Đây là tài liệu nhóm, không chia sẻ ra dưới bất kỳ hình thức nào

1.Xử lí HTML

<div *class*="container">

    <ul *class*="list-items">

        <!-- mặc định là tất cả li.list-item đều ẩn đi, chỉ khi nào có class show thì mới hiện ra  -->

        <!-- khi thêm class show thì sẽ có hiệu ứng xuất hiện -->

        <!-- khi có class hidden thì có hiệu ứng biến mất -->

        <!-- ảnh để trực tiếp trong thẻ li bằng background-image -->

        <li *class*="list-item show"></li>

        <li *class*="list-item"></li>

        <li *class*="list-item"></li>

        <li *class*="list-item"></li>

        <li *class*="list-item"></li>

    </ul>

</div>

<!-- hai nút để nhấn tiến - lùi slide -->

<div *class*="btn-group">

    <i *class*="fas fa-angle-left"></i>

    <i *class*="fas fa-angle-right"></i>

</div>

2.Xử lí css

2.1. Add background cho các thẻ li, đảm bảo rằng các ảnh khi hiển thị luôn phải căn giữa màn hình. Đảm bảo thứ tự của các ảnh đúng theo dự định bằng thuộc tính z-index. Tất nhiên để có z-index thì phần tử bố (ul) phải có position: absolute;

ul*.list-items* li*.list-item* {

    min-height: 100vh;

    min-width: 80%;

    position: absolute;

    opacity: 0;

    visibility: hidden;

    background-size: cover;

}

li*.list-item:nth-child*(even){

    background-image: url('./image/canhdong.jpg');

    z-index: 5;

}

li*.list-item:nth-child*(odd){

    background-image: url('./image/canhdong.jpg');

    z-index: 5;

}

li*.list-item:nth-child*(1){

    background-image: url('./image/canhdong.jpg');

    z-index: 5;

}

li*.list-item:nth-child*(2){

    background-image: url('./image/cotden.jpg');

    z-index: 4;

}

li*.list-item:nth-child*(3){

    background-image: url('./image/dunglai.jpg');

    z-index: 3;

}

li*.list-item:nth-child*(4){

    background-image: url('./image/guongchieuduong.jpg');

    z-index: 2;

}

li*.list-item:nth-child*(5){

    background-image: url('./image/langque.jpg');

    z-index: 1;

}

2.2.Viết 4 class chứa hiệu ứng là ‘show’, ‘show1’ ,‘hidden’,’hidden2’. Trong đó:

show và hidden -> hiệu ứng cho nút trái (tức là nút lùi slide)

show1 và hidden1 -> hiệu ứng cho nút phải (tức là nút tiến slide) .

Trong đó, phải viết rõ là .list-item.show chứ ko viết riêng show -> đảm bảo là chỉ khi có đủ 2 class là list-item và show thì mới nhận hiệu ứng. Các phần khác tương tự

*.list-item.show*{

    animation: toLeft 1s forwards 1;

}

@-webkit-keyframes *toLeft* {

    from {

        transform: translateX(-100%); opacity: 0;visibility: hidden;

    }

    to {

        transform: translateX(0); opacity: 1; visibility: visible;

    }

}

*.list-item.hidden*{

    animation: toLeftHidden 1s forwards 1;

}

@-webkit-keyframes *toLeftHidden* {

    from {

        transform: translateX(0); opacity: 1; visibility: visible;

    }

    to {

        transform: translateX(100%); opacity: 0; visibility: hidden;

    }

}

2.3. Về xử lí nút chuyển slide, ko nên để z-index -> tức là ko cho nút đè lên slide. Việc này sẽ gián tiếp tạo ra 1 hiệu ứng là khi slide chưa hoàn thành chuyển động thì nút sẽ không bấm được ( vì khi chưa chuyển động xong thì slide vẫn nhận visibility: visible; nên nút sẽ bị slide đè lên trên ).

Còn nếu không làm như thế thì phải chặn người dùng click liên tiếp bằng js (webkit-animation-end)

3.Xử lí Javascript

3.1.Lấy các DOM cần dùng và console.log để kiểm tra

    const LEFT = document.querySelector('i.fas.fa-angle-left');

    console.log(LEFT);

    const RIGHT = document.querySelector('i.fas.fa-angle-right');

    console.log(RIGHT);

    const SLIDE = document.querySelectorAll('ul.list-items li.list-item');

    console.log(SLIDE);

3.2.Tạo biến để lưu trữ giá trị của slide

    // chỉ số của slide đang hiển thị

    var presentIndex = 0;

    // chỉ số của slide phía trước

    var previousIndex;

    // chỉ số của slide phía sau

    var nextIndex;

3.3.Xử lí khi người dùng ấn nút tiến - lùi slide

LEFT.addEventListener('click', () => {

    // dùng vòng lặp để xóa class show, show1 tại slide hiện tại, sau đó thêm class

hidden

    // phải xóa cả show1 vì có thể người dùng nhấn trái hoặc phải chứ ko nhấn mãi về 1 phía

