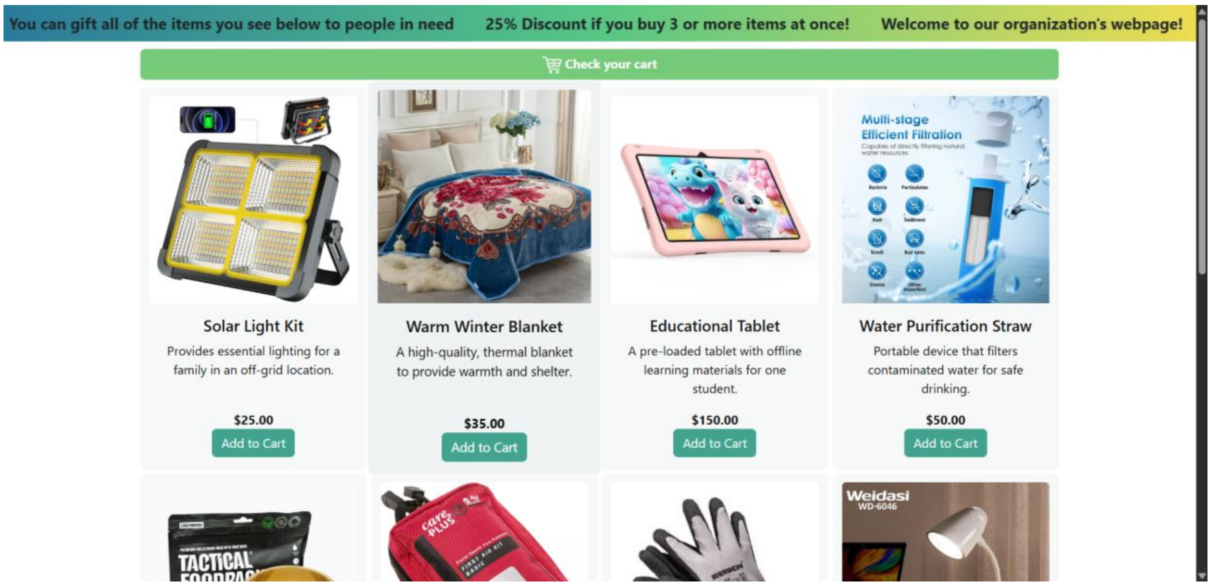
Documentation

Contents

Picture of the website



You can gift all of the items you see below to people in need 25% Discount if you buy 3 or more items at once! Welcome to our organization's webpage!

Check your cart

Solar Light Kit

Provides essential lighting for a family in an off-grid location.

$25.00

Add to Cart

Warm Winter Blanket

A high-quality, thermal blanket to provide warmth and shelter.

$35.00

Add to Cart

Educational Tablet

A pre-loaded tablet with offline learning materials for one student.

$150.00

Add to Cart

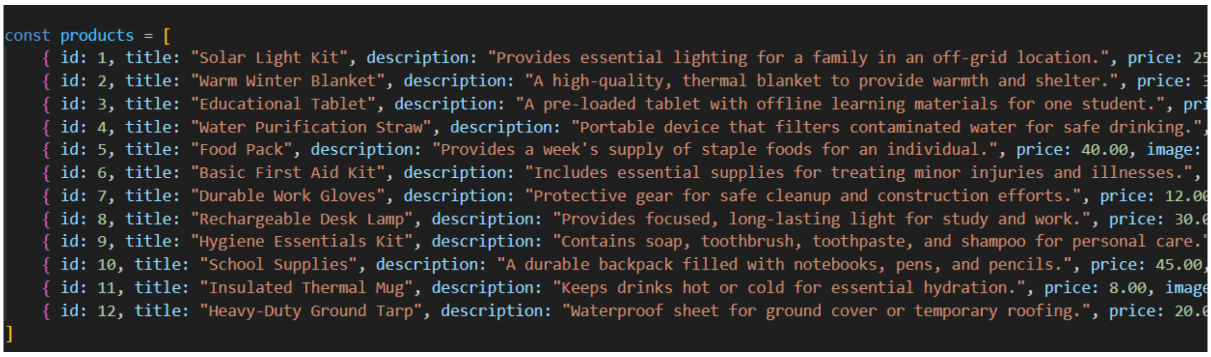
Water Purification Straw

Portable device that filters contaminated water for safe drinking.

$50.00

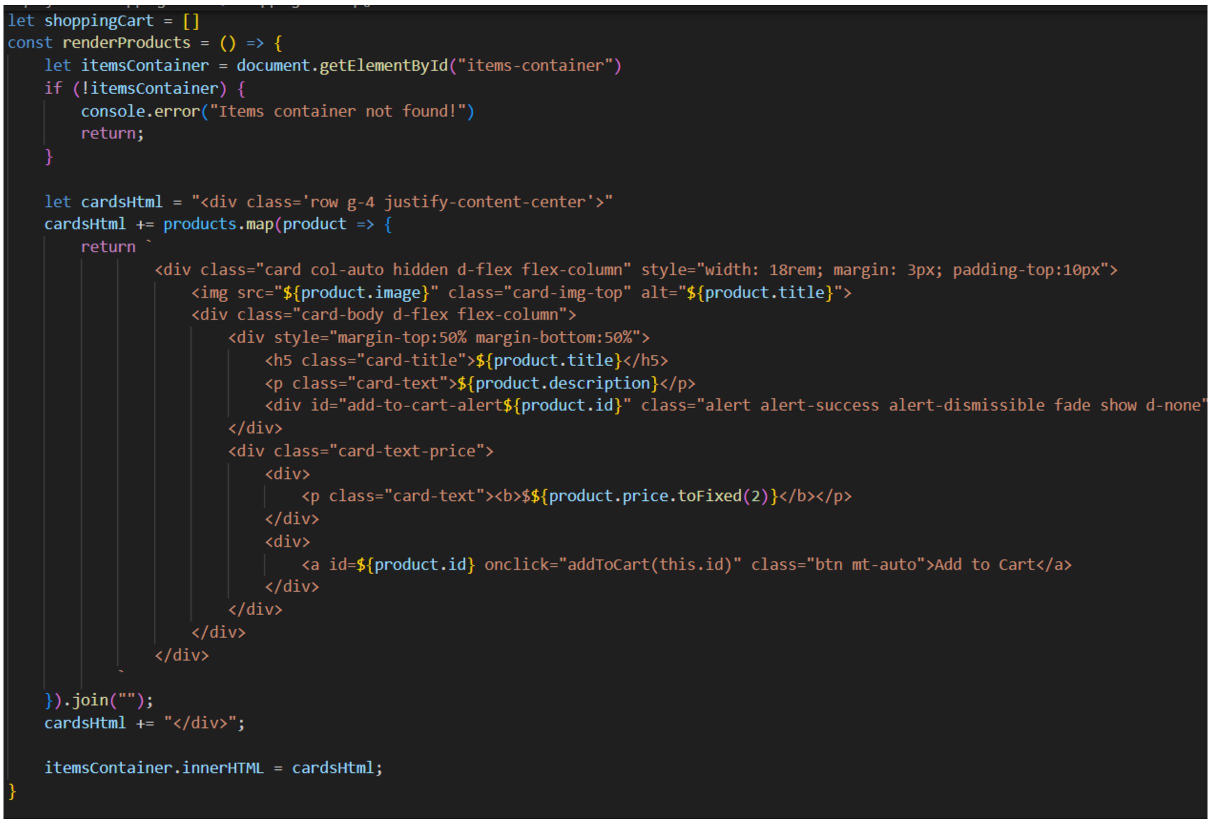
Add to Cart

Product rendering



```
const products = [
    { id: 1, title: "Solar Light Kit", description: "Provides essential lighting for a family in an off-grid location.", price: 25
    { id: 2, title: "Warm Winter Blanket", description: "A high-quality, thermal blanket to provide warmth and shelter.", price: 3
    { id: 3, title: "Educational Tablet", description: "A pre-loaded tablet with offline learning materials for one student.", pri
    { id: 4, title: "Water Purification Straw", description: "Portable device that filters contaminated water for safe drinking.",
    { id: 5, title: "Food Pack", description: "Provides a week's supply of staple foods for an individual.", price: 40.00, image:
    { id: 6, title: "Basic First Aid Kit", description: "Includes essential supplies for treating minor injuries and illnesses.",
    { id: 7, title: "Durable Work Gloves", description: "Protective gear for safe cleanup and construction efforts.", price: 12.00
    { id: 8, title: "Rechargeable Desk Lamp", description: "Provides focused, long-lasting light for study and work.", price: 30.0
    { id: 9, title: "Hygiene Essentials Kit", description: "Contains soap, toothbrush, toothpaste, and shampoo for personal care."
    { id: 10, title: "School Supplies", description: "A durable backpack filled with notebooks, pens, and pencils.", price: 45.00,
    { id: 11, title: "Insulated Thermal Mug", description: "Keeps drinks hot or cold for essential hydration.", price: 8.00, image
    { id: 12, title: "Heavy-Duty Ground Tarp", description: "Waterproof sheet for ground cover or temporary roofing.", price: 20.0
]
```

