Lesson Wednesday

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / More Branching and Error Handling

Text Cheat sheet

Sometimes, our branching logic can get more complicated. Let's build a website to determine which amusement park ride someone can go on based on their age and height. In the process, we'll learn about using the logical "and" `&&` and "or" `||` operators, nesting if statements, and the concepts of "truthy" and "falsey". We'll also include polish our website by adding error handling and a button to clear the form and results.

Amusement Park

At the Fun 4 All Amusement Park, we'll have 3 rides, each with a condition that we need to check for:

- You need to be at least 6 years old to ride the "Swings" swinging chair ride.
- You need to be either 12 years old or 48 or more inches tall to ride the Roller Coaster.

- You need to be 12 years old and 60 or more inches tall to ride the "Tower of Doom" with a big vertical drop.

On our webpage, we'll only display information about rides that the user is tall and old enough to safely ride.

Project Setup

Here's the HTML and JS that we'll start with:

css/styles.css

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

amusement-park.html

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <link href="css/styles.css" rel="stylesheet" type="text/c
ss">
  <script src="js/scripts.js"></script>
  <title>Amusement Park</title>
</head>
<body>
  <div>
    <h1>Fun 4 All</h1>

    <p>Welcome to the Fun 4 All amusement park!</p>
    <p>At Fun 4 All you need to be a certain age and height
to ride on our rides.</p>
    <p>Enter your age and height in the form below and we'l
l show you what rides you can go on.</p>

    <form id="userInfo">
      <label for="age">Age</label>
      <input id="age" type="text" name="age">
      <label for="height">Height in inches</label>
      <input id="height" type="text" name="height">
      <button type="submit">Show me what I can ride!</butto
n>
    </form>

    <div id="swings" class="hidden">
      <h1>You can ride the Swings.</h1>
      <p>To ride the Swings you need to be at least 6 years
old.</p>
    </div>
    <div id="coaster" class="hidden">
      <h1>You can ride the Roller Coaster.</h1>
      <p>To ride the Roller Coaster you need to be either 1
2 years old or 48 or more inches tall.</p>
    </div>
    <div id="tower" class="hidden">
      <h1>You can ride the Tower of Doom.</h1>
      <p>To ride the Tower of Doom you need to be at least
12 years old and 60 or more inches tall.</p>
```

```
    </div>
    <div id="sorry" class="hidden">
      <h1>You are too young to ride! Sorry!</h1>
      <p>Come back when you are 6 years old.</p>
    </div>
  </div>
</body>
</html>
```

js/scripts.js

```
// User Interface Logic

function hideResults() {
  document.getElementById("swings").setAttribute("class",
"hidden");
  document.getElementById("coaster").setAttribute("class",
"hidden");
  document.getElementById("tower").setAttribute("class", "h
idden");
  document.getElementById("sorry").setAttribute("class", "h
idden");
}

window.onload = function() {
  document.querySelector("form").onsubmit = function(event)
{
    event.preventDefault();
    hideResults();
    const age = parseInt(document.querySelector("input#ag
e").value);
    const height = parseInt(document.querySelector("input#h
eight").value);
  };
};
```

This initial setup should look somewhat familiar to previous ones: we set up an `onsubmit` event handler for our form and we gather the two values for `age` and `height`; we also make sure to prevent the default behavior of a form submission.

Note that we also have a separate function called `hideResults();` that we use to hide our results. We call this function in the `onsubmit` event handler in order to clear results every time that the form is submitted so that the user can submit it again and again and always see new results.

Adding the First Condition — Tower of Doom

We're now ready to put our `age` and `height` variables through an `if` statement. Let's add the next bit of code. Take note that the following code snippet only highlights the `onsubmit` event handler where we'll put our `if` statement.

js/scripts.js

```
...
  // Note: we're only looking at the onsubmit event handler
now!
  document.querySelector("form").onsubmit = function(event)
{
    event.preventDefault();
    hideResults();
    const age = parseInt(document.querySelector("input#ag
e").value);
    const height = parseInt(document.querySelector("input#h
eight").value);

