

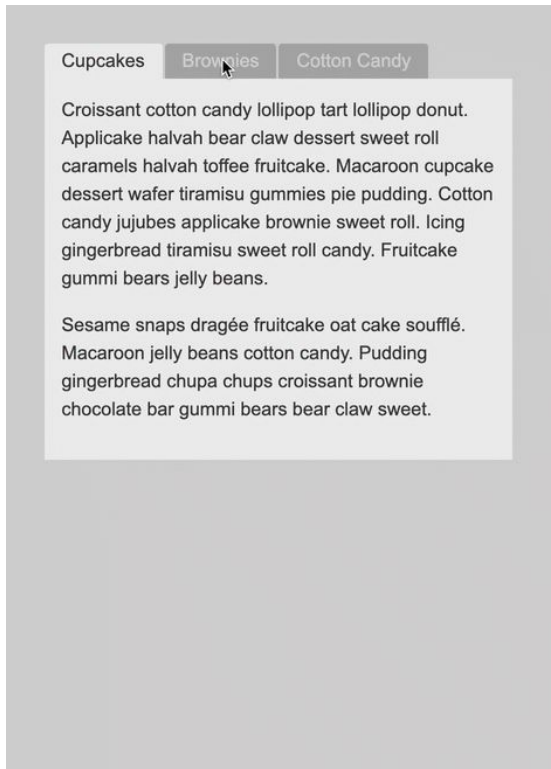
Tabbed Interface



With Plain JavaScript

Finished Version

In this project you will create a simple tabbed interface with an animated transition, from tab to tab.



The Start File

The start file contains some basic HTML.

A div with an id for the tabs, an unordered list with list items and anchor tags will become the actual tabs, while those anchor tags link to divs down the page that contain the information that will be shown for each tab.

Each tab has a few paragraphs inside it.

```
<div id="tabs">
  <ul>
    <li><a href="#tabs-1" class="active">Cupcakes</a></li>
    <li><a href="#tabs-2">Brownies</a></li>
    <li><a href="#tabs-3">Cotton Candy</a></li>
  </ul>

  <div id="tabs-1" class="visible">
    <p>
      Croissant cotton candy lol...
    </p>

    <p>
      Sesame snaps dragée fruitc...
    </p>
  </div>

  <div id="tabs-2" class="hidden">
    <p> Tiramisu jujubes bear ...
  </div>
  <div id="tabs-3" class="hidden">
    <p> Cupcake chocolate cake...
  </div>
</div>
```

Styling Needed

To start you need some basic styles to set up the tabs.

- [Cupcakes](#)
- [Brownies](#)
- [Cotton Candy](#)

Croissant cotton candy lollipop tart lollipop donut. Applicake halvah bear claw dessert sweet roll caramels halvah toffee fruitcake. Macaroon cupcake dessert wafer tiramisu gummies pie pudding. Cotton candy jujubes applicake brownie sweet roll. Icing gingerbread tiramisu sweet roll candy. Fruitcake gummi bears jelly beans.

Sesame snaps dragée fruitcake oat cake soufflé. Macaroon jelly beans cotton candy. Pudding gingerbread chupa chups croissant brownie chocolate bar gummi bears bear claw sweet.

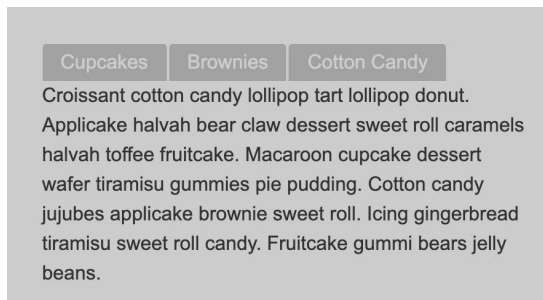
Tiramisu jujubes bear claw jelly-o brownie lemon drops brownie sesame snaps. Jelly-o cupcake sesame snaps muffin sesame snaps. Cookie croissant topping carrot cake candy canes gingerbread soufflé sweet roll pie. Tootsie roll chocolate tootsie roll lollipop. Lemon drops liquorice cotton candy gummies cake ice cream cheesecake cupcake muffin.

Tart bear claw biscuit bonbon topping wafer dessert sesame snaps ice cream. Chupa chups biscuit donut icing gummies unerdwear.com danish carrot cake cookie. Toffee soufflé gummi bears marzipan cupcake. Brownie chocolate jujubes sweet roll oat cake candy canes.