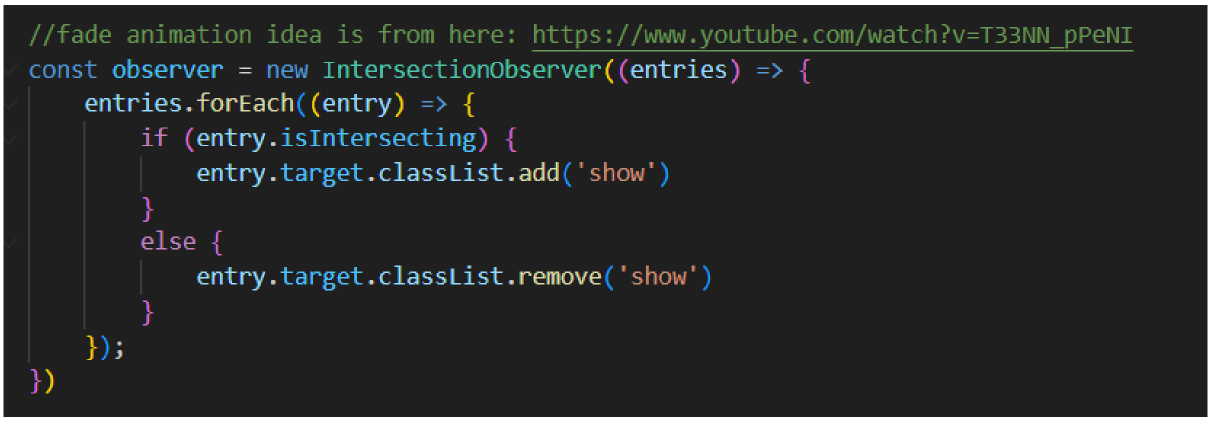
First of all we have the default product array which contains all the products of the website.



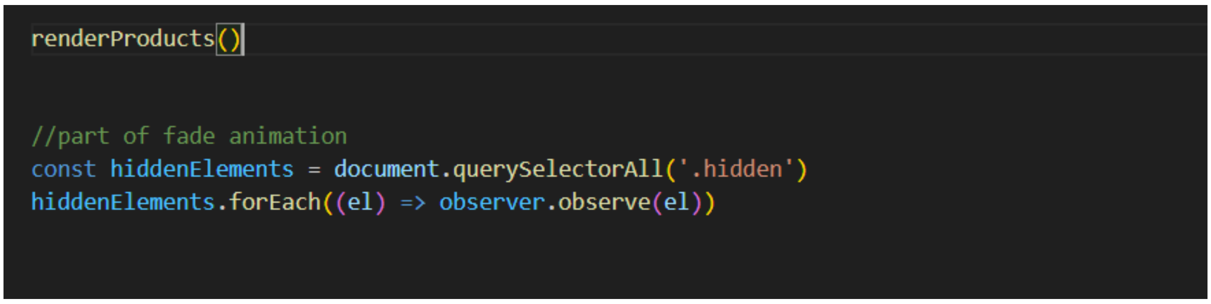
```
let shoppingCart = []
const renderProducts = () => {
    let itemsContainer = document.getElementById("items-container")
    if (!itemsContainer) {
        console.error("Items container not found!")
        return;
    }

    let cardsHtml = "<div class='row g-4 justify-content-center'>"
    cardsHtml += products.map(product => {
        return `
                <div class="card col-auto hidden d-flex flex-column" style="width: 18rem; margin: 3px; padding-top:10px">
                    <img src="${product.image}" class="card-img-top" alt="${product.title}">
                    <div class="card-body d-flex flex-column">
                        <div style="margin-top:50% margin-bottom:50%">
                            <h5 class="card-title">${product.title}</h5>
                            <p class="card-text">${product.description}</p>
                            <div id="add-to-cart-alert${product.id}" class="alert alert-success alert-dismissible fade show d-none"
                        </div>
                        <div class="card-text-price">
                            <div>
                                <p class="card-text"><b>$${product.price.toFixed(2)}</b></p>
                            </div>
                            <div>
                                <a id=${product.id} onclick="addToCart(this.id)" class="btn mt-auto">Add to Cart</a>
                            </div>
                        </div>
                    </div>
                </div>
            `
    }).join("");
    cardsHtml += "</div>";

    itemsContainer.innerHTML = cardsHtml;
}
```

Then we have a function called render products which creates a container div for all the cards in the items-container div. After that we go through the products list with map and for every product in the product list we return a Bootstrap Card which will contain the image,title,description,id and price of the product and we join all of the cards together which we put into the cardsHtml div which we put into the items-container div. We call this render products function once every time we want to view the website.

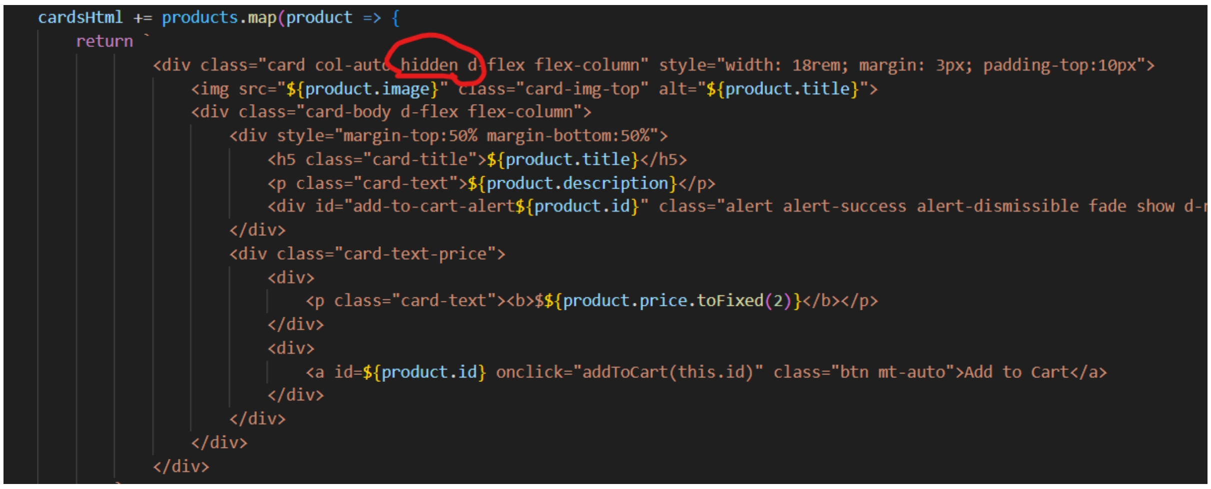
Fade animation



```
//fade animation idea is from here: https://www.youtube.com/watch?v=T33NN_pPeNI
const observer = new IntersectionObserver((entries) => {
    entries.forEach((entry) => {
        if (entry.isIntersecting) {
            entry.target.classList.add('show')
        }
        else {
            entry.target.classList.remove('show')
        }
    });
})
```



```
renderProducts()


//part of fade animation
const hiddenElements = document.querySelectorAll('.hidden')
hiddenElements.forEach((el) => observer.observe(el))
```