    // we're evaluating the Tower of Doom ride first
    if (age >= 12 && height >= 60) {
      document.getElementById("swings").removeAttribute("cl
ass");
      document.getElementById("coaster").removeAttribute("c
lass");
      document.getElementById("tower").removeAttribute("cla
ss");
    }
  };
...
```

With our first `if` statement, we've used the logical **and** operator, `&&`. The expression `age >= 12 && height >= 60` only evaluates to `true` if both conditions evaluate to `true`. In other words, a user's age needs to be 12 years or older AND their height needs to be 60 inches or more.

You may be wondering why we started with the Tower of Doom condition first. It seems like it would be easier to check if a user is older than 6 to let them know they can ride on the Swings. However, when it comes to handling multiple types of conditions, you always want to start with the condition that's the most specific and complex. Since we need two conditions to be true to ride the Tower of Doom, we need to start with this ride first.

Adding the Second Condition — Roller Coaster

So what's next? Based on what we learned about handling our conditions in the order of most complex/specific to least, we should handle the Roller Coaster next. Let's add an `else if` to our branching. Again, the following code snippet only highlights the `onsubmit` event handler.

js/scripts.js

```
...
  // Note: we're only looking at the onsubmit event handler
now!
  document.querySelector("form").onsubmit = function(event)
{
    event.preventDefault();
    hideResults();
    const age = parseInt(document.querySelector("input#ag
e").value);
    const height = parseInt(document.querySelector("input#h
eight").value);

    if (age >= 12 && height >= 60) {
      document.getElementById("swings").removeAttribute("cl
ass");
      document.getElementById("coaster").removeAttribute("c
lass");
      document.getElementById("tower").removeAttribute("cla
ss");
      // Next, we're evaluating the Roller Coaster ride.
    } else if (age >= 12 || height >= 48) {
      document.getElementById("swings").removeAttribute("cl
ass");
      document.getElementById("coaster").removeAttribute("c
lass");
    }
  };
...
```

With our `else if` statement, we've used the `||` logical **or** operator. We use this operator when only one of two (or more) conditions needs to be true. In this case, a user who is 11 years old and 49 inches tall can ride the roller coaster, and so can a user who is 12 years old and 45 inches tall.

Adding the Last Two Conditions

The next condition is for the Swings ride — we need to make sure the user is at least 6 years old. Let's add code for this condition now. While we're at it, let's add the `else` statement that will handle all other conditions, which is when a user is not 6 years old and can't ride anything. In this case, we'll give them a message asking them to come back when they are years old!

js/scripts.js

```
...
  // Note: we're only looking at the onsubmit event handler
now!
  document.querySelector("form").onsubmit = function(event)
{
    hideResults();
    event.preventDefault();
    const age = parseInt(document.querySelector("input#ag
e").value);
    const height = parseInt(document.querySelector("input#h
eight").value);

    if (age >= 12 && height >= 60) {
      document.getElementById("swings").removeAttribute("cl
ass");
      document.getElementById("coaster").removeAttribute("c
lass");
      document.getElementById("tower").removeAttribute("cla
ss");
    } else if (age >= 12 || height >= 48) {
      document.getElementById("swings").removeAttribute("cl
ass");
      document.getElementById("coaster").removeAttribute("c
lass");
      // Next, we're evaluating the Swings ride.
    } else if (age >= 6) {
      document.getElementById("swings").removeAttribute("cl
ass");
      // Finally, we have our "catch-all" for when a user i
s less than 6 years old.
    } else {
      document.getElementById("sorry").removeAttribute("cla
ss");
    }
  };
...
```

The last `else if` and `else` statements at this point should be straight forward.

Remember that when you need to create branching to handle multiple conditions: start by handling the more complex and specific conditions first. When figuring out the order of conditions, take your time and talk about it with a pair. It's okay if you need to refactor and try out a different order mid-way through your branching.

Adding Error Handling and Exploring Truthy and Falsey

There's two additional pieces of functionality we can add to make this website polished and complete. The first is error handling. What happens if a user leaves our form blank? We can use branching to help us out!

Let's add some HTML for an error message:

amusement-park.html

