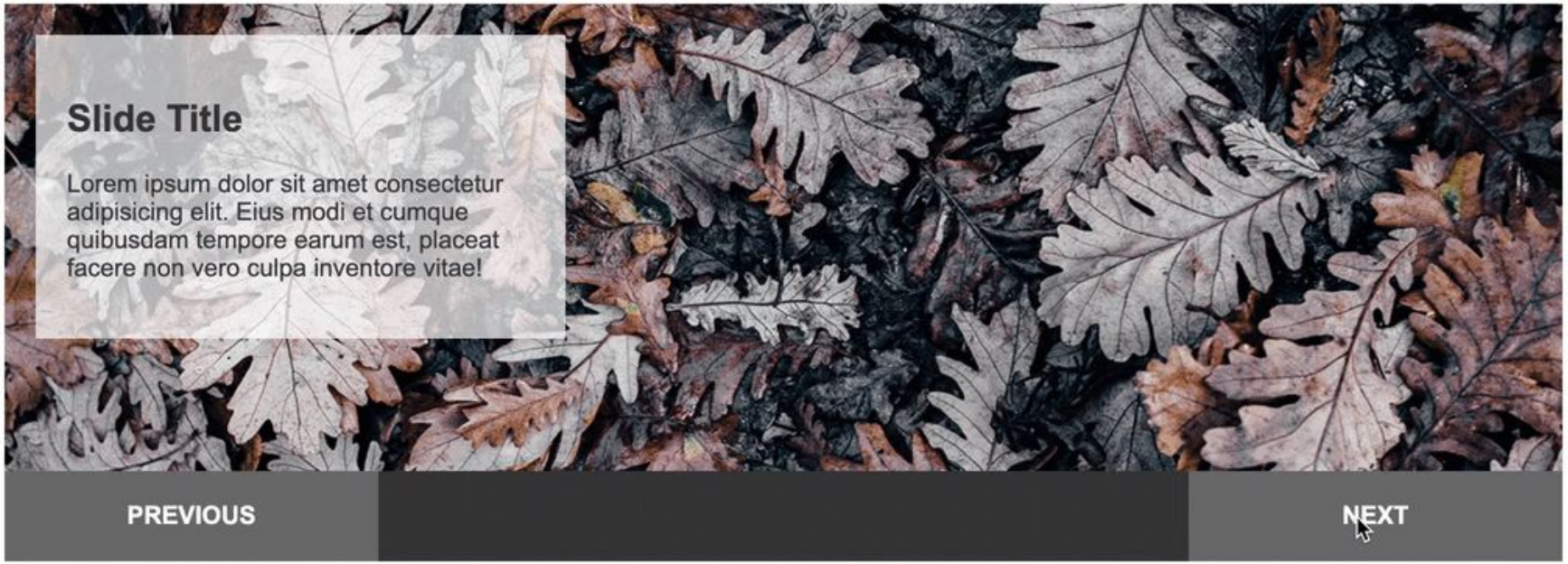
Slider

Using JavaScript

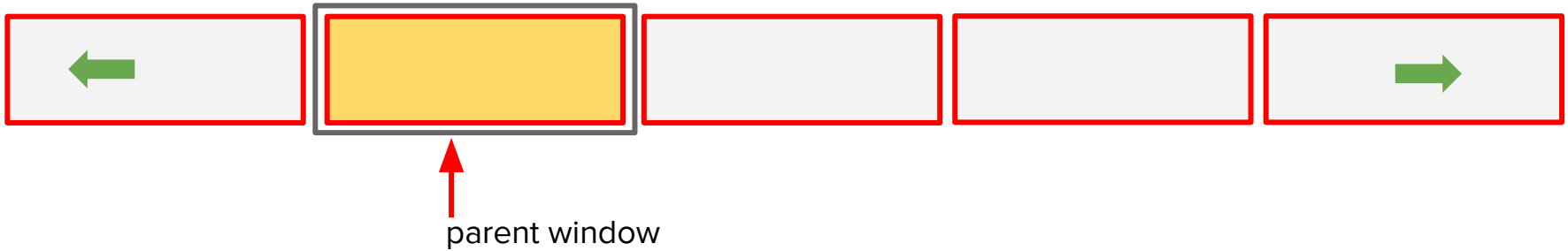
Final Version



Basic Strategy

The basic strategy is that there will be an element that wraps around the slider, with overflow: hidden, so that it acts like a window for the slides.