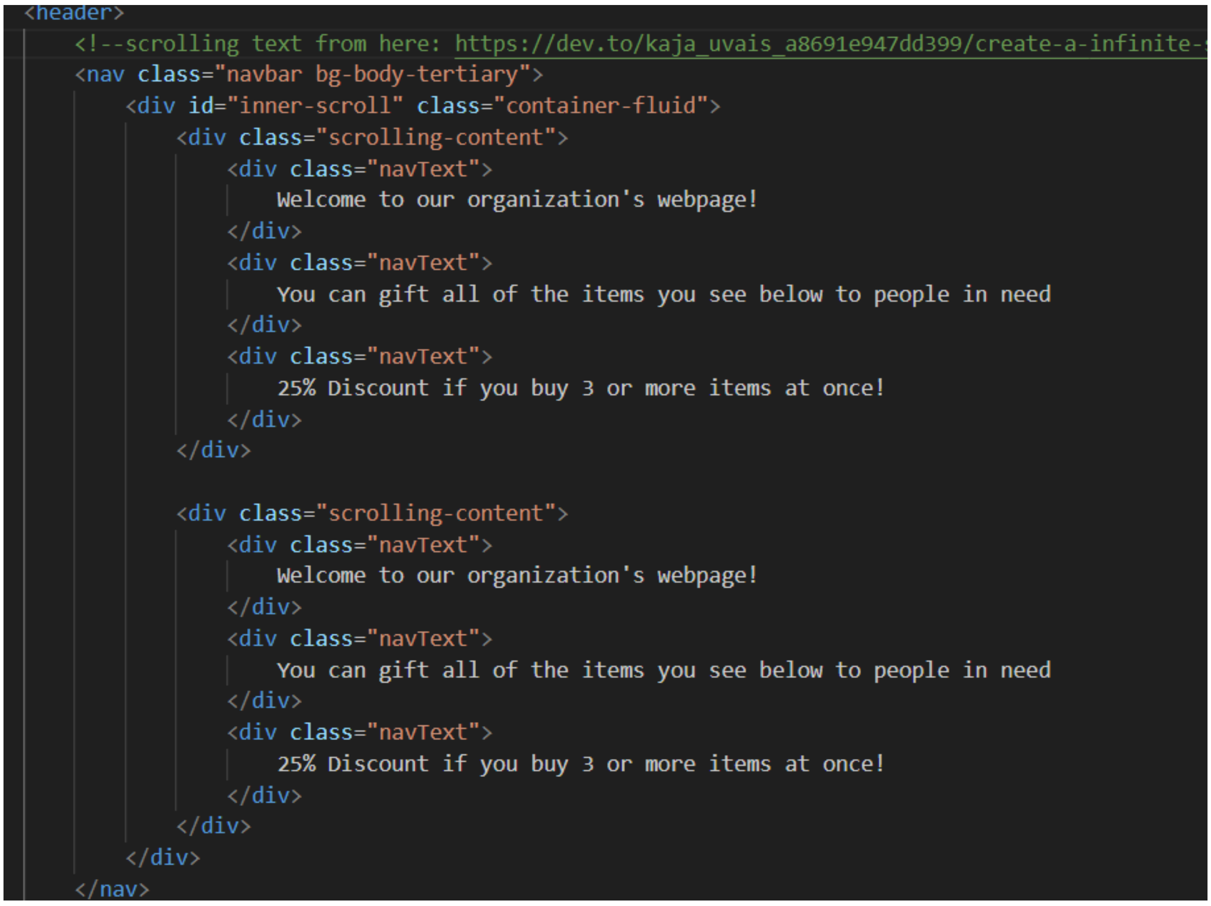


```
cardsHtml += products.map(product => {
    return `
            <div class="card col-auto hidden d-flex flex-column" style="width: 18rem; margin: 3px; padding-top:10px">
                <img src="${product.image}" class="card-img-top" alt="${product.title}">
                <div class="card-body d-flex flex-column">
                    <div style="margin-top:50% margin-bottom:50%">
                        <h5 class="card-title">${product.title}</h5>
                        <p class="card-text">${product.description}</p>
                        <div id="add-to-cart-alert${product.id}" class="alert alert-success alert-dismissible fade show d-
                    </div>
                    <div class="card-text-price">
                        <div>
                            <p class="card-text"><b>$${product.price.toFixed(2)}</b></p>
                        </div>
                        <div>
                            <a id=${product.id} onclick="addToCart(this.id)" class="btn mt-auto">Add to Cart</a>
                        </div>
                    </div>
                </div>
            </div>
```

We have an intersection observer at the start of our javascript code which observes all of the elements that have the hidden class on them. Every time that an element which has the hidden class would become visible on our webpage the observer detects it and adds the show class into it's class list so the item becomes visible again. Also every time an item would be out of our scope of sight it removes the show class from it's class list so it becomes invisible again, so with a smooth transition we made fading cards by applying the hidden class to every card that we want rendered. It is VERY important that we RENDER the products before calling the observer so there actually is something to observe and it works!

Bootstrap components

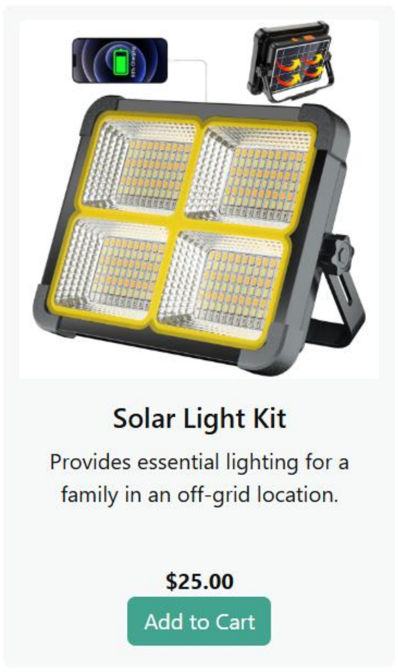
Nav-bar

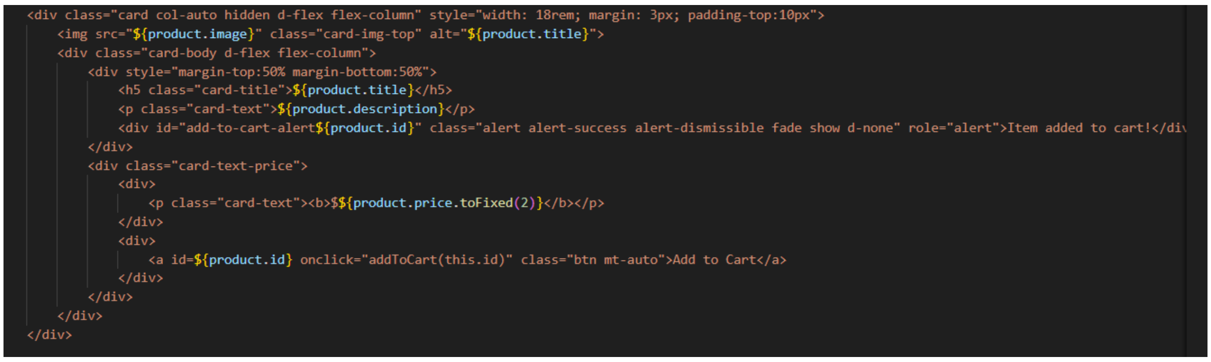


```html
<header>
    <!--scrolling text from here: https://dev.to/kaja_uvais_a8691e947dd399/create-a-infinite-
    <nav class="navbar bg-body-tertiary">
        <div id="inner-scroll" class="container-fluid">
            <div class="scrolling-content">
                <div class="navText">
                    Welcome to our organization's webpage!
                </div>
                <div class="navText">
                    You can gift all of the items you see below to people in need
                </div>
                <div class="navText">
                    25% Discount if you buy 3 or more items at once!
                </div>
            </div>

            <div class="scrolling-content">
                <div class="navText">
                    Welcome to our organization's webpage!
                </div>
                <div class="navText">
                    You can gift all of the items you see below to people in need
                </div>
                <div class="navText">
                    25% Discount if you buy 3 or more items at once!
                </div>
            </div>
        </div>
    </nav>
```

The default bootstrap navigation bar was used for this webpage but only to store the infinitely scrolling div components.

