MIKESCHOBER.DEV

JavaScript Logic for your Portfolio - Part Three

Build your portfolio site and use JavaScript to make it come alive.

ROADMAP

- Code demo setup
- JavaScript arrays, and NodeLists
- For loops and iteration
- Counter and incrementing/decrementing
- Comparison operators
- Event.target
- Inline CSS styles
- Bringing it all together and making our demo work!

CODE DEMO SETUP

- The starter code for this video can be found on my GitHub account at: https://github.com/MikeSchober/carousel-demo
- This link is also in the description (YouTube) or below the video (my site)

JAVASCRIPT ARRAYS, AND NODELISTS

- A JavaScript array is a special object that in meant to store data
  - Think of it as an ordered-list
- Example of a JS array (assigned to variable):

```
let myArray = [1, 2, 3, 4];
```

- A NodeList is very similar to an array, but it is returned from a DOM method and it holds references to elements of our page (DOM elements):

Input: const boxes = document.querySelectorAll('.box');

Output: [div#box-one.box, div#box-two.box, div#box-three.box]

FOR LOOPS AND ITERATION

- “Iteration”
  - A very important concept in programming, iteration is the act of repeatedly executing some specific code
  - To “iterate over” the elements in an array is to execute some block of logic for each element
- We can use what are called “for loops” to iterate over an array or a NodeList

```
let myArray = [1, 2, 3, 4];
```
(array defined)

```
for (let x of myArray) {
      console.log(x*2);
  };
```
(for each element in the array, multiply it by 2, and print it to console)

COUNTER

- To keep track of our carousel elements, we will use a variable that holds a number
  - This variable will increase and decrease as we view different elements in the carousel
  - We will call this variable our “counter”
  - It will look like this:

```
let boxIndex = 0;
(variable initialized at zero)
```

INCREMENTING AND DECREMENTING

- As we move through our carousel elements, we will use what are called “increment assignment” and “decrement assignment” operators to adjust our counter accordingly
  - So, to add one to our counter (as we view the next element in our carousel), we use the increment assignment operator `+=`

```
boxIndex += 1;
(basically: boxIndex = boxIndex + 1)
```

  - So, to subtract one to our counter (as we view the previous element in our carousel), we use the decrement assignment operator `-=`

```
boxIndex -= 1;
(basically: boxIndex = boxIndex - 1)
```

COMPARISON OPERATORS

- Comparison operators are used to compare values and return (output) a boolean value (true or false) as to whether the comparison statement is true or false
  - Examples of comparison operators are `>`, `<`, `>=`, `<=`, and `===`
- Example of comparision operators in use:

```
let myNumber = 7;
```

(variable defined to hold Number)

Input: myNumber > 1;
Output: true
(because 7 >1, the comparison returns true)

EVENT.TARGET

- Target is a property within the DOM, that allows us to access the DOM element from which an event was triggered
  - For example, when we use .addEventListener() to listen for a click, event.target allows us to identify the element that was clicked
- Example of event.target in use:

```
controlPanel.addEventListener('click', (e) => {
    console.log(e.target.id);
});
```

(eventListener added to container that holds buttons)
(uses e.target to print the id of clicked button to the browser console)

Input: (previous button click in browser)
Output: prev-btn
(prints prev-btn because it is the id of the button that was clicked)

INLINE CSS STYLES

- Cascading Style Sheets (CSS) is the language that we use to define how elements are rendered in the browser
- This series does not cover CSS, but we will use the CSS transform property to move our carousel elements through our carousel viewer

- There are three ways of defining CSS styles for a document
  - External
    - using a separate file to define CSS
  - Internal
    - defining CSS within the <style> element in the HTML doc
  - Inline
    - defining CSS within individual HTML elements

- We will use inline CSS to move our elements through the carousel

INLINE CSS STYLES

- Example of inline CSS:

  `style="transform: translateX(-425px);";`

  (seen within the elements below, this is an example of inline CSS)

```
<div id="carousel">  flex
    <div class="box" id="box-one" style="transform: translateX(-425px);"></div>
    <div class="box" id="box-two" style="transform: translateX(-425px);"></div> ==
    <div class="box" id="box-three" style="transform: translateX(-425px);"></div>
</div>
```

- transform: translateX(); moves an HTML element horizontally by a percentage of the width of the element or by a specific number of pixels
  - In the example above, we are moving the elements to the left by 425px

  (-425px moves element left, 425px would move it to the right)

BRINGING IT ALL TOGETHER

- Let’s go write our code!
- Here is an outline of the structure of our script

- Variables defined
  - Variables defined to hold the following DOM elements:
    - All carousel elements (class=”box”)
    - The “PREVIOUS” button (id=”prev-btn”)
    - The “NEXT” button (id=”next-btn”)
    - The container that holds the buttons (id=”control-panel”)
  - Counter variable defined

- EventListener added to the button container to handle button clicks

BRINGING IT ALL TOGETHER

- Logic within the EventListener:
  - if “PREVIOUS” button clicked:
    - if we are viewing an element beyond the first element in the carousel:
      - move to previous element in the carousel by moving all carousel elements to right by 425px
      - subtract 1 from the boxIndex counter
    - else (we are viewing the first element in the carousel):
      - move to the last element in the carousel by moving all carousel elements to the left by 425px multiplied by the total number of carousel elements minus one
      - reassign assign boxIndex as the number of carousel elements minus one (to hold correct element number)

BRINGING IT ALL TOGETHER

- Logic within the EventListener continued:
  - if “NEXT” button clicked:
    - if we are viewing an element before the last element in the carousel:
      - move to next element in the carousel by moving all carousel elements to left by 425px
      - add 1 to the boxIndex counter
    - else (we are viewing the last element in the carousel):
      - move to the first element in the carousel by removing all inline styles
      - reassign assign boxIndex to zero (because we are now back to the first element in the carousel)

GREAT WEBSITES FOR REFERENCE!

Assignment before next video:

Use what you learned in this video, and try to write the logic for the carousel in our portfolio project. Don’t worry if it’s difficult. Give it a try, and in the next lesson, we will write this logic together.

GREAT WEBSITES FOR REFERENCE!

- I use the following websites when I need to look something up or understand something better:
  - https://www.w3schools.com/js/
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
  - https://stackoverflow.com/questions

  Check them out!

IN THE NEXT VIDEO...

- See links on this page for:
  - This presentation in PDF form
  - The link to the code for this demo from my GitHub profile
  - Additional learning resources coming soon!

- Reach out to me with any questions or feedback!
  - My email address:
    mike@mikeschober.dev