TechMaster
Học là có việc

DOM Basic p2

Trần Huy Hoàng
Email: huyhoang.tran6669@gmail.com
Phone: 0788.719.666

Nội dung

1. HTML Events
2. Xử lý các events
3. Lab 1: Thực hành xử lý sự kiện
4. Mouse events
5. Keyboard events
6. Lab 2: Thực hành xử lý sự kiện chuột và bàn phím
7. Lập trình đồng bộ && bất đồng bộ
8. Callback Function
9. Promise trong js
10. Call AJAX
11. Lab 3: Weather web
12. Homeworks

1. HTML Events

Events là các hành động xảy ra khi người dùng tương tác với trang web.

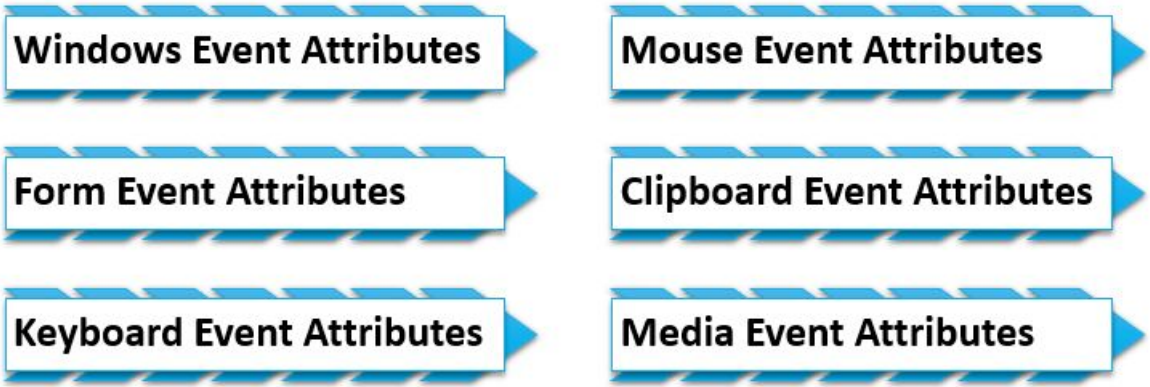
Ví dụ:

- Khi người dùng click chuột vào 1 phần tử
- Khi nhập dữ liệu vào ô input
- Khi submit form gửi dữ liệu lên server
- …

Xem thêm: Links

1. HTML Events

Các loại sự kiện HTML



2. Xử lý các events

Để xử lý sự kiện, chúng ta có thể thực thi một hàm trong trường hợp xảy ra sự kiện

Chúng ta có 1 số cách sau để xử lý sự kiện trong Javascript DOM

✅ Sử dụng HTML-attribute

✅ Sử dụng DOM property

✅ Sử dụng addEventListener

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .red {
            color: red;
        }
    </style>
</head>
<body>
    <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Ea quod tempore ut molestias odio assumenda quasi sapiente natus adipisci,

    <!-- Sử dụng HTML-attribute -->
    <button onclick="sayHello()">Click me</button>

    <script>
        function sayHello() {
            console.log('Xin chào các bạn');
        }

        let btn = document.querySelector('button');
        let para = document.querySelector('p');

        // Sử dụng DOM property
        btn.onclick = function() {
            console.log('Xin chào các bạn 1');
        }

        // Sử dụng addEventListener
        btn.addEventListener('click', function() {
            para.classList.toggle('red');
        })
    </script>
</body>
</html>
```

```
<!-- Sử dụng HTML-attribute -->
<button onclick="sayHello()">Click me</button>
```

```
// Sử dụng DOM property
btn.onclick = function() {
    console.log('Xin chào các bạn 1');
}

// Sử dụng addEventListener
btn.addEventListener('click', function() {
    para.classList.toggle('red');
})
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .red {
            color: red;
        }
    </style>
</head>
<body>
    <p>Lorem ipsum, dolor sit amet consectetur adipisicing elit. Ea quod tempore ut molestias odio assumenda quasi sapiente natus adipisci,

    <!-- Sử dụng HTML-attribute -->
    <button onclick="sayHello()">Click me</button>

    <script>
        function sayHello() {
            console.log('Xin chào các bạn');
        }

        let btn = document.querySelector('button');
        let para = document.querySelector('p');

        // Sử dụng DOM property
        btn.onclick = function() {
            console.log('Xin chào các bạn 1');
        }

        // Sử dụng addEventListener
        btn.addEventListener('click', function() {
            para.classList.toggle('red');
        })
    </script>
</body>
</html>
```

3. Lab 1: Thực hành xử lý sự kiện

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="text">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>

    <button id="btn-1">Change content</button>
    <button id="btn-2">Change color</button>
    <button id="btn-3">Change background</button>
</body>
</html>
```