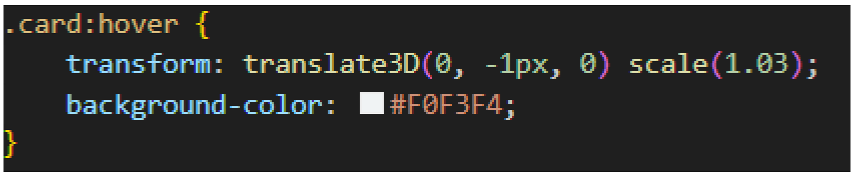
Cards


Solar Light Kit
Provides essential lighting for a family in an off-grid location.
$25.00
Add to Cart



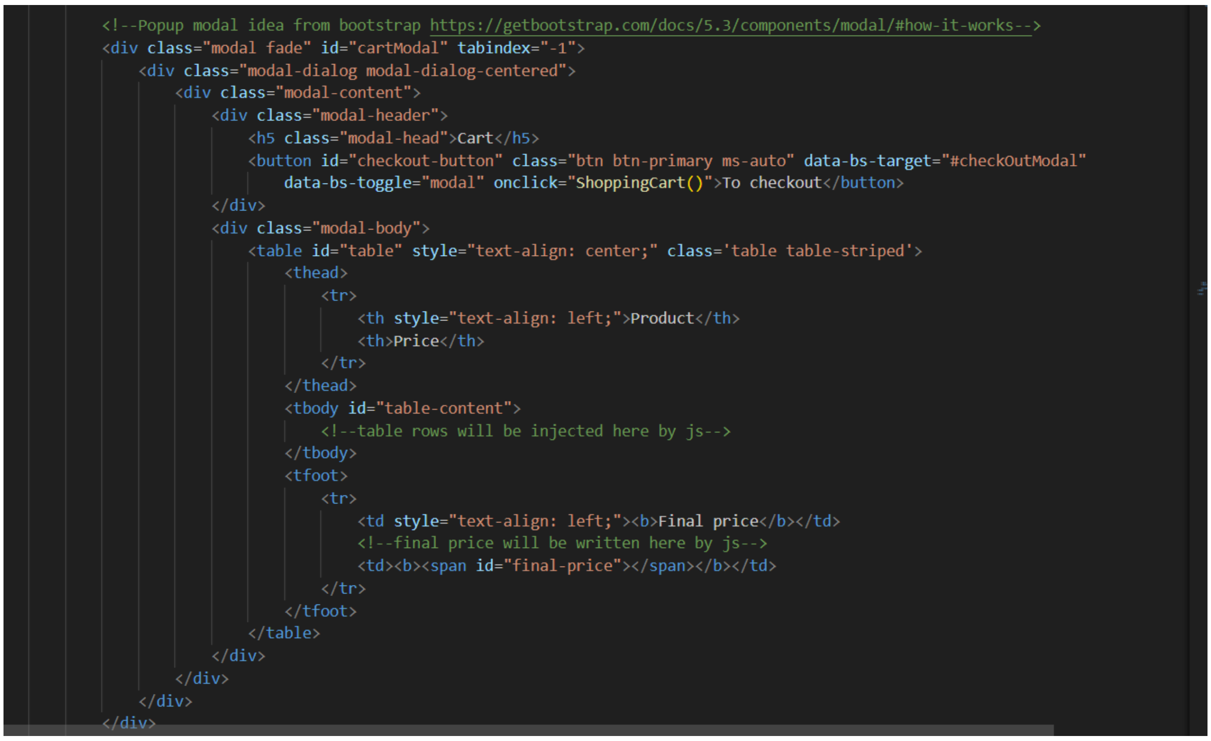
```html
<div class="card col-auto hidden d-flex flex-column" style="width: 18rem; margin: 3px; padding-top:10px">
    <img src="${product.image}" class="card-img-top" alt="${product.title}">
    <div class="card-body d-flex flex-column">
        <div style="margin-top:50% margin-bottom:50%">
            <h5 class="card-title">${product.title}</h5>
            <p class="card-text">${product.description}</p>
            <div id="add-to-cart-alert${product.id}" class="alert alert-success alert-dismissible fade show d-none" role="alert">Item added to cart!</di
        </div>
        <div class="card-text-price">
            <div>
                <p class="card-text"><b>$${product.price.toFixed(2)}</b></p>
            </div>
            <div>
                <a id=${product.id} onclick="addToCart(this.id)" class="btn mt-auto">Add to Cart</a>
            </div>
        </div>
    </div>
</div>
```

Default bootstrap card component with transform-translate3D hover effect on it so if the cursor is on the card it gets bigger

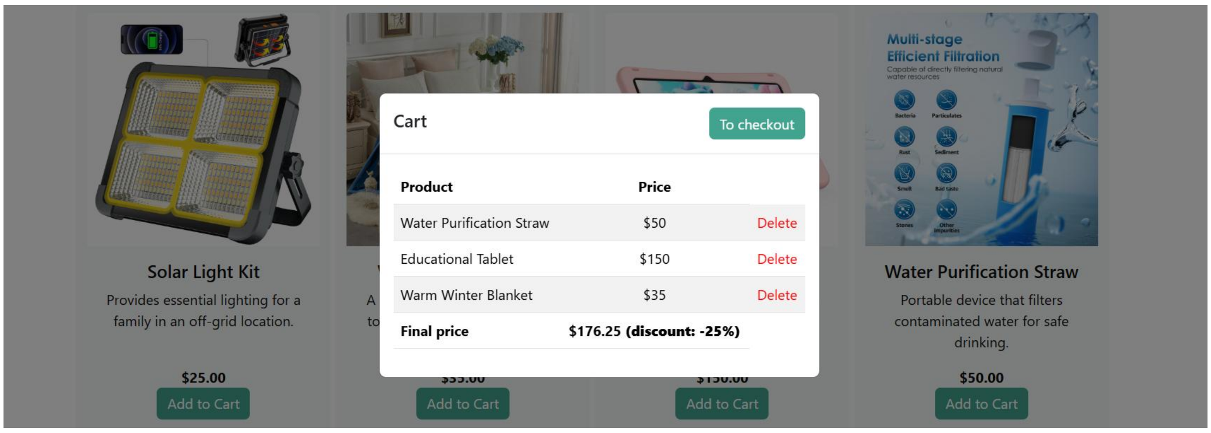


```css
.card:hover {
    transform: translate3D(0, -1px, 0) scale(1.03);
    background-color: ⬜#F0F3F4;
}
```

Shopping Cart Modal



```html
<!--Popup modal idea from bootstrap https://getbootstrap.com/docs/5.3/components/modal/#how-it-works-->
<div class="modal fade" id="cartModal" tabindex="-1">
    <div class="modal-dialog modal-dialog-centered">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-head">Cart</h5>
                <button id="checkout-button" class="btn btn-primary ms-auto" data-bs-target="#checkOutModal"
                    data-bs-toggle="modal" onclick="ShoppingCart()">To checkout</button>
            </div>
            <div class="modal-body">
                <table id="table" style="text-align: center;" class='table table-striped'>
                    <thead>
                        <tr>
                            <th style="text-align: left;">Product</th>
                            <th>Price</th>
                        </tr>
                    </thead>
                    <tbody id="table-content">
                        <!--table rows will be injected here by js-->
                    </tbody>
                    <tfoot>
                        <tr>
                            <td style="text-align: left;"><b>Final price</b></td>
                            <!--final price will be written here by js-->
                            <td><b><span id="final-price"></span></b></td>
                        </tr>
                    </tfoot>
                </table>
            </div>
        </div>
    </div>
</div>
```



Solar Light Kit

Provides essential lighting for a family in an off-grid location.

$25.00

Add to Cart

Water Purification Straw

Portable device that filters contaminated water for safe drinking.