```
...
  <form id="userInfo">
    <!-- There's a new line below this comment! -->
    <p id="error-message" class="hidden">Please enter an ag
e and height in whole numbers.</p>
    <label for="age">Age</label>
    <input id="age" type="text">
    <label for="height">Height in inches</label>
    <input id="height" type="text">
    <button type="submit">Show me what I can ride!</button>
  </form>
...
```

In the above code snippet we've added an error message in a `<p>` tag to our HTML. We've hidden it, because we only want to display it if there is an error.

Let's add some styles to make the text bold and red to look like an error message:

css/styles.css

```
...

#error-message {
  font-weight: 600;
  color: red;
}
```

We'll also want to make sure that our error message gets hidden every time the user submits the form, so let's adapt the `hideResults()` function to also hide the error message. Note, we've omitted some code that we haven't updated with an ellipsis `...`

```
// User Interface Logic

// New function name!
function hideResultsAndError() {
  // New line of code to hide the error message.
  document.getElementById("error-message").setAttribute("cl
ass", "hidden");
  document.getElementById("swings").setAttribute("class",
"hidden");
  document.getElementById("coaster").setAttribute("class",
"hidden");
  document.getElementById("tower").setAttribute("class", "h
idden");
  document.getElementById("sorry").setAttribute("class", "h
idden");
}

window.onload = function() {
  document.querySelector("form").onsubmit = function(event)
{
    // updated to use new function name!
    hideResultsAndError();
    ...
  };
};
```

Now we're ready to add code to handle the error if a user inputs nothing and submits the form. We'll do this with a new `if...else` statement. Let's now just look at the code in the `onsubmit` event handler, and pay close attention to the comments.

js/scripts.js

```
...
  // Note: we're only looking at the onsubmit event handler
now!
  document.querySelector("form").onsubmit = function(event)
{
    hideResultsAndError();
    event.preventDefault();
    const age = parseInt(document.querySelector("input#ag
e").value);
    const height = parseInt(document.querySelector("input#h
eight").value);

    if (age && height) {
      // Our original if statement starts here!
     if (age >= 12 && height >= 60) {
        document.getElementById("swings").removeAttribute
("class");
        document.getElementById("coaster").removeAttribute
("class");
        document.getElementById("tower").removeAttribute("c
lass");
      } else if (age >= 12 || height >= 48) {
        document.getElementById("swings").removeAttribute
("class");
        document.getElementById("coaster").removeAttribute
("class");
      } else if (age >= 6) {
        document.getElementById("swings").removeAttribute
("class");
      } else {
        document.getElementById("sorry").removeAttribute("c
lass");
      }
      // Our original if statement ends here!
    } else {
      document.getElementById("error-message").removeAttrib
ute("class");
    }
  };
...
```

We've wrapped our original `if` statement in a new `if...else`. Here's another look at the new `if` statements we've just added, but separated out from the code around it:

```
if (age && height) {
  // check if user can ride the rides
  // by sending age and height through another if statement
} else {
  // print error message
  // tell user to input age and height correctly
}
```

Does the condition `if (age && height)` strike you as a little funny? Before now, all of the conditions for our `if` statements have evaluated to booleans. This time, we're looking at two variables that are numbers.

If the user doesn't input a value, we'll get an empty string. When we run `parseInt()` on an empty string, we get `NaN` — not a number. Neither `NaN` nor an empty string is `true` or `false`. So what's going on here?

What's happening here is that JavaScript has concepts called **truthy** and **falsey**.

Falsey values include the boolean `false` as well as:

- empty strings
- the number `0`
- the number `NaN`
- `undefined`
- `null`

If JavaScript sees any of these as a branching condition, it will treat them as false. Everything else is truthy. Truthy values include numbers and strings that have a value.

Looking again at `if (age && height)`, this is asking, do the variables `age` and `height` both have a value greater than 0? If not, then we need to display an error message asking that the user input the correct information. If so, then `age` and `height` are considered truthy in JavaScript and the code in the block (within the curly brackets `{ }`) following that condition will run.

Truthy and Falsey Are Caused By Data Type Coercion

Truthy and falsey values exist because of the data type coercion, that is the implicit forced conversion JavaScript automatically performs. When we put expressions into our `if (expression)` or `else if (expression)` statements, these need to evaluate to `true` or `false`. When the expression we put in is something that's not a typical comparison like `2 > 1` or `typeof "phrase" === "string"`, but instead something like an empty string, JavaScript uses data type coercion to evaluate the expression as a boolean.

Let's review another example of passing in a variable set to an empty string:

```
let myVar = "";
if (myVar) {
  // This code will run if JavaScript implictly converts my
Var to the boolean true.
  // This conversion will only happen if myVar is NOT an em
pty string and has some value.
  // myVar will then be considered a "truthy" value.
} else {
  // This code will run if JavaScript implictly converts my
Var to the boolean false.
  // This conversion will only happen if myVar is an empty
string,
  // or any other falsey value like null or undefined.
  // myVar will then be considered a "falsey" value.
}
```

In the above example, JavaScript implicitly converts `myVar` into one of two options:

- The boolean `false`, if `myVar` is an empty string.
- The boolean `true`, if `myVar` has any value inside of the string.

Since `myVar` is set to an empty string, JavaScript will convert this "falsey" value into the boolean `false`.

An Alternative: Using the Logical Not `!` Operator

Let's discuss another operator you should know about. If you want to determine whether something is not true, you can use the logical **not** `!` operator. Let's rewrite our error handling `if` statements to use the logical not operator:

js/scripts.js

```
// with the logical not operator, we check for whether
// age or height do NOT exist, and if they don't exist,
// then we display the error message
if (!age || !height) {
  document.getElementById("error-message").removeAttribute
("class");
// in all other cases, we run our code to see what ride a u
ser can go on
} else {
  if (age >= 12 && height >= 60) {
    document.getElementById("swings").removeAttribute("clas
s");
    document.getElementById("coaster").removeAttribute("cla
ss");
    document.getElementById("tower").removeAttribute("clas
s");
  } else if (age >= 12 || height >= 48) {
    document.getElementById("swings").removeAttribute("clas
s");
    document.getElementById("coaster").removeAttribute("cla
ss");
  } else if (age >= 6) {
    document.getElementById("swings").removeAttribute("clas
s");
  } else {
    document.getElementById("sorry").removeAttribute("clas
s");
  }
}
```

Like the comments describe, in our first `if` statement, we're checking if age or height do NOT exist: `(!age || !height)`. If either variable is `0` or `NaN`, meaning they do not exist and they are considered falsey, then our code will show the error message. Otherwise, our code will execute the else statement with the nested branching to see what rides the user can go on.

We have switched to the logical or `||` operator because we want to show our error message if one or the other form field is left blank. Or, in other words if `age` or `height` do not exist and are falsey.

Next Steps and Completed Code

With error handling in place, what else can we do with our website? It's always helpful to think of additional polish we can add, or more cases we can handle in our branching. For example, we could:

- Add a button that clears form values.
- Refactor our code to handle unlikely cases. For example, if we enter an age of `3` and a height of `60`, we'll be told that we can ride the roller coaster and swings. However, you must be at least 6 years old to ride any ride at the park! While it is unlikely that there will be a 3 year old person who is 5 feet tall (60 inches), this is a possible case that our form should handle. How would you update the `if` statement to handle this case?

To see the completed code for this project, visit the cheat sheet.

Previous (/introduction-to-programming/javascript-and-web-browsers/practice-branching)
Next (/introduction-to-programming/javascript-and-web-browsers/practice-more-branching)

Lesson 61 of 75
Last updated March 24, 2023

disable dark mode

(http://www.epicodus.com)