~CoolWater~

Water charity so no one goes without water

Founder: Alyssa Ayala
Giving water to those in need since November 2022

Table of Contents

Business Information

If water is so important, then why are people still going without it? Despite the growth in the number of developed countries and the improvements of developing countries, water insecurity is sadly still a common problem today. Water is essential for life and how our bodies function, and if people are going without it for long periods of time, their quality of life will undoubtedly decrease as more people get sick easily. The problem is that people in poverty may not be able to afford clean water at all. If they get any water, the water quality may not be fresh and contain harmful bacteria, hence doing more harm than good. Financial issues should not get in the way of getting our basic needs just to survive.

CoolWater is a non-profit water charity that aims to supply clean water to those who may not be able to afford it. With money donated by people around the world, even just $1 helps get fresh water to someone in need. This money is used to buy the best quality water out there to give to anyone who needs it. We travel to places all around the world to get to anyone we need to. Once people get fresh water, they will undoubtedly feel better and live long and healthy lives if they keep getting it.

We only get our water from brands that have a good reputation and are known for always having the cleanest water. They are tested for any poor working conditions, bacteria, and filtration systems that make up their water. We also never accept any evidence of tap water, as this type of water is not fresh and is more likely to contain bacteria and hence get people sick.

Donating is easy; you only need to have your credit or debit card handy. You can donate and hence save lives right on my website in just minutes. Filling out a short form is all it takes to ensure that at least one person does not go without clean water.

Website Information

This website was created with the help of the “50 Projects in 50 Days” course on *Udemy*, which gives instructions on how to code HTML, CSS, and JavaScript that makes a website stand out the most. In total, I used five projects from that course to create this website.

This website is also run by Node.js, with installed modules express, nedb, and body-parser. To run this website to its full functionality, this website must be run on a local server. This server is under the **app.js** file. Do this command in the same directory as this file to run the website: **node app.js**. It’ll be at the address **127.0.0.1:3000**, where you’ll be initially taken to the home page. Do Ctrl + C to kill the connection. The express module is used to create and use the website altogether, nedb is the database for donator information collected from the form on the Donator page, and body-parser is used to get responses from the form using the POST method.

Every Page: Every page has a loading screen that appears before the actual page does. This is an animation of triangles spinning around that covers the content of the page while the page’s real content is hidden. After that time has passed, the animation will disappear and be replaced with the real content of the page. The animation itself was in the Udemy project “Kinetic CSS Loader” in the Udemy course, but not the JavaScript part. Therefore, my own JavaScript is based on this part of the website. While I used the same style and animation as the course, I made my own timer to show and take away parts of the page.

Home: The home page was styled using the “Double Vertical Slider” project in the Udemy course. There is a large picture on the right, while there is a description relating to it on the left. You can get more pictures and descriptions by clicking on the up and down arrows in the middle of these containers. The idea here is to describe my charity business in an aesthetic and minimalist way. The background of the description matches the overall color of the picture. For each of these sets of containers, there is a fact about what my charity business does with an example of that action being done.

Statistics: This page was styled using the “Incrementing Counter” project in the Udemy course. When the page loads, the numbers rapidly increase until it gets to the desired value. I have three of these increasing values for the number of donated water bottles, families fed, and student fed.This is done to give my website visitors an idea of how much my charity business has helped other people thanks to other people’s donations, to encourage them to donate more money thus far to help save more lives and ensure someone gets water to drink today.

How Much to Drink: This page was styled using the “Drink Water” project in the Udemy course. This project is actually what inspired this charity business because I was thinking about how important it actually is to drink water but also forget to drink enough water at the same time. This page tells you how much water you have left to drink during the day based on how much water you have drunk so far. This is based on needing to drink at least two liters of water a day. As a

water charity business, we are committed to making sure everyone gets as much water as they need.

Donate: This page was styled using the “Content Placeholder” project from the Udemy course. This page is actually a combined two pages; the “submitted” part is what contains this project. You fill out a form with information about yourself and your payment method to donate to my charity business. After that’s done, you’ll be taken to the content placeholder that thanks you for donating, along with a button that takes you back to do another donation if you wish to do so. This page was made to be a convenient way to donate quickly and effortlessly from wherever you are. In other words, it’s a quick and effortless way to ensure that one person gets the water they need today.

Salient Points

The salient points below should be considered since these are the most important parts of my code that ultimately make my webpage act the way it does.

#1: I had no problems getting the node server to work, but the CSS or JavaScript code would not work; only the unmodified HTML code would show up. I knew that I needed a way to make the CSS and JavaScript code for my HTML files to show up on the website. Solving this problem involved changing what modules I used overall. Since we are allowed to use modules like express, I decided to take advantage and put it into my project. Express is good for this type of thing because referring to CSS and JavaScript files as "static files," we just need to locate where they are in the directory, like this:

```
// Static files
app.use(express.static('CoolWater'));
app.use('/css', express.static(__dirname + 'CoolWater/css'));
app.use('/javascript', express.static(__dirname +
'CoolWater/javascript'));
```

CoolWater (1st line) is where all of the files are, and then I refer to the CSS and JavaScript folders in the last two lines.