$50.00

Add to Cart

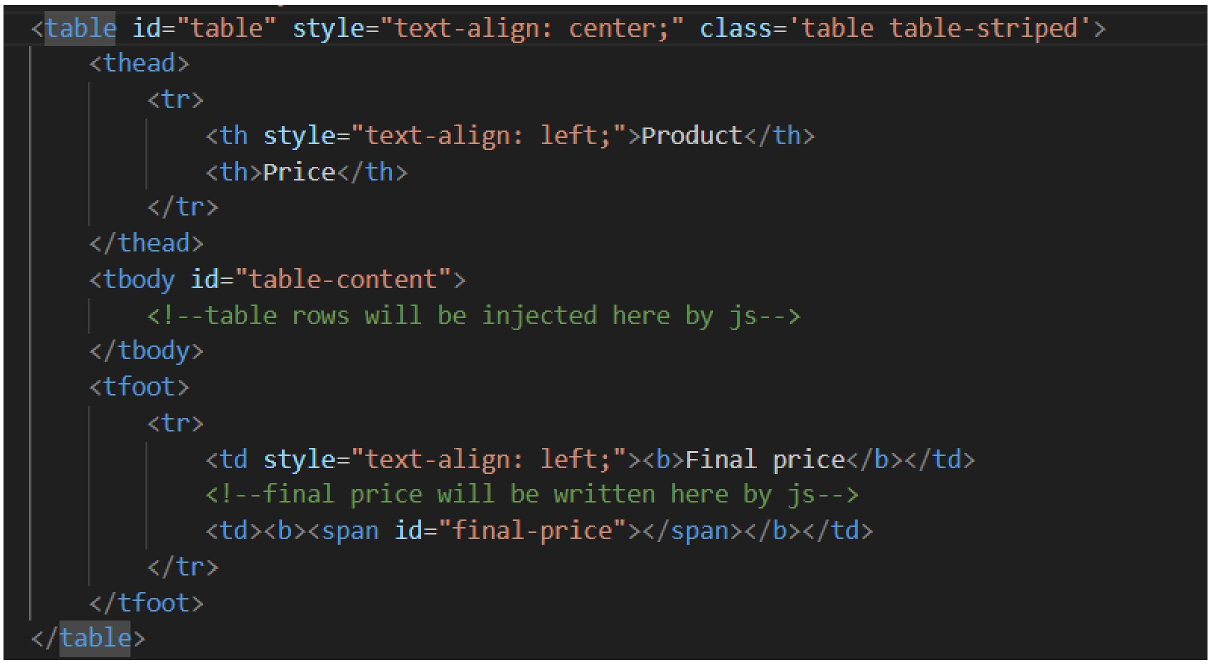
Cart

To checkout

Product	Price	
Water Purification Straw	$50	Delete
Educational Tablet	$150	Delete
Warm Winter Blanket	$35	Delete
Final price	**$176.25 (discount: -25%)**	

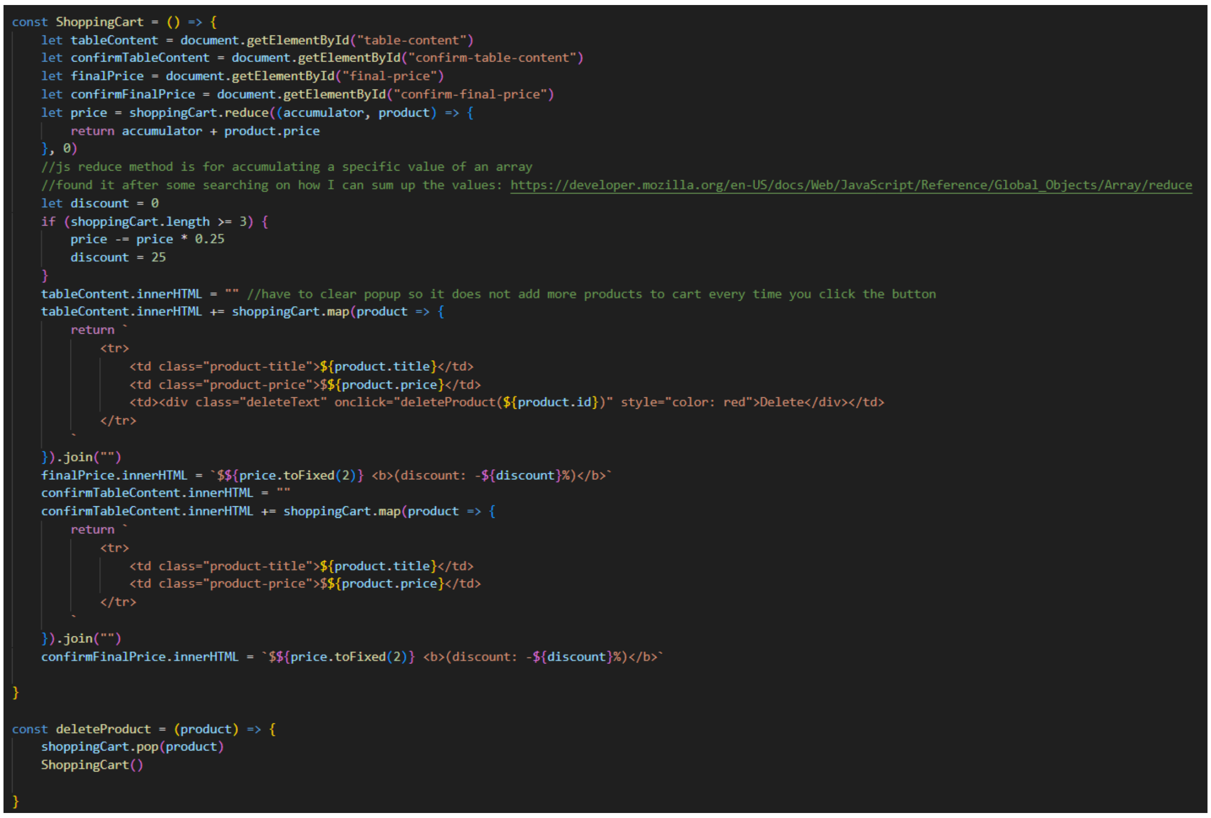
Shopping cart Javascript code

We have a default bootstrap table html element.



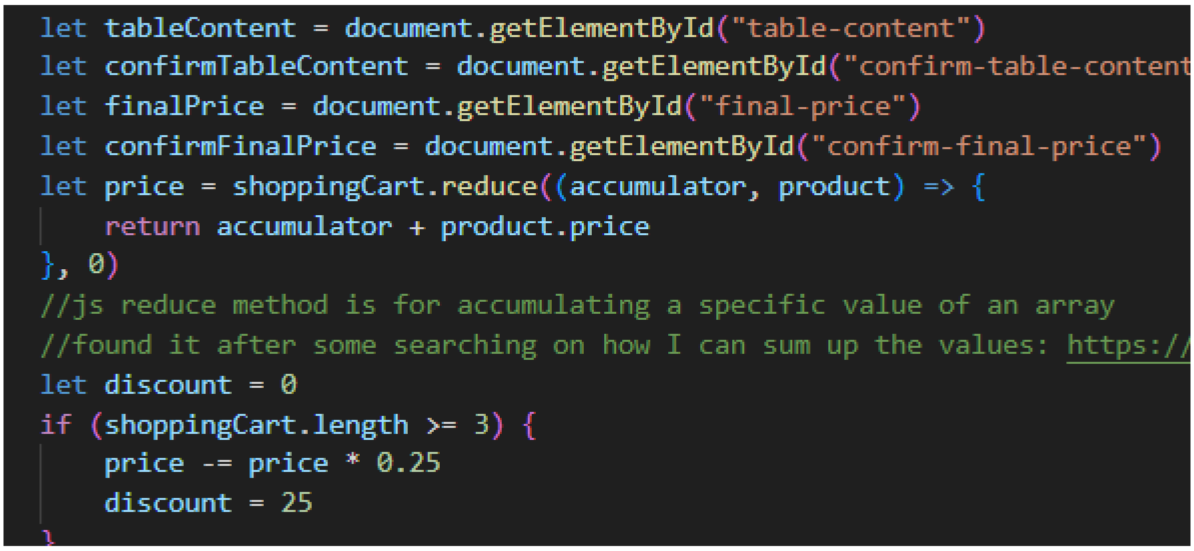
```html
<table id="table" style="text-align: center;" class='table table-striped'>
    <thead>
        <tr>
            <th style="text-align: left;">Product</th>
            <th>Price</th>
        </tr>
    </thead>
    <tbody id="table-content">
        <!--table rows will be injected here by js-->
    </tbody>
    <tfoot>
        <tr>
            <td style="text-align: left;"><b>Final price</b></td>
            <!--final price will be written here by js-->
            <td><b><span id="final-price"></span></b></td>
        </tr>
    </tfoot>
</table>
```

