Lesson Wednesday

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Event Handling with Event Listeners

Text Cheat sheet

Summary

In this lesson we learned about event listening. Some key topics we covered are:

- Event listening is another method of event handling. **Event handling** is an umbrella term that describes the processes and tools that developers use to write code that responds to events.
- **Event listening** is the process of creating event listeners in our code to listen for and react to events that happen in our webpage.
- When we call `addEventListener()` on a target (an HTML element, `window`, or `document`), this process is called **registering** the event listener.
- We can create an event listener with the `addEventListener()` method, which takes two arguments:

    - The first argument is the event name as a string, like `"click"` or `"submit"`
    - The second argument is the **"handler" function**, the function that handles reacting to the event.
- The benefit of using event listeners is that we can create multiple handlers for the same event on the same target (HTML element, `window`, `document` or otherwise). This improves our code organization and how easy it is to read and understand. Event listeners are considered the recommended way to set up event handling in applications.
- Any function that is passed into another function/method as an argument is called a callback function.
    - The "handler" function that we pass into the `addEventListener()` method is a **callback function**.
    - Callback functions are all about the application of functions — where they are being used in our code — and they are important to asynchronous JavaScript, which we'll learn about down the road.

Code Examples

The Breakdown of `addEventListener()` Syntax

`addEventListener()` has two required parameters. In pseudocode, this looks like:

```
// this is pseudocode!
target.addEventListener(eventName, callbackFunc);
```

- `target` is the object we are targeting (an HTML element, the `document` object, or the `window` object)
- `eventName` is the first parameter. Here we'll pass in a string with the name of the event we're creating the listener for.
- `callbackFunc` is the second parameter. Here we'll pass in a function that contains all of the code that we want to run in

reaction to the event.

Using One Event Listener in the Mad Libs Project

The following code uses an event listener for the form 'submit' event.

js/scripts.js

```
// These are the scripts from the Mad Libs project.
window.onload = function() {
  let form = document.querySelector("form");
  form.addEventListener("submit", function(event) {
    ...
    ...
    event.preventDefault();
  // new code below
  });
};
```

Completed Scripts, CSS, and HTML for Using Multiple Event Listeners in Mad Libs

Here is the final script for our Mad Libs project that incorporates the new functionality of the reset button and the advertisement, and a separate event listener for each reaction to the form submission event.

css/styles.css

```
.hidden {
  display: none;
}
```

mad-libs.html

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <script src="js/scripts.js"></script>
  <link rel="stylesheet" href="css/styles.css" type="text/c
ss">
  <title>A fantastical adventure</title>
</head>
<body>
  <h1>Fill in the blanks to write your story!</h1>
  <form>
    <label for="person1Input">A name</label>
    <input id="person1Input" type="text" name="person1Inpu
t">
    <label for="person2Input">Another name</label>
    <input id="person2Input" type="text" name="person2Inpu
t">
    <label for="animalInput">An animal</label>
    <input id="animalInput" type="text" name="animalInput">
    <label for="exclamationInput">An exclamation</label>
    <input id="exclamationInput" type="text" name="exclamat
ionInput">
    <label for="verbInput">A past tense verb</label>
    <input id="verbInput" type="text" name="verbInput">
    <label for="nounInput">A noun</label>
    <input id="nounInput" type="text" name="nounInput">
    <button type="submit">Show me the story!</button>
  </form>
  <br />
  <button type="button" class="hidden" id="reset">Reset</bu
tton>
  <div id="story" class="hidden">
    <h1>A fantastical adventure</h1>
    <p>
      One day, <span id="person1a">_________</span> and <sp
an id="person2a">_________</span> were walking through the
woods, when suddenly a giant <span id="animal">_________</s
pan> appeared. "<span id="exclamation">_________</span>", <
span id="person1b">_________</span> cried. The two of them
<span id="verb">_________</span> as quickly possible, and w
hen they were safe, <span id="person1c">_________</span> an
```

```
d <span id="person2b">_________</span> gave each other a gi
ant <span id="noun">_________</span>.
    </p>
  </div>
</body>
</html>
```

js/scripts.js

```
// User Interface Logic
window.addEventListener("load", function() {
  let form = document.querySelector("form");
  let resetBtn = document.getElementById("reset");
  let story = document.getElementById("story");

  form.addEventListener("submit", function(event) {
    const person1Input = document.getElementById("person1In
put").value;
    const person2Input = document.getElementById("person2In
put").value;
    const animalInput= document.getElementById("animalInpu
t").value;
    const exclamationInput = document.getElementById("excla
mationInput").value;
    const verbInput = document.getElementById("verbInput").
value;
    const nounInput = document.getElementById("nounInput").
value;

    document.querySelector("span#person1a").innerText = per
son1Input;
    document.querySelector("span#person1b").innerText = per
son1Input;
    document.querySelector("span#person1c").innerText = per
son1Input;
    document.querySelector("span#person2a").innerText = per
son2Input;
    document.querySelector("span#person2b").innerText = per
son2Input;
    document.querySelector("span#animal").innerText = anima
lInput;
    document.querySelector("span#verb").innerText = verbInp
ut;
    document.querySelector("span#noun").innerText = nounInp
ut;
    document.querySelector("span#exclamation").innerText =
exclamationInput;

    story.removeAttribute("class");
    event.preventDefault();
```

```
  });

  form.addEventListener("submit", function() {
    resetBtn.removeAttribute("class");
  });

  form.addEventListener("submit", function() {
    window.alert("Do you need a new computer? Visit www.sup
erextracomputersales.com to find the best deals!");
  });

  resetBtn.addEventListener("click", function() {
    story.setAttribute("class", "hidden");
    document.getElementById("person1Input").value = null;
    document.getElementById("person2Input").value = null;
    document.getElementById("animalInput").value = null;
    document.getElementById("exclamationInput").value = nul
l;
    document.getElementById("verbInput").value = null;
    document.getElementById("nounInput").value = null;
  });
});
```

Previous (/introduction-to-programming/javascript-and-web-browsers/practice-more-branching)
Next (/introduction-to-programming/javascript-and-web-browsers/using-function-declarations-in-event-handling)

Lesson 63 of 75
Last updated March 24, 2023

disable dark mode

(http://www.epicodus.com)