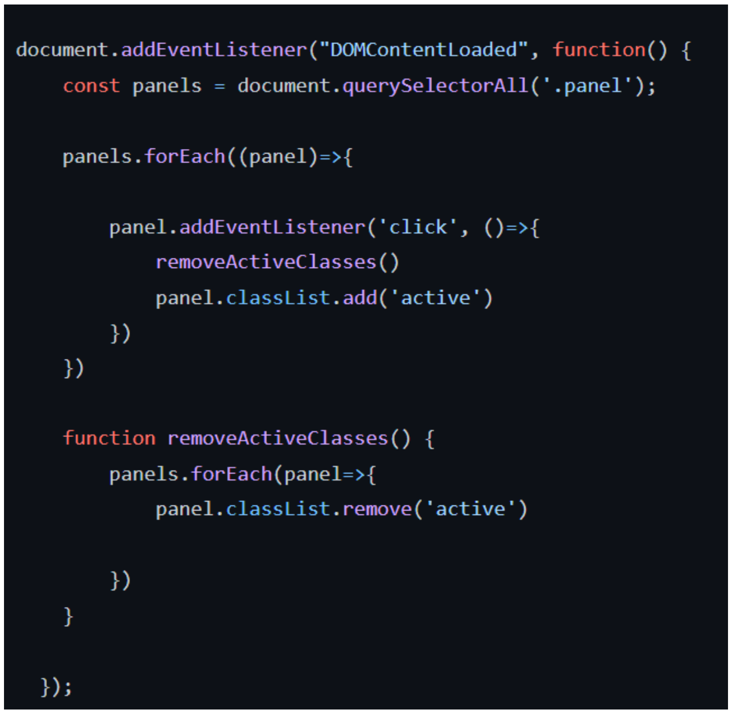
Gray Paper

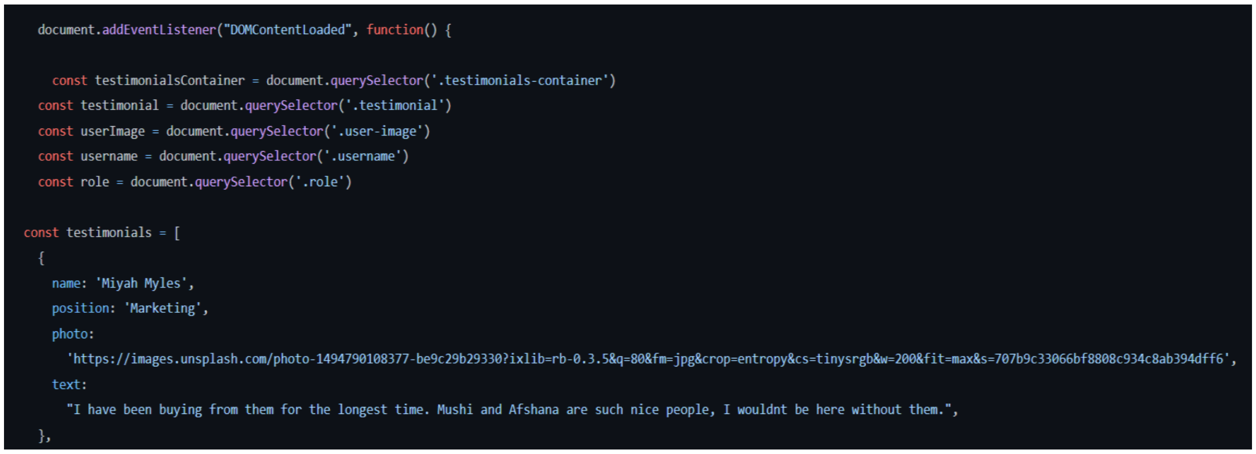
Code Snippet #1



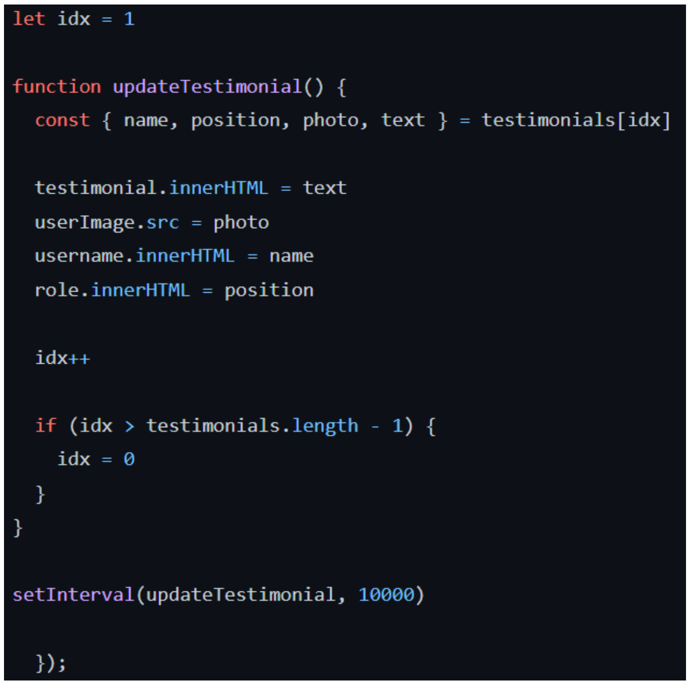
```
document.addEventListener("DOMContentLoaded", function() {
    const panels = document.querySelectorAll('.panel');

    panels.forEach((panel)=>{

        panel.addEventListener('click', ()=>{
            removeActiveClasses()
            panel.classList.add('active')
        })
    })

    function removeActiveClasses() {
        panels.forEach(panel=>{
            panel.classList.remove('active')

        })
    }

  });
```

Explanation: This is the code for expanding cards. A click event listener is set up on all the HTML elements with the `.panel` class. So, when one panel (card) is clicked, the `active` class is added to that panel after which it is able to ‘expand’ (we are able to see that card fully). Doing this causes all the other panels to have the `active` class removed. We don’t see the other cards fully. This acts like a set of collapsible panels basically.

Code Snippet #2



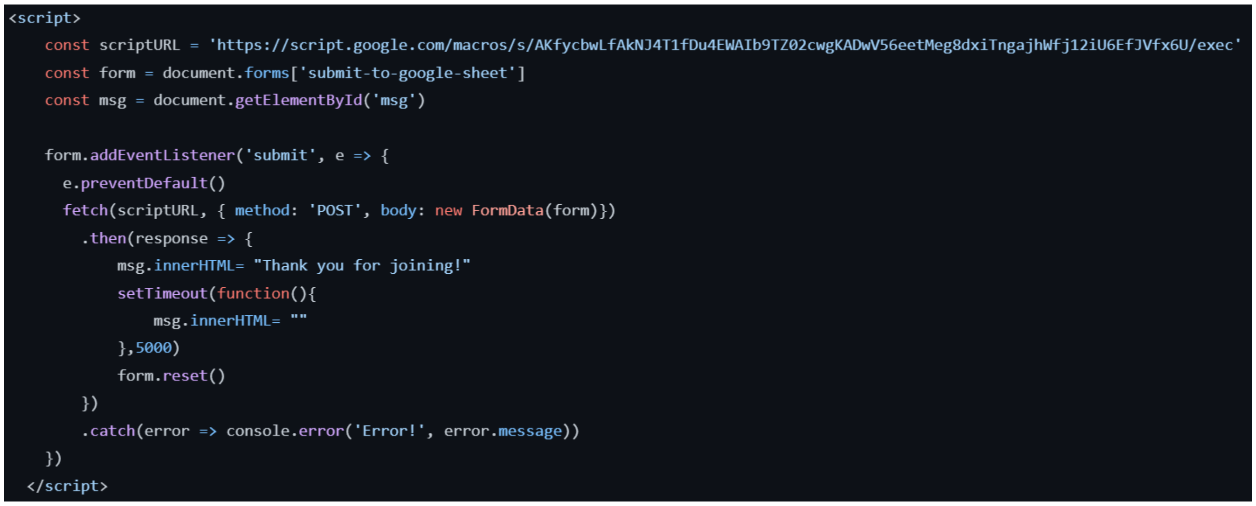
```
  document.addEventListener("DOMContentLoaded", function() {

    const testimonialsContainer = document.querySelector('.testimonials-container')
  const testimonial = document.querySelector('.testimonial')
  const userImage = document.querySelector('.user-image')
  const username = document.querySelector('.username')
  const role = document.querySelector('.role')

const testimonials = [
  {
    name: 'Miyah Myles',
    position: 'Marketing',
    photo:
      'https://images.unsplash.com/photo-1494790108377-be9c29b29330?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=200&fit=max&s=707b9c33066bf8808c934c8ab394dff6',
    text:
      "I have been buying from them for the longest time. Mushi and Afshana are such nice people, I wouldnt be here without them.",
  },
```



```
let idx = 1

function updateTestimonial() {
  const { name, position, photo, text } = testimonials[idx]

  testimonial.innerHTML = text
  userImage.src = photo
  username.innerHTML = name
  role.innerHTML = position

  idx++

  if (idx > testimonials.length - 1) {
    idx = 0
  }
}

setInterval(updateTestimonial, 10000)

  });
```