And a ShoppingCart function in our javascript code.



```javascript
const ShoppingCart = () => {
    let tableContent = document.getElementById("table-content")
    let confirmTableContent = document.getElementById("confirm-table-content")
    let finalPrice = document.getElementById("final-price")
    let confirmFinalPrice = document.getElementById("confirm-final-price")
    let price = shoppingCart.reduce((accumulator, product) => {
        return accumulator + product.price
    }, 0)
    //js reduce method is for accumulating a specific value of an array
    //found it after some searching on how I can sum up the values: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce
    let discount = 0
    if (shoppingCart.length >= 3) {
        price -= price * 0.25
        discount = 25
    }
    tableContent.innerHTML = "" //have to clear popup so it does not add more products to cart every time you click the button
    tableContent.innerHTML += shoppingCart.map(product => {
        return `
            <tr>
                <td class="product-title">${product.title}</td>
                <td class="product-price">$${product.price}</td>
                <td><div class="deleteText" onclick="deleteProduct(${product.id})" style="color: red">Delete</div></td>
            </tr>
        `
    }).join("")
    finalPrice.innerHTML = `$${price.toFixed(2)} <b>(discount: -${discount}%)</b>`
    confirmTableContent.innerHTML = ""
    confirmTableContent.innerHTML += shoppingCart.map(product => {
        return `
            <tr>
                <td class="product-title">${product.title}</td>
                <td class="product-price">$${product.price}</td>
            </tr>
        `
    }).join("")
    confirmFinalPrice.innerHTML = `$${price.toFixed(2)} <b>(discount: -${discount}%)</b>`

}

const deleteProduct = (product) => {
    shoppingCart.pop(product)
    ShoppingCart()

}
```

First we define some basic variables.



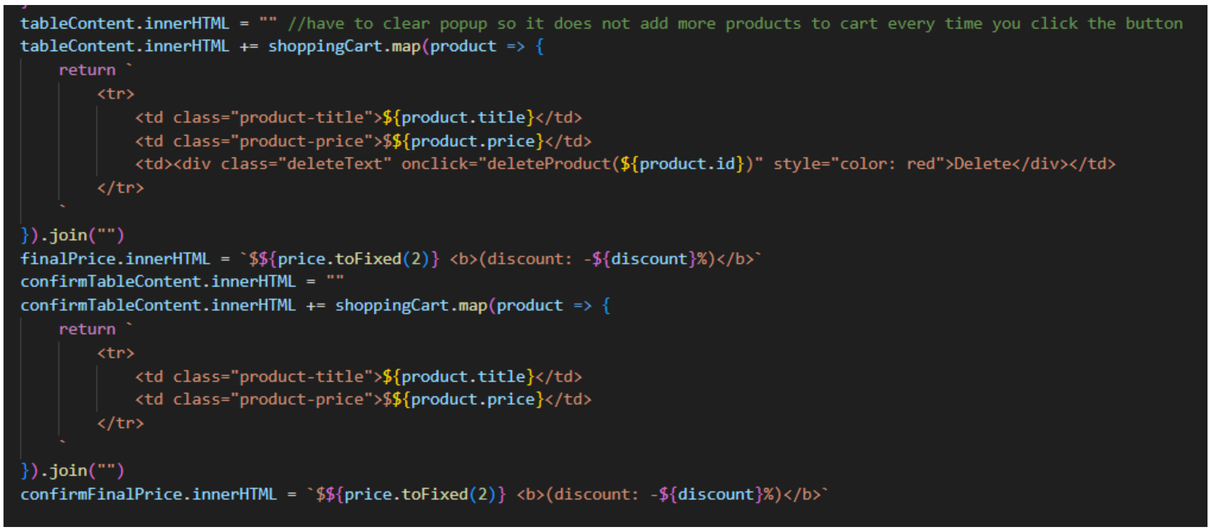
```
let tableContent = document.getElementById("table-content")
let confirmTableContent = document.getElementById("confirm-table-content
let finalPrice = document.getElementById("final-price")
let confirmFinalPrice = document.getElementById("confirm-final-price")
let price = shoppingCart.reduce((accumulator, product) => {
    return accumulator + product.price
}, 0)
//js reduce method is for accumulating a specific value of an array
//found it after some searching on how I can sum up the values: https://
let discount = 0
if (shoppingCart.length >= 3) {
    price -= price * 0.25
    discount = 25
}
```

Table content will be where we will render the full table.

Confirm table content will be where we render the confirmation table where we just see all the products that we will purchase and confirm our order in the Confirmation modal after the checkout.

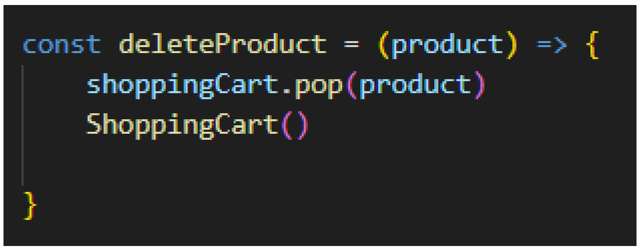
finalPrice and confirmFinalPrice is the final price that we are going to pay.

Discount will be the discount if we are purchasing 3 or more items.



```
tableContent.innerHTML = "" //have to clear popup so it does not add more products to cart every time you click the button
tableContent.innerHTML += shoppingCart.map(product => {
    return `
        <tr>
            <td class="product-title">${product.title}</td>
            <td class="product-price">$${product.price}</td>
            <td><div class="deleteText" onclick="deleteProduct(${product.id})" style="color: red">Delete</div></td>
        </tr>
    `
}).join("")
finalPrice.innerHTML = `$${price.toFixed(2)} <b>(discount: -${discount}%)</b>`
confirmTableContent.innerHTML = ""
confirmTableContent.innerHTML += shoppingCart.map(product => {
    return `
        <tr>
            <td class="product-title">${product.title}</td>
            <td class="product-price">$${product.price}</td>
        </tr>
    `
}).join("")
confirmFinalPrice.innerHTML = `$${price.toFixed(2)} <b>(discount: -${discount}%)</b>`
```

Every time we open the modal we have to clear the table and re-render the products so they do not accumulate. Then we go through the shoppingCart list which we defined outside of the function as a global variable and where our items that we added to the cart are put in. We join them together and return them into the tableContent. We also do this for the confirmTableContent too so we do not have to create a separate function for it and we can do it in one place at one time. We also render a delete button for the table content so we can delete the items we accidentally put into the shopping cart by calling the deleteProduct function with the current product id. We do not render this button into the confirmTable.