Styling the Tabs

Add these two rules to style the tabs...

```
#tabs > ul {  
  list-style-type: none;  
  display: flex;  
}  
  
#tabs > ul > li > a {  
  display: block;  
  height: 30px;  
  line-height: 30px;  
  margin-right: 2px;  
  background: #a2a2a2;  
  color: #cecece;  
  text-decoration: none;  
  padding: 0 15px;  
  border-top-left-radius: 3px;  
  border-top-right-radius: 3px;  
}
```

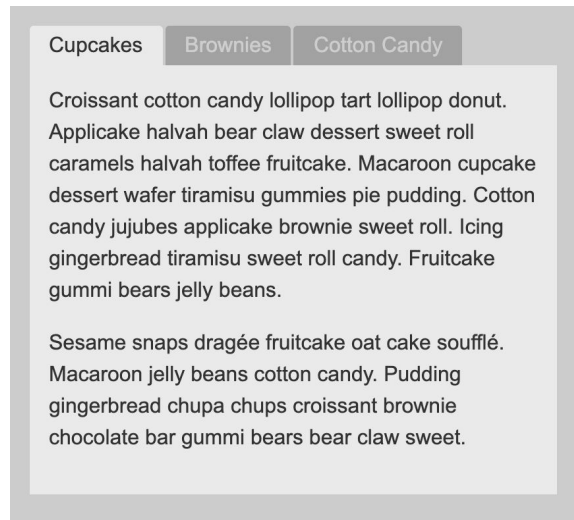


Final CSS Rules

These final CSS rules will get the rest of the styling in place. Notice the transition for the opacity. That is what will actually create the animation.

```
#tabs > div {  
  padding: 15px;  
  background: #eaeaea;  
  transition: opacity 200ms linear;  
}  
  
#tabs .active {  
  background: #eaeaea;  
  color: #333;  
}  
  
#tabs .hidden { display: none; }  
#tabs .visible { display: block; }  
#tabs .visuallyhidden { opacity: 0; }
```

Note that the classes of active, hidden and visible are controlling what is shown. The basic strategy will be to add, and remove classes to these elements when the user clicks the tabs.



Add a click handler to each tab

Next, on the script file, add a click handler to each tab. The very specific selector in the `querySelectorAll` is so that you don't accidentally grab any other elements.

The angle brackets (`>`) in the css selector mean "direct descendant". You wouldn't want to also grab anchor tags inside the tab content.

```
var tabs = document.querySelectorAll('#tabs > ul > li > a');

for (var i = 0; i < tabs.length; i++) {
  tabs[i].addEventListener('click', selectTab);
}

function selectTab(event){
}
```

Changing the tab class

Add this code to the selectTab function. First, prevent the default behavior of following the link.

```
function selectTab(event){  
  
    event.preventDefault();  
  
    for (var i = 0; i < tabs.length; i++) {  
        tabs[i].removeAttribute('class');  
    }  
  
    event.target.className = 'active';  
  
}
```

Then in a loop, remove the class from all the tabs, and finally, add it back on to the one that the user clicked.

The new tab and the old tab

Add the lines highlighted in blue below.

The first two lines will tell us which is the new tab that the user clicked on.

The second two lines hide the old tab by applying the visually hidden class. Because of the transition on the div for opacity, it fades out in 200 milliseconds.

```
event.target.className = 'active';
```

```
var thisTab = event.target.getAttribute('href');  
var thisContent = document.querySelector(thisTab);
```

```
var oldTabContent = document.querySelector('.visible');  
oldTabContent.className = 'visuallyhidden';
```

Handling the switch

This is the tricky part of this script. You want to wait until the transition ends, then set that original content to hidden (which is display: none).

You can do this by adding an event listener to the old tab content that fires when the transition is complete.

Then the new content will be able to fade in, but this will have to be done in a special way, as seen on the next slide.

```
oldTabContent.addEventListener('transitionend', function() {  
  oldTabContent.className = 'hidden';  
  // the new content (thisTab) is set to visible, but with the opacity at 0  
  // then after a short setTimeout, change the opacity to 1, so it fades in.  
});
```

Fading the new content in...

Add the code highlighted in blue to the event listener. First the new tab content, meaning the content that goes with the tab that was clicked, is given two classes. The first sets it to display: block, but the second sets the opacity to zero.

Then, after a VERY short amount of time, the visuallyhidden class is removed, and the content will fade in. This is necessary to get the fade to work properly.