    // tương tự với các vòng lặp xóa class khác

    for (let i = 0; i < SLIDE.length - 1 ; i++) {

            SLIDE[presentIndex].classList.remove('show');

            SLIDE[presentIndex].classList.remove('show1');

            SLIDE[presentIndex].classList.add('hidden');

    }

    // tính chỉ số của phần tử phía trước

    // Nếu slide đang ở vị trí 0, bấm lùi sẽ phải trả về chỉ số của slide cuối cùng

(slide xoay vòng)

    // nếu chưa ở vị trí 0, thì vẫn lùi bình thường

    previousIndex = presentIndex - 1;

    presentIndex = previousIndex < 0 ?  SLIDE.length - 1  :  previousIndex;

    console.log("Chỉ số slide hiện tại là : " ,presentIndex);

    // dùng vòng for để xóa class hidden đã thêm phía trên

    // nếu không xóa class hidden đi thì chỉ slide chỉ hiện được 1 lần

    // bởi từ lần thứ 2 các thẻ sẽ có cả 2 class show và hidden thì ko nhận hiệu ứng

được

    for (let y = 0; y < SLIDE.length - 1; y++){

        SLIDE[presentIndex].classList.remove('hidden');

        SLIDE[presentIndex].classList.remove('hidden1');

    }

    // phần tử tiếp theo hiển thị ra thêm class show vào

    SLIDE[presentIndex].classList.add('show');

});

// nút phải tương tự, chỉ có điều sửa chỗ tính toán phần tử tiếp theo

RIGHT.addEventListener('click', () => {

    for (let i = 0; i < SLIDE.length - 1 ; i++) {

        for (let y = 0; y < SLIDE.length - 1; y++){

            SLIDE[presentIndex].classList.remove('show');

            SLIDE[presentIndex].classList.remove('show1');

            SLIDE[presentIndex].classList.add('hidden1');

        }

    }

    // tính chỉ số của phần tử phía sau

    // Nếu slide đang ở vị trí cuối cùng, bấm tiến sẽ phải trả về 0 (slide xoay vòng)

    // nếu chưa ở vị trí cuối cùng, thì vẫn tiến bình thường

    nextIndex = presentIndex + 1;

    presentIndex = nextIndex > SLIDE.length - 1 ?  0  :  nextIndex;

    console.log("Chỉ số slide hiện tại là : " ,presentIndex);

    for (let y = 0; y < SLIDE.length - 1; y++){

        SLIDE[presentIndex].classList.remove('hidden');

        SLIDE[presentIndex].classList.remove('hidden1');

    }

    SLIDE[presentIndex].classList.add('show1');

});

3.4. Xử lí việc người dùng click nhiều lần

3.4.1. Ý tưởng

    // tạo ra biến để kiểm tra

    var movingStatus = false;

    // thêm sự kiện khi nút chuyển slide phía bên trái được click

    LEFT.addEventListener('click', () => {

        // nếu moving = true thì return luôn, ko chạy đoạn code phía dưới

        if (movingStatus == true){ return false;}

        // set moving = true để chắc chắn rằng hàm ko chạy liên tiếp 2 lần

        movingStatus = true;

// code ...

      // dùng sự kiện webkitAnimationEnd để kiểm tra

// chỉ khi animation kết thúc thì mới trả movingStatus về lại false

// từ đó người dùng có thể nhấn tiếp

SLIDE[presentIndex].addEventListener('webkitAnimationEnd',()=> {

     movingStatus = false;

})

3.4.2.Code đầy đủ

    // tạo ra biến để kiểm tra

    var movingStatus = false;

    // thêm sự kiện khi nút chuyển slide phía bên trái được click

    LEFT.addEventListener('click', () => {

        // nếu moving = true thì return luôn, ko chạy đoạn code phía dưới

        if (movingStatus == true){ return false;}

        // set moving = true để chắc chắn rằng hàm ko chạy liên tiếp 2 lần

        movingStatus = true;

        for (let i = 0; i < SLIDE.length - 1 ; i++) {

                SLIDE[presentIndex].classList.remove('show');

                SLIDE[presentIndex].classList.remove('show1');

                SLIDE[presentIndex].classList.add('hidden');

        }

        // tính chỉ số của phần tử phía trước

        previousIndex = presentIndex - 1;

        presentIndex = previousIndex < 0 ?  SLIDE.length - 1  :  previousIndex;

        console.log("Chỉ số slide hiện tại là : " ,presentIndex);

        for (let y = 0; y < SLIDE.length - 1; y++){

            SLIDE[presentIndex].classList.remove('hidden');

            SLIDE[presentIndex].classList.remove('hidden1');