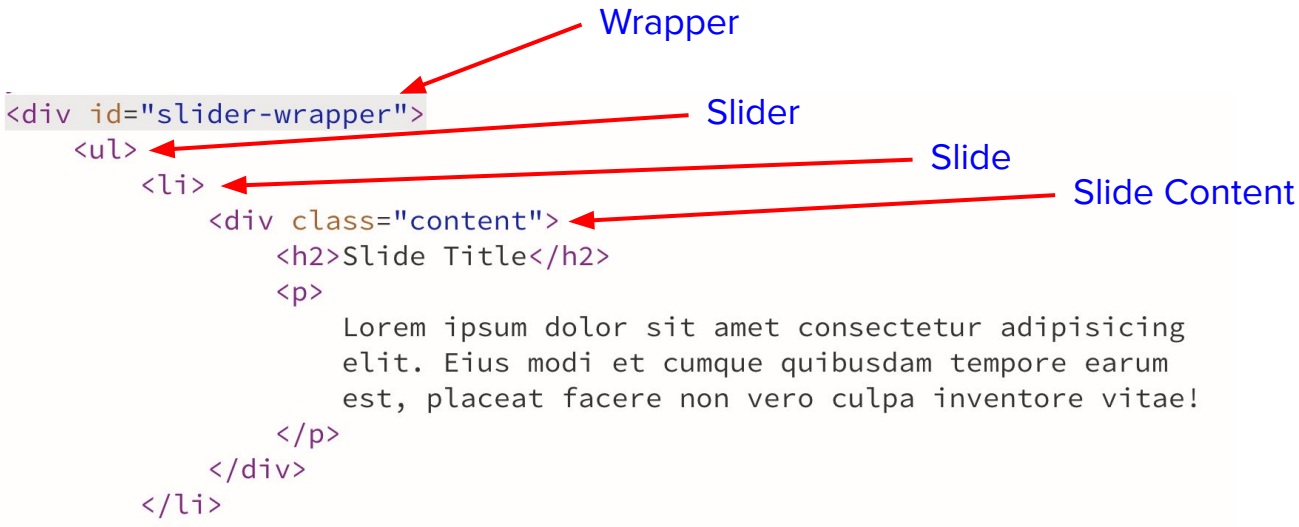
The the slides are all lined up in a row, and slide left and right into and out of the parent container window.



HTML

There is a wrapper that is set to a specific width and has overflow: hidden, so you don't see the slides that are outside of the view.

The slider is the unordered list and each slide is a list item. The slides can contain content, and the background images are in the CSS.




```
<div id="slider-wrapper">
    <ul>
        <li>
            <div class="content">
                <h2>Slide Title</h2>
                <p>
                    Lorem ipsum dolor sit amet consectetur adipisicing
                    elit. Eius modi et cumque quibusdam tempore earum
                    est, placeat facere non vero culpa inventore vitae!
                </p>
            </div>
        </li>
```

CSS for the Wrapper

The slider-wrapper is set to position relative, so that the slider inside it can be positioned absolutely in relation to that parent element.

Overflow: hidden is added so that you don't see the slides that are outside the element.

```
#slider-wrapper {
    width: 1000px;
    height: 358px;
    margin: auto;
    position: relative;
    overflow: hidden;
}
```

CSS for the Slider

The unordered list inside the slider is given a width, but it's not really necessary as that will also get calculated by the JS.

It is position absolutely and pinned, initially to the top left of the slider-wrapper element

The transition highlighted in this code below will provide all the animation for each slide as they slide left or right.

You get try different easing options by getting the code from the ceaser website.

```
#slider-wrapper ul {
    width: 5000px;
    overflow: hidden;
    position: absolute;
    top: 0;
    left: 0;
    transition: left 700ms cubic-bezier(0.165, 0.84, 0.44, 1);
}
```

CSS for the Slides

Each slide is 1000px wide and is floated left, so they all line up horizontally in one long strip. Each one has width and a height.

Background cover is there to take the background image, which is in the other rules.

Position: relative is set so that text content inside the slide can be positioned.

Each slide has a background image.

```
#slider-wrapper ul li {
    background-size: cover;
    position: relative;
    width: 1000px;
    height: 300px;
    float: left;
}
```

```
#slider-wrapper ul li:nth-child(1) {
    background-image: url(images/leaves01.jpg);
}
#slider-wrapper ul li:nth-child(2) {
    background-image: url(images/leaves02.jpg);
}
```

Adding Variables

It's a good idea to put this entire script inside an event handler for when the window loads so that it doesn't do anything until all the pictures have downloaded.