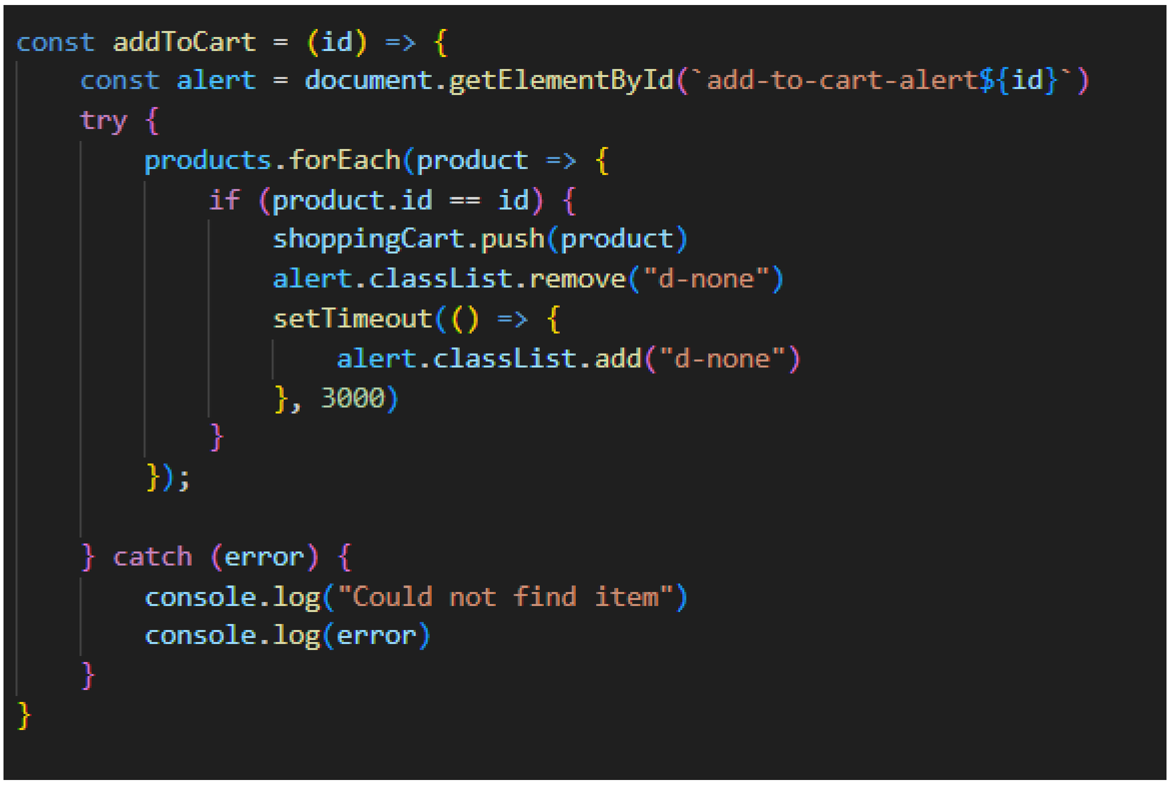


```
const deleteProduct = (product) => {
    shoppingCart.pop(product)
    ShoppingCart()

}
```

It just pop's the item from the shoppingCart global array and re-renders the shopping cart by calling the ShoppingCart function.

Add to Cart button/Bootstrap Buttons

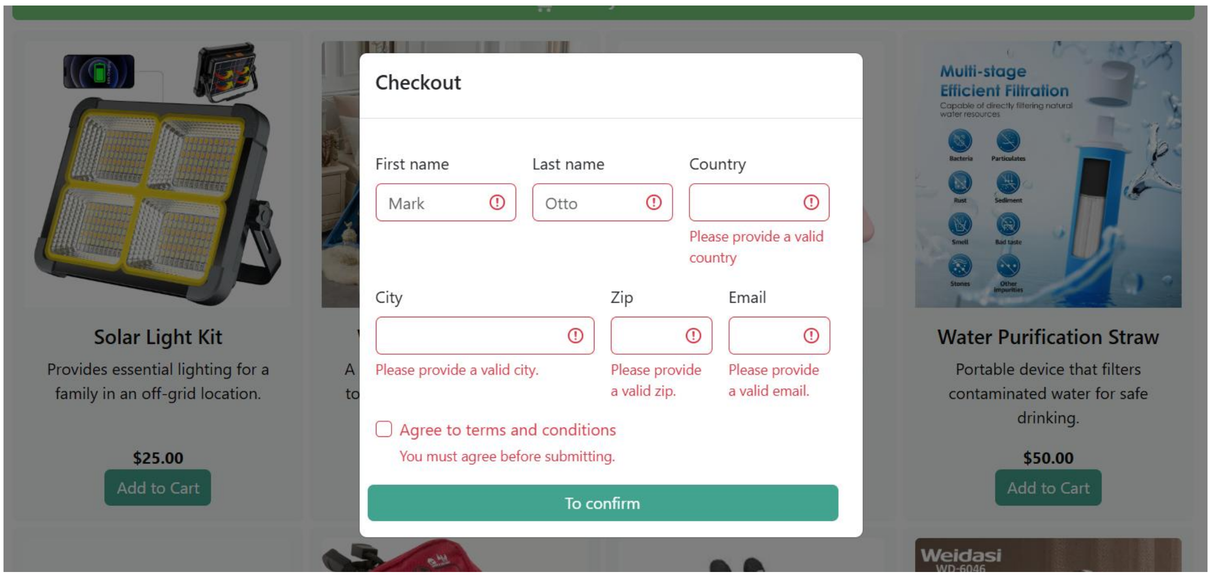
We have a default Bootstrap button as an add to cart button.



```
const addToCart = (id) => {
    const alert = document.getElementById(`add-to-cart-alert${id}`)
    try {
        products.forEach(product => {
            if (product.id == id) {
                shoppingCart.push(product)
                alert.classList.remove("d-none")
                setTimeout(() => {
                    alert.classList.add("d-none")
                }, 3000)
            }
        });

    } catch (error) {
        console.log("Could not find item")
        console.log(error)
    }
}
```

Every time someone click on the button the addToCart function runs. The parameter in the addToCart function (id) is the product's Id which is needed for rendering an alert that the product is added to the cart which becomes visible every time that the button is clicked inside the card of the product. Every product has an invisible alert element added to them and if the button is clicked we remove the bootstrap d-none property so the element becomes visible then after 3 seconds we remove the d-none property so the alert fades away. We go through all of the elements of the products list and if the id's are the same we push the element into the shoppingCart array that is defined as a global variable.

Checkout Modal (Bootstrap Form With Validation)



Checkout

First name: Mark

Last name: Otto

Country:

Please provide a valid country

City:

Please provide a valid city.

Zip:

Please provide a valid zip.

Email:

Please provide a valid email.

☐ Agree to terms and conditions

You must agree before submitting.

To confirm

Solar Light Kit

Provides essential lighting for a family in an off-grid location.

$25.00

Add to Cart

Water Purification Straw

Portable device that filters contaminated water for safe drinking.

$50.00

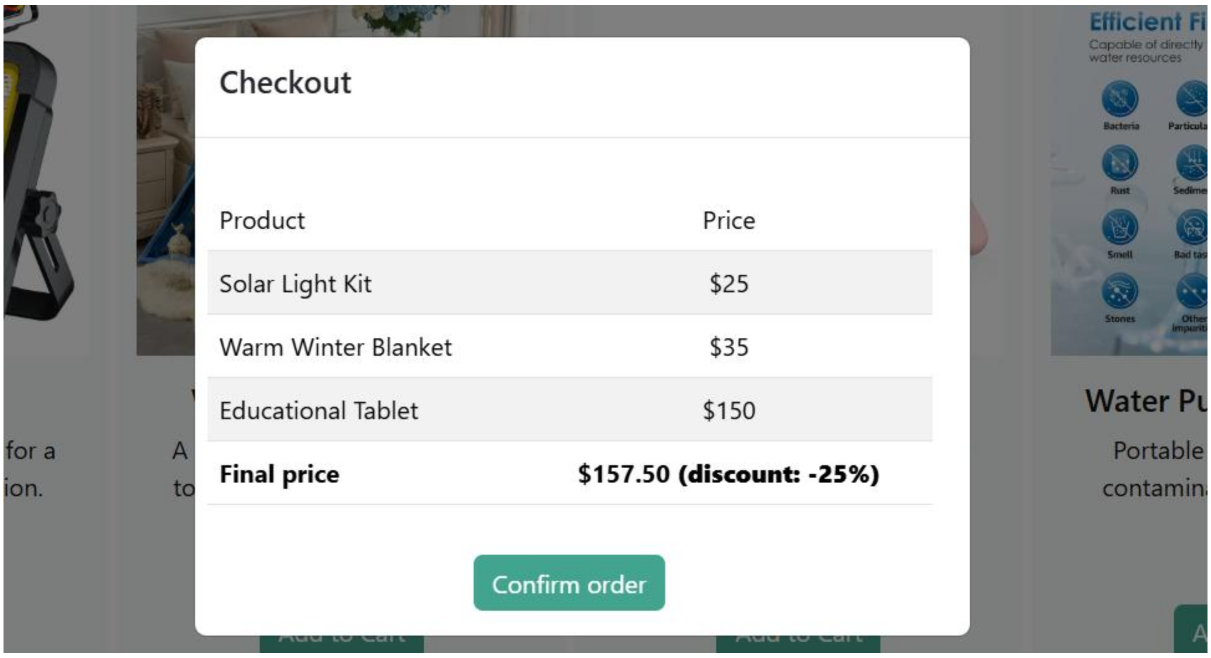
Add to Cart

An other default modal component which contains the form with some basic bootstrap validations.

