Lesson **Wednesday**

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Other Ways to Organize UI Logic

Text

Not all UI logic should automatically be placed inside of the `window.onload` event handler! Even though we need to make sure that our page loads before setting up any event handlers, that doesn't mean that *all* event handler logic needs to be inside of the `window.onload` event handler. In this lesson, we're going to get a sense of what exactly this means by looking at other configurations for our Mad Libs' UI logic.

The goal of this lesson is to give you ideas of how you can organize your code, and encourage you to start thinking about different ways to organize your code so that it is easier to read and understand. As you read through each code snippet, pay attention to the comments.

Current Mad Libs' Scripts Organization

Currently, all of our scripts for the Mad Libs project are located within the `window.onload` event handler. Review the scripts now, before we move on to looking at other ways to organize our UI logic.

js/scripts.js

```
window.onload = function() {
  let form = document.querySelector("form");
  form.onsubmit = 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;

    document.querySelector("div#story").removeAttribute("cl
ass");

    event.preventDefault();
    };
  };
```

Other Ways to Organize our UI Logic

We'll look at two other ways to organize our UI logic, and we'll also address some best practices! Start by reviewing the code snippet below, and remember to pay attention to the comments as you go.

Configuration 2

js/scripts.js

```
// UI Logic

// this function handles getting form values and
// setting the values for the Mad Libs story
function getAndSetMadLibValues() {
  // in this section we get the value for each form input
  const person1Input = document.getElementById("person1Inpu
t").value;
  const person2Input = document.getElementById("person2Inpu
t").value;
  const animalInput= document.getElementById("animalInpu
t").value;
  const exclamationInput = document.getElementById("exclama
tionInput").value;
  const verbInput = document.getElementById("verbInput").va
lue;
  const nounInput = document.getElementById("nounInput").va
lue;

  // then we set the story variables to the values we got f
rom the form
  document.querySelector("span#person1a").innerText = perso
n1Input;
  document.querySelector("span#person1b").innerText = perso
n1Input;
  document.querySelector("span#person1c").innerText = perso
n1Input;
  document.querySelector("span#person2a").innerText = perso
n2Input;
  document.querySelector("span#person2b").innerText = perso
n2Input;
  document.querySelector("span#animal").innerText = animalI
nput;
  document.querySelector("span#verb").innerText = verbInpu
t;
  document.querySelector("span#noun").innerText = nounInpu
t;
  document.querySelector("span#exclamation").innerText = ex
clamationInput;
}
```

```
window.onload = function() {
  let form = document.querySelector("form");
  form.onsubmit = function(event) {
    // notice that event.preventDefault() can be located an
ywhere within
    // the onsubmit event handler
    event.preventDefault();
    // we call the new function in the onsubmit event handl
er
    getAndSetMadLibValues();
    document.querySelector("div#story").removeAttribute("cl
ass");
  };
};
```

The above scripts now includes a new function called `getAndSetMadLibValues()`:

- This function handles getting the form input values and updating the `<span>` elements in the Mad Libs story with the user inputted values.
- We call this function from within the `onsubmit` event handler.

Notably, we only need to create the event handlers (IE: `form.onsubmit`) inside of the `window.onload` event handler, but we don't need to include all of the logic (everything in `getAndSetMadLibValues()`) for that event handler inside of the `window.onload` event handler.

Using the `getAndSetMadLibValues()` function separates our UI logic by its functionality. The functionality of getting and setting the values for our Mad Libs story is all contained within one function that has a descriptive name. Because of this separation and descriptive naming, our code is a bit more organized and easier to read.

Notice the location of `getAndSetMadLibValues()` within our scripts:

- It's within UI logic. All code that handles accessing or updating the DOM is considered user interface logic. Code that does not access or update the DOM is considered business logic, and we'll see more examples of that in an upcoming lesson.
- It's outside of the `window.onload` event handler. In this case, we don't need to define this function within the `window.onload` event handler. We only need to *call* the function inside of the `window.onload` event handler (specifically the `onsubmit` event handler that is located inside of the `window.onload` event handler).
- It's above the `window.onload` event handler. We always place a function's definition above where we call on that function in our scripts.