        }

        // phần tử tiếp theo hiển thị ra thêm class show vào

        SLIDE[presentIndex].classList.add('show');

        // dùng sự kiện webkitAnimationEnd để kiểm tra

        // chỉ khi animation kết thúc thì mới trả movingStatus về lại false

        // từ đó người dùng có thể nhấn tiếp

        SLIDE[presentIndex].addEventListener('webkitAnimationEnd',()=> {

            movingStatus = false;

        })

    });

    // nút phải tương tự, chỉ có điều sửa chỗ tính toán phần tử tiếp theo

    RIGHT.addEventListener('click', () => {

        if (movingStatus == true){ return false;}

        movingStatus = true;

        for (let i = 0; i < SLIDE.length - 1 ; i++) {

                SLIDE[presentIndex].classList.remove('show');

                SLIDE[presentIndex].classList.remove('show1');

                SLIDE[presentIndex].classList.add('hidden1');

        }

        // tính chỉ số của phần tử phía sau

        // Nếu slide đang ở vị trí cuối cùng, bấm tiến sẽ phải trả về 0 (slide xoay vòng)

        // nếu chưa ở vị trí cuối cùng, thì vẫn tiến bình thường

        nextIndex = presentIndex + 1;

        presentIndex = nextIndex > SLIDE.length - 1 ?  0  :  nextIndex;

        console.log("Chỉ số slide hiện tại là : " ,presentIndex);

        for (let y = 0; y < SLIDE.length - 1; y++){

            SLIDE[presentIndex].classList.remove('hidden');

            SLIDE[presentIndex].classList.remove('hidden1');

        }

        SLIDE[presentIndex].classList.add('show1');

        SLIDE[presentIndex].addEventListener('webkitAnimationEnd',()=> {

            movingStatus = false;

        })

    });

< còn tiếp phần nâng cao phía dưới nhá >

4.Nâng cao 1

Từ đoạn code trên, ta thấy rõ việc 2 nút này thực hiện các thao tác giống nhau, nhưng phải viết lại 2 lần cho 2 nút. vậy tại sao không viết 1 function để xử lí 1 lần cả 2 nút nhỉ?

4.1.ý tưởng

Viết 1 function để check xem là nút được nhấn là nút trái hay nút phải

Sau đó thêm sự kiện cho 2 nút với tham số tương ứng

function leftOrRight(*button*){

        if (*button* == 'left'){

            // code ở đây

            console.log('left');

        }

        if (*button* == 'right'){

            // code ở đây

            console.log('right');

        }

    }

LEFT.addEventListener('click', () => {leftOrRight('left')});

RIGHT.addEventListener('click', () => { leftOrRight('right')});

4.2. Code hoàn chỉnh

function leftOrRight(*button*){

    // dùng vòng lặp để xóa class show trước khi thêm mới class hidden

    if (movingStatus == true){ return false;}

        movingStatus = true;

    for (let i = 0; i < SLIDE.length - 1 ; i++) {

        SLIDE[presentIndex].classList.remove('show');

        SLIDE[presentIndex].classList.remove('show1');

    }

    // kiểm tra nút được nhấn là nút nào, từ đó thêm class tương ứng

    if (*button* == 'left'){

        SLIDE[presentIndex].classList.add('hidden');

        presentIndex = presentIndex - 1;

        if (presentIndex < 0){

            presentIndex = SLIDE.length - 1;

        }

    }

    if (*button* == 'right'){

        SLIDE[presentIndex].classList.add('hidden1');

        presentIndex = presentIndex + 1;

        if (presentIndex > SLIDE.length - 1){

            presentIndex = 0;

        }

    }

    // dùng vòng lặp để xóa class hidden trước khi thêm class show

    for (let y = 0; y < SLIDE.length - 1; y++){

        SLIDE[presentIndex].classList.remove('hidden');

        SLIDE[presentIndex].classList.remove('hidden1');

    }

    // kiểm tra nút để thêm class tương ứng

    if (*button* == 'left'){

        SLIDE[presentIndex].classList.add('show');

    } else {

        SLIDE[presentIndex].classList.add('show1');

    }

    SLIDE[presentIndex].addEventListener('webkitAnimationEnd',()=> {

        movingStatus = false;

    })

}

LEFT.addEventListener('click', () => {leftOrRight('left')});

RIGHT.addEventListener('click', () => { leftOrRight('right')});

5.Nâng cao 2

Phần này sẽ dùng toán tử 3 ngôi để thu gọn code hơn nữa ( rút ngắn gần 1 nửa số dòng code ), nhưng sẽ gây khó đọc hiểu code, người sau tiếp nhận code sẽ khó hình dung luồng logic đi như thế nào, chủ yếu cho nó ngầu ấy mà :)) . Vì thế nên phần này mình sẽ không đề cập ở đây nha.

6.Auto slide

Sẽ cập nhật sau ... Coming soon :))