3. Lab 1: Thực hành xử lý sự kiện

Yêu cầu:

Lắng nghe **sự kiện click** ở 3 nút button theo 3 cách sau :

- Button "**Change content**" sử dụng "**HTML-attribute**"
- Button "**Change color**" sử dụng "**DOM property**"
- Button "**Change background**" sử dụng "**addEventListener**"

Thực hiện các chức năng sau:

Yêu cầu 1:

- Khi nhấn vào button "**Change content**" hiển thị một nội dung quote bất kỳ

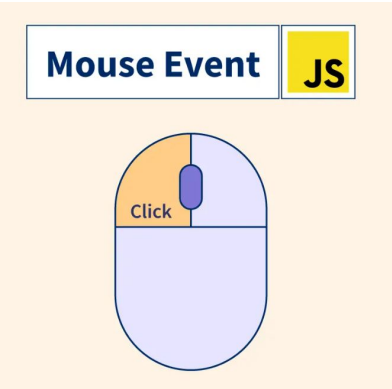
Yêu cầu 2:

- Khi nhấn vào button "**Change color**" thì thay đổi màu của thẻ p (sử dụng màu HEX - cần viết 1 function để random màu HEX)

Yêu cầu 3:

- Khi nhấn vào button "**Change background**" thì thay đổi màu background-color của thẻ p (sử dụng màu RGB - cần viết 1 function để random màu RGB)

4. Mouse events



Các sự kiện liên quan đến chuột

1. **click :** Sự kiện xảy ra khi người dùng click vào một phần tử
2. **mousedown :** Sự kiện xảy ra khi người dùng nhấp chuột
3. **mousemove :** Sự kiện xảy ra khi người dùng di chuyển chuột
4. **mouseup :** Sự kiện xảy ra khi người dùng thả chuột

4. Mouse events

Các ví dụ về add các sự kiện mouse events

```
document.addEventListener('click', function() {
 console.log('click');
})

document.addEventListener('mousedown', function() {
 console.log('mousedown');
})

document.addEventListener('mousemove', function() {
 console.log('mousemove');
})

document.addEventListener('mouseup', function() {
 console.log('mouseup');
})
```

4. Mouse events

Sử dụng tham số **event** trong sự kiện mouse click

```
document.addEventListener('click', function(event) {
 console.log(event);
})
```

Các giá trị cần quan tâm trong biến event của hàm xử lý sự kiện

- **screen X, screen Y :** Tọa độ của con trỏ chuột theo màn hình máy tính của người dùng
- **client X, client Y :** Tọa độ của con trỏ chuột theo cửa sổ trình duyệt
- **offset X, offset Y :** Tọa độ của con trỏ chuột theo đối tượng đang kích hoạt sự kiện

5. Keyboard events

Các sự kiện liên quan đến bàn phím

1. **keydown:** được gọi khi bạn nhấn xuống một key
2. **keyup:** được gọi khi bạn nhả key đó ra
3. **keypress:** được gọi khi bạn nhấn và giữ key

5. Keyboard events

```
// Lắng nghe sự kiện keydown
document.addEventListener("keydown", function () {
    console.log("keydown");
});

// Lắng nghe sự kiện keyup
document.addEventListener("keyup", function () {
    console.log("keyup");
});

// Lắng nghe sự kiện keypress
document.addEventListener("keypress", function () {
    console.log("keypress");
});
```

5. Keyboard events

Sử dụng tham số **event** trong sự kiện keyboard

```
// Lắng nghe sự kiện keydown
document.addEventListener("keydown", function (event) {
    console.log(event);
});
```

Các giá trị cần quan tâm trong biến event của hàm xử lý sự kiện

- ✅ **key :** Tên phím được nhấn
- ✅ **keyCode :** Mã unicode của phím được nhấn

6. Lab 2: Thực hành xử lý sự kiện chuột và bàn phím

Yêu cầu:

- Nhấn chuột vào một vị trí bất kỳ trên trang web, tạo 1 hình tròn tại vị trí đó.
- Nếu nhấn phím “B” sẽ vẽ 1 hình chữ nhật vào một vị trí bất kỳ trên trang web

6. Lab 2: Thực hành xử lý sự kiện chuột và bàn phím

Lab2.1

Yêu cầu:

- Nhấn chuột vào một vị trí bất kỳ trên trang web, tạo 1 hình tròn tại vị trí đó.
- Nếu nhấn phím “B” sẽ vẽ 1 hình chữ nhật vào một vị trí bất kỳ trên trang web