Anytime we organize our scripts into multiple functions, we'll follow the best practices listed above.

Configuration #3

Now let's look at another configuration that builds off of the previous one. Check out the code below.

```
// UI Logic

function getAndSetMadLibValues() {
  const person1Input = document.getElementById("person1Inpu
t").value;
  const person2Input = document.getElementById("person2Inpu
t").value;
  const animalInput= document.getElementById("animalInpu
t").value;
  const exclamationInput = document.getElementById("exclama
tionInput").value;
  const verbInput = document.getElementById("verbInput").va
lue;
  const nounInput = document.getElementById("nounInput").va
lue;

  document.querySelector("span#person1a").innerText = perso
n1Input;
  document.querySelector("span#person1b").innerText = perso
n1Input;
  document.querySelector("span#person1c").innerText = perso
n1Input;
  document.querySelector("span#person2a").innerText = perso
n2Input;
  document.querySelector("span#person2b").innerText = perso
n2Input;
  document.querySelector("span#animal").innerText = animalI
nput;
  document.querySelector("span#verb").innerText = verbInpu
t;
  document.querySelector("span#noun").innerText = nounInpu
t;
  document.querySelector("span#exclamation").innerText = ex
clamationInput;
}

// this function creates the onsubmit event handler
// it needs to be called in the window.onload event handler
function setFormSubmissionEventHandler() {
  let form = document.querySelector("form");
  form.onsubmit = function(event) {
    event.preventDefault();
    getAndSetMadLibValues();
    document.querySelector("div#story").removeAttribute("cl
ass");
  }
}

window.onload = function() {
  // inside of the window.onload handler function we only i
nclude
  // the code that we want to run when
  // the webpage has finished loading all resources
  setFormSubmissionEventHandler();
};
```

We now use two functions to organize and separate our UI logic:

- `setFormSubmissionEventHandler()` handles creating the event handler for the form submission.
- `getAndSetMadLibValues()` does the same thing as when we used it in the last example: after the submission event, it gets the form values and sets the Mad Libs story values.

This new configuration separates our JavaScript UI logic into different functions based on its purpose. Each function is named descriptively so that it communicates what its purpose is in our website's functionality. There are more lines of code now, but this code is arguably more descriptive and easier to read and understand. What do you think?

If the above code organization is harder for you to understand, that is completely okay. Take some time to talk about it with your pair, dev team, or instructor, but don't spend too much time understanding it now. It's not super important as we'll be talking about code organization a lot in future lessons and projects. As always, do what makes sense to you and start exploring the idea of organizing your code into separate functions.

Summary

In this lesson, we learned that all JS that accesses and manipulates the DOM is considered UI logic, and that we can organize our UI logic into separate functions. Organizing code into separate functions makes our code easier to read and understand.

We also learned that we need to create event handlers (like for form submissions) within the `window.onload` event handler, but not all logic for each event handler needs to be inside of the `window.onload` event handler.

As you go along in this course section, we want you to start thinking about how you might organize your code in a way that promotes its readability. However, it's important to reiterate that when you are just starting out there's no one right way to code a solution or to organize your code. Do what makes sense to you and don't get hung up on achieving the most efficient or organized solution. As a baseline, make sure to:

- Use descriptive variable and function names.
- Always place a function's definition above where we call on that function in our scripts.
- Use comments to identify what different code does.
- As a starting point, ask yourself "what can I do to make my code easier to read and understand?"

Previous (/introduction-to-programming/javascript-and-web-browsers/forms-hiding-and-showing-elements-and-the-event-object)
Next (/introduction-to-programming/javascript-and-web-browsers/debugging-in-javascript-using-console-log)
Lesson 55 of 75
Last updated more than 3 months ago.

disable dark mode

Epicodus (http://www.epicodus.com)