Explanation: Both snippets are the code for implementing a testimonial box switcher. The setInterval() method repeatedly calls the updateTestimonial() function, to update the HTML elements for the testimonial text, user-image, username, and role with the values from the current testimonial object. The testimonials are stored in an array of objects that includes the name, position, photo, and text of each testimonial. The `idx` variable keeps track of the current index in the array of testimonials and once the end of the array is reached, the `idx` variable is reset to 0 and the slider loops back from the beginning. Complications occurred while dealing with the CSS part of this code and through trial and error, we were able to manage to set the display properties correctly for the testimonial box and its contents inside.

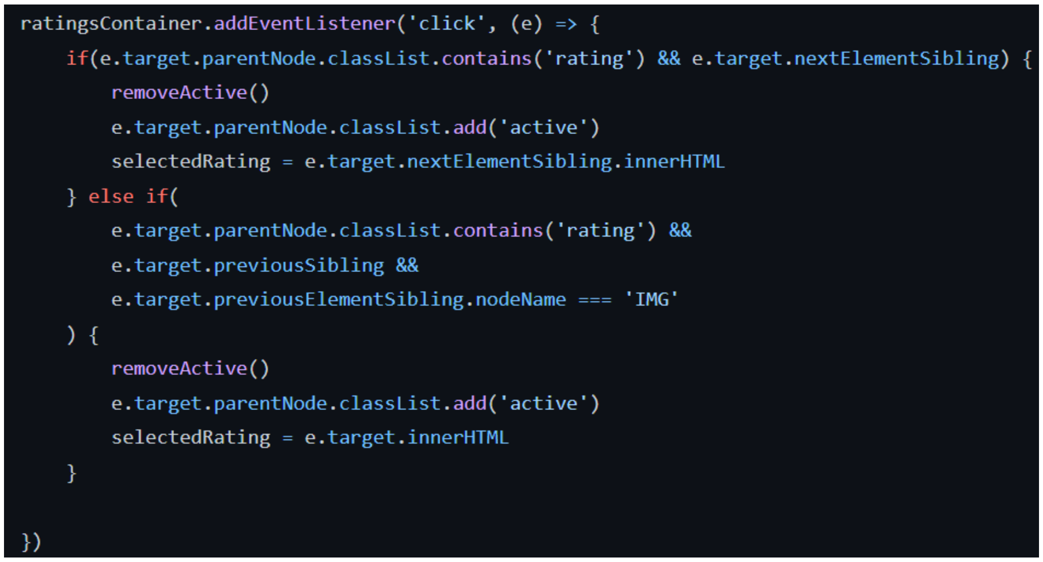
Code Snippet #3



```
<script>
    const scriptURL = 'https://script.google.com/macros/s/AKfycbwLfAkNJ4T1fDu4EWAIb9TZ02cwgKADwV56eetMeg8dxiTngajhWfj12iU6EfJVfx6U/exec'
    const form = document.forms['submit-to-google-sheet']
    const msg = document.getElementById('msg')

    form.addEventListener('submit', e => {
      e.preventDefault()
      fetch(scriptURL, { method: 'POST', body: new FormData(form)})
        .then(response => {
            msg.innerHTML= "Thank you for joining!"
            setTimeout(function(){
                msg.innerHTML= ""
            },5000)
            form.reset()
        })
        .catch(error => console.error('Error!', error.message))
    })
  </script>
```

Explanation: This code sets up a form submission event listener that sends the form data to a Google Sheets Spreadsheet using Google Apps Script. The constant variable `scriptURL` points to the URL of the Google Apps Script web app. An event listener is added to the form for the `'submit'` event. ` e.preventDefault()` stops the default form submission action and instead has the data from the form be sent to the Google Apps Script web page using a `fetch` request. If the request is successful, a message is displayed to the user using the `msg` element, thanking them for joining, and the form is reset. The difficulty with implementing this was that the data from the form was not being sent to a file, when the form was submitted, the data would disappear. This was handled by the addition of the Google Apps Script web app that allowed to submit form data to a Google sheet where we were able to collect all information entered by a user.