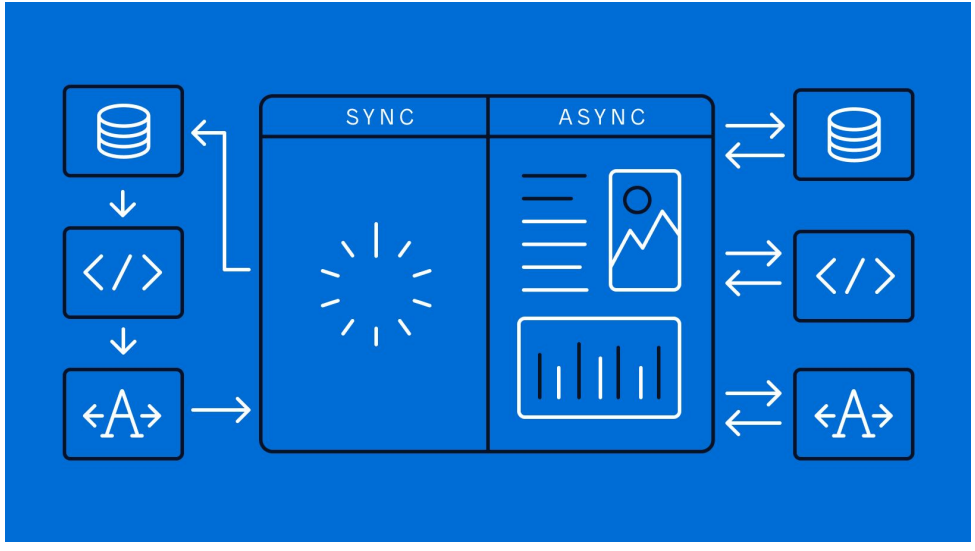
Lab2.2

Yêu cầu:

- Khi người dùng nhập password và bấm nút “Show”, lúc này hiển thị password và nội dung button chuyển từ “**Show**” --> “**Hide**”
- Ngược lại khi người dùng bấm vào nút “Hide”, lúc này ẩn password và nội dung button chuyển từ “**Hide**” --> “**Show**”

7. Lập trình đồng bộ && bất đồng bộ

- **Lập trình đồng bộ** là các công việc được thực hiện một cách tuần tự.
- **Lập trình bất đồng bộ** là các công việc có thể chạy cùng một lúc mà không cần công việc trước kết thúc rồi mới thực hiện.



8. Callback Function

Callback là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong.

```
function doTask1(name, callback) {
    console.log("Bắt đầu công việc");
    console.log(`Thực hiện công việc ${name}`);
    setTimeout(function () {
        callback();
    }, 3000);
}

function doTask2(name, callback) {
    console.log(`Thực hiện công việc ${name}`);
    setTimeout(function () {
        callback();
    }, 4000);
}
```

9. Promise

Một object **Promise** đại diện cho một giá trị ở thời điểm hiện tại có thể chưa tồn tại, nhưng sẽ được xử lý và có giá trị vào một thời gian nào đó trong tương lai.

Ví dụ: nếu bạn sử dụng Promise cho việc gọi API để lấy dữ liệu, bạn sẽ tạo ra một đối tượng Promise đại diện cho data lấy được từ API. Điều cốt lõi ở đây là dữ liệu sẽ chưa tồn tại ở thời điểm đối tượng Promise được tạo ra, mà chỉ có thể truy cập sau khi có response từ web service. Trong thời gian chờ lấy dữ liệu, Promise object sẽ đóng vai trò như một proxy cho dữ liệu. Hơn nữa, bạn có thể đính các callback vào Promise object để thực hiện việc xử lý dữ liệu. Các callback này sẽ chỉ thực hiện khi dữ liệu đã sẵn sàng.

9. Promise

```
// HỨA : Nếu có trên 30 triệu, sẽ mua iphone 13 pro max
let money = 35

// Hứa
let buyIphone = new Promise(function (resolve, reject) {
    if (money > 30) {
        resolve("Mua iphone thôi") // Lời hứa được thực hiện thành công
    } else {
        reject("Kiếm thêm tiền đi") // Lời hứa được thực hiện thất bại
    }
})
```

9. Promise

```
// Hứa tiếp
// Nếu mua iphone xong, còn thừa tiền, nếu đủ thì mua con airpod (4 củ)

let buyAirpod = new Promise(function (resolve, reject) {
    if (money - 30 - 4 >= 0) {
        resolve("Mua thêm airpod")
    } else {
        reject("Không đủ tiền mua airpod")
    }
})
```

9. Promise

```
// Thực hiện lời hứa
buyIphone
    .then(res => { // res là nội dung trong resolve
        console.log(res);
        return buyAirpod;
    })
    .then(res1 => {
        console.log(res1);
    })
    .catch(error => { // error là nội dung trong reject
        console.log(error)
    })
    .finally(() => { // Luôn được thực hiện kể cả thành công hay thất bại
        console.log("Đi về nhà");
    })
```

9. Promise

Callback là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong.

9. Promise

Callback là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong.

9. Promise

Callback là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong.

The End