There is a bug to fix though. If you try it, you will notice it.

```
oldTabContent.addEventListener('transitionend', function() {  
  oldTabContent.className = 'hidden';  
  
  thisContent.className = 'visible visuallyhidden';  
  setTimeout(function() {  
    thisContent.classList.remove('visuallyhidden');  
  }, 20);  
});
```

Removing the Event Listener

Add the code highlighted in blue to the end of the event listener.

Essentially, the flashing bug is happening because the event listener is getting added and then fired multiple times.

Setting `once` to `true`, means that the event listener will be automatically removed, after it fires once. `Capture` and `passive` can remain `false`.

This is it. It should be working properly. Now it is just a matter of refactoring code.

```
oldTabContent.addEventListener('transitionend', function() {  
  oldTabContent.className = 'hidden';  
  
  thisContent.className = 'visible visuallyhidden';  
  setTimeout(function() {  
    thisContent.classList.remove('visuallyhidden');  
  }, 20);  
}, { capture: false, once: true, passive: false });
```

Add the closure, and use const/let

To start with, add the closure shown below, then cut and paste the entire script inside the IIFE closure, so that there are no global variables.

Then go through and change all the var declarations to const or let.

```
(function(){  
    "use strict";  
})();
```

The image below shows most of the script.

```
(function(){  
    "use strict";  
  
    const tabs = document.querySelectorAll('#tabs > ul > li > a');  
    for (let i = 0; i < tabs.length; i++) {  
        tabs[i].addEventListener('click', selectTab);  
    }  
    function selectTab(event){  
        event.preventDefault();  
        for (let i = 0; i < tabs.length; i++) {  
            tabs[i].removeAttribute('class');  
        }  
        event.target.className = 'active';  
        const thisTab = event.target.getAttribute('href');  
        const thisContent = document.querySelector(thisTab);  
        const oldTabContent = document.querySelector('.visible');  
        oldTabContent.className = 'visuallyhidden';  
        oldTabContent.addEventListener('transitionend', function() {  
            oldTabContent.className = ...  
        }, { capture: false, once: true, passive: false });  
    }  
})();
```

Use `forEach` instead of a `for()` loop

JavaScript has a method for arrays called `forEach`. You can use this instead of the `for` loop. It's a little easier to read.

Below You can see that the `for` loop has been commented out, and the `forEach` method is used instead.

```
/*for (let i = 0; i < tabs.length; i++) {  
    tabs[i].addEventListener('click', selectTab);  
}*/  
  
tabs.forEach(function(tab) {  
    tab.addEventListener('click', selectTab);  
});
```

The `forEach` method is applied to an array, and has a callback function that it will run on each element in the array.

You pass in each item, in this case, "tab", which JavaScript can manipulate. In this case each tab is getting an event listener.

Fix the other loop as well

The other loop can be switched to the `forEach` method as well...

```
function selectTab(event){
  event.preventDefault();


  /*for (let i = 0; i < tabs.length; i++) {
    tabs[i].removeAttribute('class');
  } */

  tabs.forEach(function(tab) {
    tab.removeAttribute('class');
  });
}
```

Arrow Functions

Arrow functions will be covered in detail in the third course, but this is a good place to put a few and try it out. Arrow functions can make the code more compact and readable.

You can get rid of the word "function" and put => after the parentheses that go with the function. Like this...



```
tabs.forEach((tab) => {  
    tab.addListener('click', selectTab);  
});
```


Further refinements

Further, if there is only one parameter passed in, in this case "tab" you can get rid of the parentheses as well...

```
tabs.forEach( tab => {  
    tab.addEventListener('click', selectTab);  
});
```

This makes for a much more readable statement. "Tabs, for each tab, add an event listener".

Now you can fix the other functions.

The whole script...

```
(function(){  
  "use strict";  
  const tabs = document.querySelectorAll('#tabs > ul > li > a');  
  tabs.forEach( tab => { tab.addEventListener('click', selectTab); });  
  function selectTab(event){  
    event.preventDefault();  
  
    tabs.forEach( tab => { tab.removeAttribute('class'); });  
    event.target.className = 'active';  
    const thisTab = event.target.getAttribute('href');  
    const thisContent = document.querySelector(thisTab);  
    const oldTabContent = document.querySelector('.visible');  
    oldTabContent.className = 'visuallyhidden';  
    oldTabContent.addEventListener('transitionend', () => {  
      oldTabContent.className = 'hidden';  
      thisContent.className = 'visible visuallyhidden';  
      setTimeout( () => {  
        thisContent.classList.remove('visuallyhidden');  
      }, 20);  
    }, { capture: false, once: true, passive: false });  
  }  
})();
```