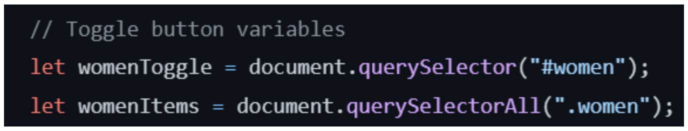
Code Snippet #4



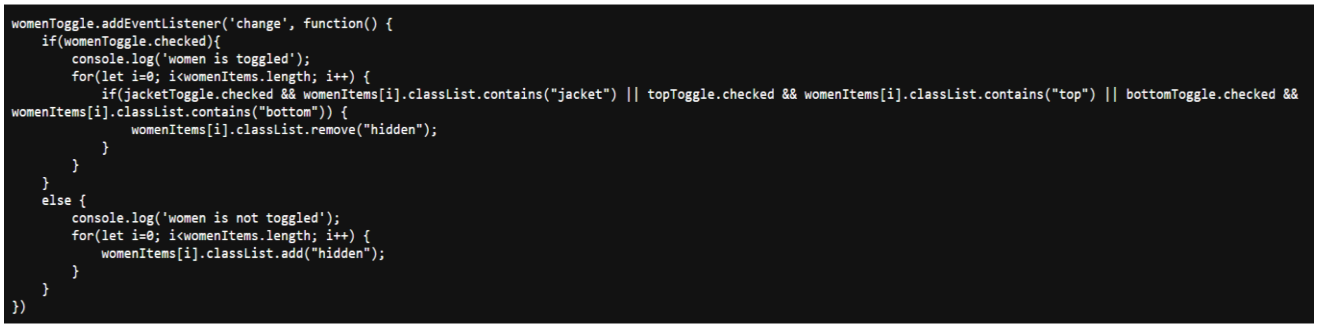
```
ratingsContainer.addEventListener('click', (e) => {
    if(e.target.parentNode.classList.contains('rating') && e.target.nextElementSibling) {
        removeActive()
        e.target.parentNode.classList.add('active')
        selectedRating = e.target.nextElementSibling.innerHTML
    } else if(
        e.target.parentNode.classList.contains('rating') &&
        e.target.previousSibling &&
        e.target.previousElementSibling.nodeName === 'IMG'
    ) {
        removeActive()
        e.target.parentNode.classList.add('active')
        selectedRating = e.target.innerHTML
    }

})
```

Explanation: This code implements a basic rating system. It chooses all HTML elements with the class `rating` and applies an event listener to their parent container. When a user clicks on one of the ratings, that rating becomes the chosen rating and is highlighted by adding the class `active` to its parent element. Then the next sibling element is assigned to the 'selectedRating' variable. If a different rating was previously chosen, the `active` class is removed from the parent element by removing the previously active rating for the previous sibling that is an 'IMG' element. The removeActive() function is called to remove the active class from any previously active rating. The 'selectedRating' variable is used to store the rating that was clicked on by the user.

Code Snippet #5



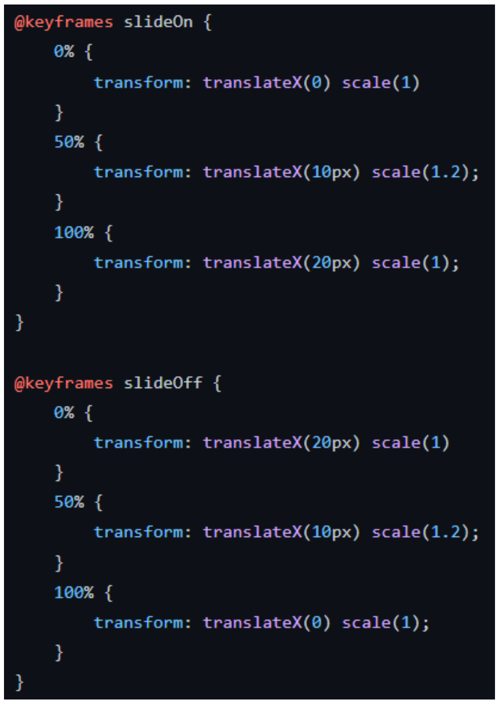
```
// Toggle button variables
let womenToggle = document.querySelector("#women");
let womenItems = document.querySelectorAll(".women");
```



```
womenToggle.addEventListener('change', function() {
    if(womenToggle.checked){
        console.log('women is toggled');
        for(let i=0; i<womenItems.length; i++) {
            if(jacketToggle.checked && womenItems[i].classList.contains("jacket") || topToggle.checked && womenItems[i].classList.contains("top") || bottomToggle.checked &&
womenItems[i].classList.contains("bottom")) {
                womenItems[i].classList.remove("hidden");
            }
        }
    }
    else {
        console.log('women is not toggled');
        for(let i=0; i<womenItems.length; i++) {
            womenItems[i].classList.add("hidden");
        }
    }
})
```