#2: I thought that an animated loading screen would look nice for my website, which is why I opted to use the "CSS Kinetic Loader" project from the Udemy course. However, when I ran it for the first time, I could see the triangles in the correct places, but not the animation. I was sure that I had typed all of the code correctly, so I couldn't see what the problem was. But it was actually quite simple. The way the code was set up, specifically the kinetic class, had its animation tags set to have a delay before it actually begins:

`animation: rotateA 2s linear infinite 0.5s;` (in kinetic after and before)

`animation: rotateB 2s linear infinite;` (in kinetic before)

In this case, there is a 0.5 second delay. So what I had to actually change was the JavaScript that I put in:

```
setTimeout(showContent,2000);
```

This is a timer to set a delay before something else happens. In this case, after 2 seconds, this function should run, hiding the loading screen and showing the actual content:

```
function showContent() {
     load.style = "display:none";
     container.style = "display:relative";
}
```

Increasing this to 2 seconds gives the website more time to make the animation start, and then I'm able to see it.

#3: I was also certain that I had typed the code correctly for the home page. The vertical sliders and buttons looked exactly as I intended them to look. When I clicked on either of the arrow buttons, the images on the right side would move as intended. However, the left side of the page would not move at all. I knew that every class and such was organized correctly, so I

couldn't figure out what the problem was. It was actually a typo in my JavaScript code that was causing an issue all along. I had forgotten to put a quotation mark for one of my query selectors. They can be difficult for me to notice at times since these errors are so minor and unnoticeable, especially since JavaScript is not compiled to check for any errors before running. I changed this line:

```
const slideLeft = document.querySelector('.left-slide);
```

To this:

```
const slideLeft = document.querySelector('.left-slide');
```

Without the quotation mark, the element was not actually being recognized since the element was not being identified. Once that quotation mark was there, the JavaScript code knew what to refer to, and the left side started moving as intended again.

#4: Referring to the second point about how the container and loading screen are shown and hidden, how the contents are shown are actually extremely important. This was especially apparent for the Statistics page of my website. Nothing was actually wrong with the project itself; every statistic was being incremented and showing data just fine. The page itself is supposed to show each statistic in a vertical line, but when I had initially tried to display it again after hiding the loading screen, they would all be in a vertical row instead. This is because pages should be shown with a relative property if you wish to get the pages exactly as they were before they were hidden, like this:

```
container.style = "display:relative";
```

I had tried other properties such as static, block, and inline, but all of them gave me the same vertical row result. Therefore, it is a lesson learned that a relative property is extremely important and the most useful property for me.

#5: It was difficult for me to figure this out at first, but I decided that I wanted to store inputted data using a database, specifically a nedb one. This is a separate module in node.js that is specifically designed to put information into a database. It happens in this way whenever a POST method request occurs:

```
// Collect data from donator form
app.post('/submitted', (req, res) => {
      res.sendFile(__dirname + '/CoolWater/submitted.html');
      database.insert(req.body);
});
```

I knew that I wanted to use this module, but I didn't know exactly how to get information from the form. The code above reflects how I made my form have a POST method with the specific action called '/submitted'. This is when I started to experience what form methods could do for a developer, especially in Node.js.

#6: Even when I figured out the POST method, however, I could not figure out why req.body kept coming up empty for me, which I verified by temporarily using console.log. Even when I filled in every single field of the donation form, it was always empty. It turned out that not only were ids important, but I also had to name the inputs in the form, like this:

```
<input name="name1" type="text" id="name1" required></br>
```

```
<input name="email" type="email" id="email" required></br>
<input name="amount" type="number" id="amount" required></br>
<input name="card" type="number" id="card" required></br>
<input name="expirationDate" type="date" id="expirationDate"
required></br>
<input name="cvc" type="number" id="cvc" required></br>
<input name="address" type="text" id="address" required></br>
```

Before, only the types and ids were there. But now for each name that is in the form, it does into request.body.

#7: I knew that this time around, I wanted a different action on what would happen when a form was submitted to make my website stand out the most. Since I now use a POST method that could therefore take me to a different page, I decided to indeed make another page that the website would go to once the donor form was submitted. I had originally wanted to use something called toast notifications from the “Toast Notifications” project from the Udemy course, but this required me to stay on the same page since my action was javascript:submit(), which looked like this:

```
<script>
      function submit() {
            document.getElementById(‘button’).click();
      }
</script>
```

Since this was not what I was going for, I decided to use something called content placeholders from the “Content Placeholders” project from the Udemy course. Now there is no JavaScript running in the HTML page since this is handled by the node.js file:

```
// Collect data from donator form
app.post('/submitted', (req, res) => {
      res.sendFile(__dirname + '/CoolWater/submitted.html');
      database.insert(req.body);
});
```

You would go to the submitted page instead of staying on the same page, which contains the content placeholder project. At the same time, the form data is placed into the database. So this code not only does it job, but it makes the page stand out.