arguments: the event handler and a time duration (in milliseconds).

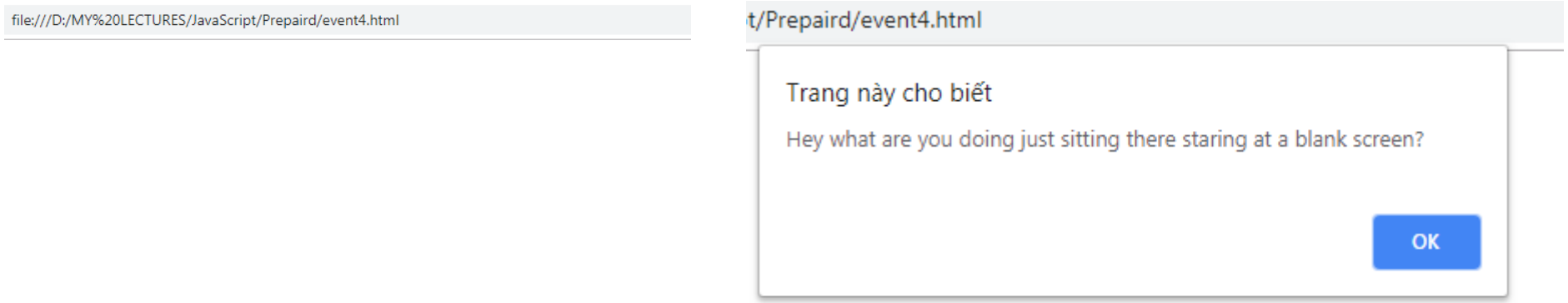
Using setTimeout is a bit like setting a stop watch.

Here we're asking the timer to wait 5000 milliseconds (5 seconds).

And then call the handler timerHandler.

Time-based events

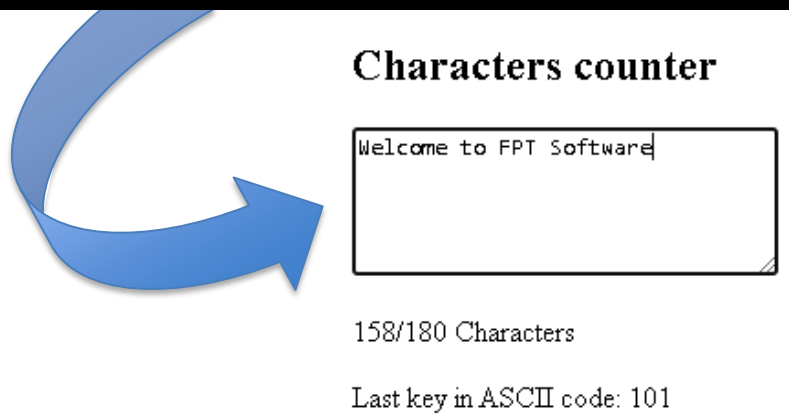
```
<script>
    window.onload = init;
    function init(){
        setTimeout(TimerHandler,3000);
    }
    function TimerHandler() {
        alert("Hey what are you doing just sitting there staring at a blank screen?");
    }
</script>
```



EVENT HANDLERS

- Ứng dụng đơn giản: Đếm ký tự và mã ký tự nhập vào Textarea thông qua việc bắt sự kiện bàn phím

```
<div class="counter-ascii">
    <h2>Characters counter</h2>
    <form action="">
        <textarea name="count" id="message" cols="30" rows="5"></textarea>
        <p id="charactersLeft"></p>
        <p id="lastKey"></p>
    </form>
</div>
```



Suggestion Code

```
<script>
        var el;
        function charCount(e){
            var textEntered, charDisplay, counter, lastKey;
            textEntered = document.getElementById('message').value;
            charDisplay = document.getElementById('charactersLeft');
            counter = (180-(textEntered.length));
            charDisplay.textContent = counter+'/180 Characters';

            lastKey = document.getElementById('lastKey');
            lastKey.textContent = 'Last key in ASCII code: ' + e.keyCode;
        }

        el = document.getElementById('message');
        el.addEventListener('keypress',charCount,false);
</script>
```

Section 3

JavaScript Regular Expressions

RegExp – What is Regular Expression?

- A regular expression is a sequence of characters that forms a **search pattern**.
- The search pattern can be used for **text search** and **text replace** operations.
- **Syntax:**

  /pattern/modifiers;

RegExp – What is Regular Expression?

- **Using String Methods:**

Method	Description
search()	The search() method uses an expression to search for a match, and returns the position of the match.
replace()	The replace() method returns a modified string where the pattern is replaced.

RegExp – search() and replace()

- **search**() method:

```
var str = "Visit MySchools";
var n = str.search(/myschools/i);
// The result in n will be: 6
```

- **replace**() method:

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/i, "MySchools");
// The result in res will be: Visit MySchools!
```

RegExp – Modifiers

- **Regular Expression Modifiers:**

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

RegExp – Syntax (1)

- **Brackets** are used to find a range of characters:

Expression	Description
[a-z]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

RegExp – Syntax (2)

- **Metacharacters** are characters with a special meaning:

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning or at the end of a word
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

RegExp – Syntax (3)

- **Quantifiers** define quantities:

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n

RegExp – Using RegExp Object (1)

- Using **test**() method:
  - ✓ The test() method is a RegExp expression method.
  - ✓ It searches a string for a pattern, and returns true or false, depending on the result.
- **Example 2:**

```
var patt = /in/;
patt.test("The best things in life are free!");
// the output of the code above will be: true
```

- **Example 2:**

```
// allow letters, numbers, and underscores
var illegalChars = /\W/; // Equivalent to [^A-Za-z0-9_].
illegalChars.test("dieunt1");
// the output of the code above will be: true
```

RegExp – Using RegExp Object (2)

- The exec() method is a RegExp expression method.
  - ✓ It searches a string for a specified pattern, and returns the found text.
  - ✓ If no match is found, it returns *null.*
- **Example:**

```
var patt = /in/;
patt.exec("The best things in life are free!");
// the output of the code above will be: in
```

RegExp – Summary

- Regular Expression is a powerful tool for **text search** and **text replace**
- Using **RegExp**, you can search a string in another string or if a string matches a pattern

Thank you