Explanation: This code adds event listeners to several checkboxes and item lists in order to toggle the visibility of items based on the checkboxes chosen. It specifically listens for changes to the ‘womenToggle’ checkbox and toggles the visibility of things in the ‘womenItems’ list as needed. If the checkbox is selected, it checks the states of the other checkboxes (‘jacketToggle’, ‘topToggle’, ‘bottomToggle’) and "hidden" class is removed from the item's class list, making it visible on the webpage; (displays items in ‘womenItems’) that have the corresponding class (e.g. "jacket") and are also selected in the respective checkbox. When the checkbox is unchecked, all things in ‘womenItems’ are hidden.

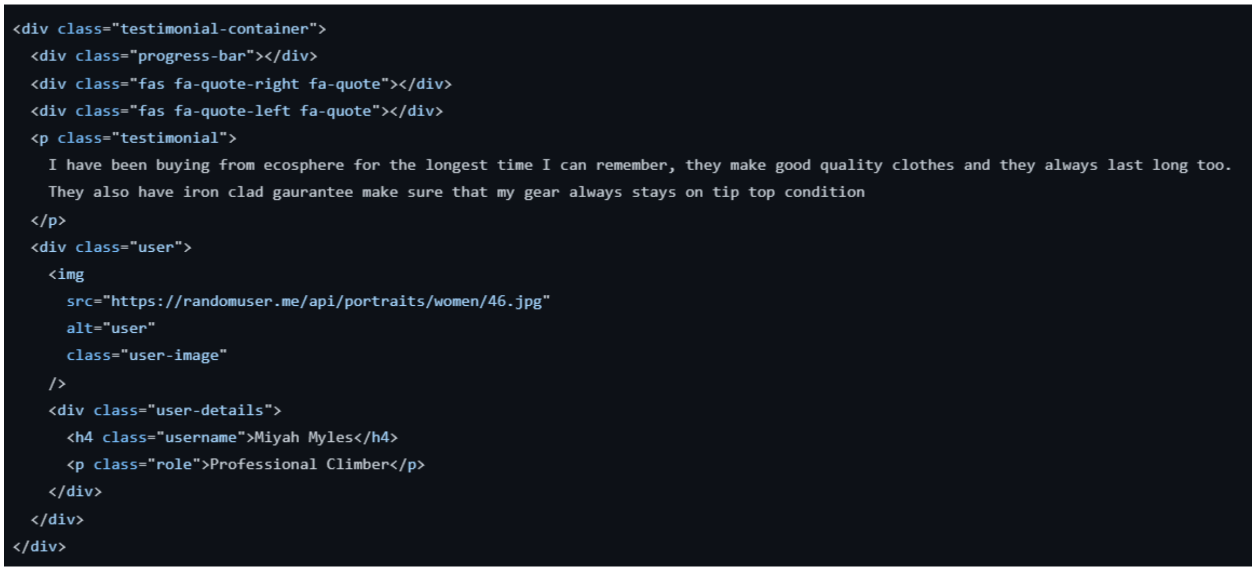
Code Snippet #6



```
@keyframes slideOn {
    0% {
        transform: translateX(0) scale(1)
    }
    50% {
        transform: translateX(10px) scale(1.2);
    }
    100% {
        transform: translateX(20px) scale(1);
    }
}

@keyframes slideOff {
    0% {
        transform: translateX(20px) scale(1)
    }
    50% {
        transform: translateX(10px) scale(1.2);
    }
    100% {
        transform: translateX(0) scale(1);
    }
}
```

Explanation: This code defines two keyframe animations: `slideOn` and `slideOff`. They use the CSS transform property to animate an element on and off the screen (position & scale). The `slideOn` animation moves an element to the right and increases its size, while the `slideOff` animation moves it back to its original position and size. The percentage values inside each keyframe rule indicate the progress of the animation, with 0% being the starting point and 100% being the end point. In `slideOn`, the element starts at its original position and size (0%) and moves to the right and increases in size by 10px at the midpoint (50%) before returning to its original position and size at the end (100%), scaled back to 1. In `slideOff`, the element starts at the end position and size of `slideOn` (0%), moves to the left and increases in size at the midpoint (50%), and returns to its original position and size at the end (100%).