Each variable is described in the comment above it.

```
window.addEventListener('load', function() {
    //How many slides?
    const slideCount = document.querySelectorAll('#slider-wrapper ul li').length;
    //How wide is each slide?
    const slideWidth = document.querySelector('#slider-wrapper').offsetWidth;
    //Total width of the slider
    const totalWidth = slideCount * slideWidth + 'px';
    //Slider DOM element
    const slider = document.querySelector('#slider-wrapper ul');
    //Next button
    const next = document.getElementById('next');
    //Previous button
    const previous = document.getElementById('prev');
});
```

A few more variables

The leftPostion variable will change as the slider slides left and right. The counter will be used to keep track of which slide is in the window.

Finally, the width of the slider is set. This is not really necessary because the same calculation is in the CSS, but it could be useful if you wanted to make it responsive or add or remove slides.

```
// Upper left corner of slider
let leftPosition = 0;
//To keep track of each slide
let counter = 0;
//Sets the width of the slider (which is also in the CSS)
slider.style.width = totalWidth;
```

Next Slide Click Handler

When you click the next button the slider will slide to the next slide. If the user is at the end of the slide strip, the slider will slide back to the beginning.

```
next.addEventListener('click', function(evt) {
    evt.preventDefault();
    counter++;
    if (counter == slideCount) {
        // set the counter to 0
        // set the left position to 0
        // move the slider into position
    } else {
        // move the slider to the next slide.
    }
});
```

Filling in the Click Handler for Next

Here is the next step for the next slide click handler. If the user is on the last slide and clicks next, set the counter and the leftPosition back to zero and move the slider to that position.

Otherwise, set the left position to the next slide and move the slider to that position.

Note that the actual animation is in the CSS.

Before we go on, if you look carefully, you can see that this click handler can be simplified.

```
next.addEventListener('click', function(evt) {
    evt.preventDefault();
    counter++;
    if (counter == slideCount) {
        counter = 0;
        leftPosition = 0;
        slider.style.left = leftPosition;
    } else {
        leftPosition = `-${counter * slideWidth}px`;
        slider.style.left = leftPosition;
    }
});
```

Refactoring the Next Click Handler

Looking at both the if statement and the else statement on the previous slide, you can see they are both doing pretty much the same thing.

In fact, the only thing you need is an if statement that resets the counter variable, which simplifies the code considerably.

Always look for opportunities to simplify your code!

Can you do the previous slide click handler on your own?

```
next.addEventListener('click', function(evt) {
    evt.preventDefault();
    counter++;
    if (counter == slideCount) { counter = 0; }
    leftPosition = `-${counter * slideWidth}px`;
    slider.style.left = leftPosition;

});
```

Previous Click Handler

The previous click handler looks similar, but goes backwards. The counter is decremented each time the button is clicked and if you are on the first slide, it sends you back to the other end of the strip of slides.

```
previous.addEventListener('click', function(evt) {
    evt.preventDefault();
    counter--;
    if (counter < 0) { counter = slideCount - 1;}
    leftPosition = `-${counter * slideWidth}px`;
    slider.style.left = leftPosition;

});
```

That's It!

That is the whole script, as shown on the right (the variable declarations are collapsed).

This script works really well because the animation is done by CSS. JS is just positioning the strip of slides with each click of the buttons.

If you wanted to make it so instead of sliding back to the beginning, the slides wrap around, what could you do to make that work?

```
window.addEventListener('load', function() {
    //How many slides?
    const slideCount = documen...

    next.addEventListener('click', function(evt) {
        evt.preventDefault();
        counter++;
        if (counter == slideCount) {
            counter = 0;
            leftPosition = 0;
            slider.style.left = leftPosition;
        } else {
            leftPosition = `-${counter * slideWidth}px`;
            slider.style.left = leftPosition;
        }
    });

    previous.addEventListener('click', function(evt) {
        evt.preventDefault();
        counter--;
        if (counter < 0) {
            counter = slideCount - 1;
            leftPosition = `-${counter * slideWidth}px`;
            slider.style.left = leftPosition;
        } else {
            leftPosition = `-${counter * slideWidth}px`;
            slider.style.left = leftPosition;
        }
    });
});
```