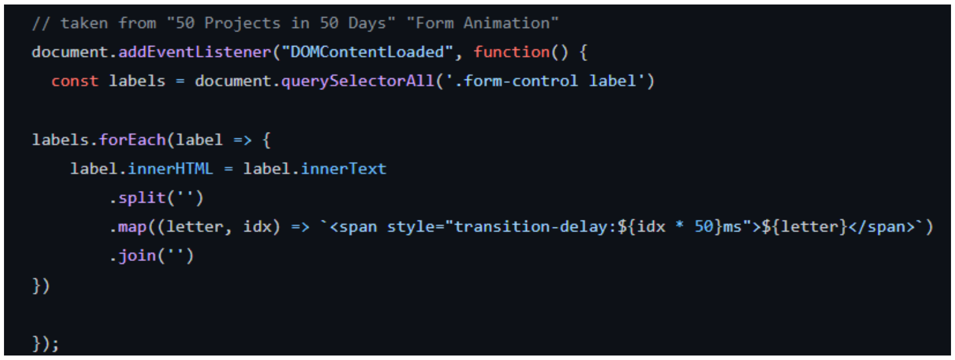
Code Snippet #7



```
<div class="testimonial-container">
  <div class="progress-bar"></div>
  <div class="fas fa-quote-right fa-quote"></div>
  <div class="fas fa-quote-left fa-quote"></div>
  <p class="testimonial">
    I have been buying from ecosphere for the longest time I can remember, they make good quality clothes and they always last long too.
    They also have iron clad gaurantee make sure that my gear always stays on tip top condition
  </p>
  <div class="user">
    <img
      src="https://randomuser.me/api/portraits/women/46.jpg"
      alt="user"
      class="user-image"
    />
    <div class="user-details">
      <h4 class="username">Miyah Myles</h4>
      <p class="role">Professional Climber</p>
    </div>
  </div>
</div>
```

Explanation: This code defines an HTML structure for a testimonial section on a webpage. The first div with a class of "testimonial-container" is the main container for the testimonial section. Inside this container, there are several other elements: A div with a class of "progress-bar" is used to visually indicate how long the testimonial box will stay on the screen before switching to the next one. Two divs with classes of "fas fa-quote-right fa-quote" and "fas fa-quote-left fa-quote" are Font Awesome icons used to indicate the beginning and end of the testimonial quote. The p element with class "testimonial" contains the actual text of the testimonial. A div with the class "user" contains information about the user who provided the testimonial. This includes an image, a username, and a role (professional climber, in this case).

Code Snippet #8



```
  // taken from "50 Projects in 50 Days" "Form Animation"
  document.addEventListener("DOMContentLoaded", function() {
    const labels = document.querySelectorAll('.form-control label')

  labels.forEach(label => {
      label.innerHTML = label.innerText
          .split('')
          .map((letter, idx) => `<span style="transition-delay:${idx * 50}ms">${letter}</span>`)
          .join('')
  })

  });
```

Explanation: This code listens for the ‘DOMContentLoaded’ event, which fires when the initial HTML document has been completely loaded and parsed. The code selects all elements with the class `form-control label` using the ‘querySelectorAll’ method and stores them in the ‘labels’ constant. For each label in the labels array, the code splits its inner text into an array of characters using the `.split` method, and maps each character to a new string that wraps it in an

HTML span element with a transition-delay CSS property. The transition-delay is set to idx * 50ms, where idx is the index of the current character in the array, multiplied by 50. This creates a staggered animation effect on the text as it is typed. Finally, the code joins the mapped string array back into a single string and sets it as the new `innerHTML` of the label, replacing the original text with the animated text. Hence, a visual affect is added to the labels by animating each letter of the label text with a slight delay between each character.