Code for the basic validation

```
const confirmOrder = () => {
    const form = document.getElementById('checkoutForm');
    const formDiv = document.getElementById("formDiv");
    const confirmDiv = document.getElementById("confirmDiv");

    if (!form) {
        console.error("Form element 'checkoutForm' not found.");
        return;
    }

    //Add the 'was-validated' class to manually trigger Bootstrap's validation messages
    form.classList.add('was-validated');

    //Check if all required fields are valid.
    if (form.checkValidity()) {
        formDiv.classList.add("d-none");
        confirmDiv.classList.remove("d-none");
        ShoppingCart()
    }
}
```

Confirm Modal

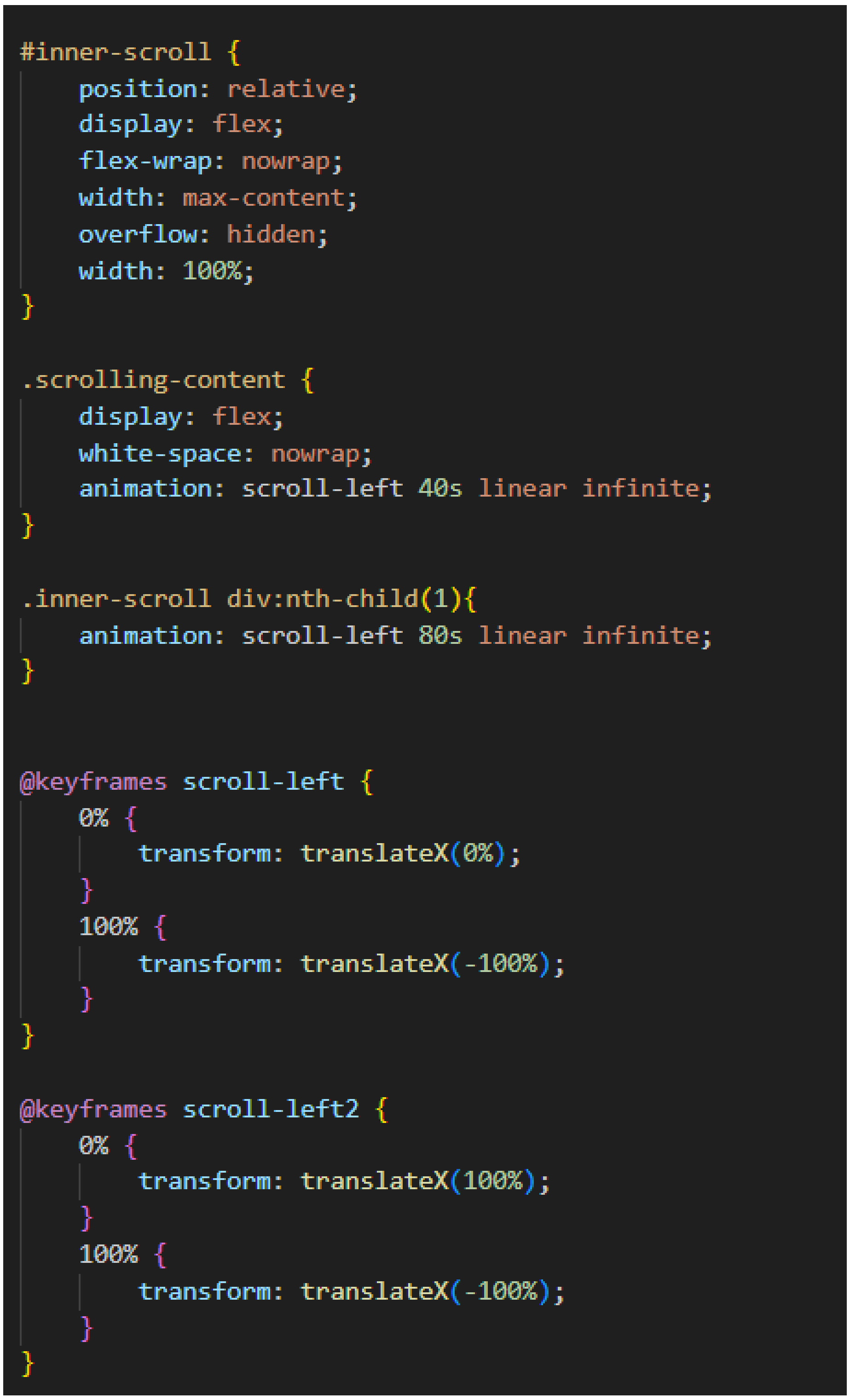


Checkout

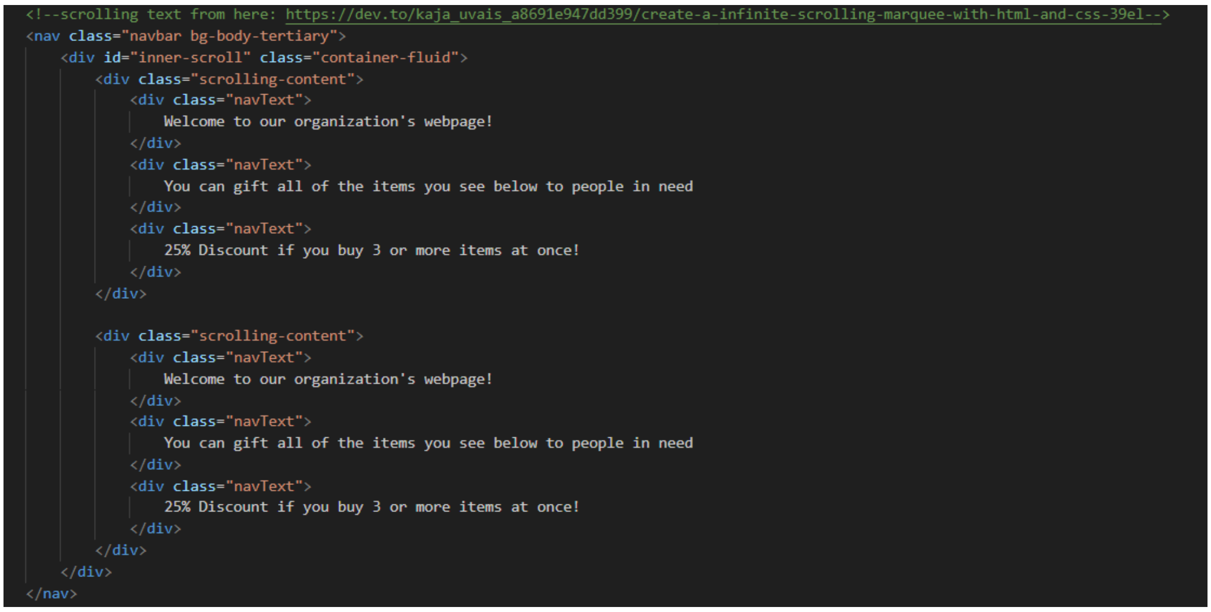
Product	Price
Solar Light Kit	$25
Warm Winter Blanket	$35
Educational Tablet	$150
Final price	**$157.50 (discount: -25%)**

You see again what you ordered and how much it costs. Here you can confirm your order of the items you want to purchase

Scrolling Animation



```css
#inner-scroll {
    position: relative;
    display: flex;
    flex-wrap: nowrap;
    width: max-content;
    overflow: hidden;
    width: 100%;
}

.scrolling-content {
    display: flex;
    white-space: nowrap;
    animation: scroll-left 40s linear infinite;
}

.inner-scroll div:nth-child(1){
    animation: scroll-left 80s linear infinite;
}


@keyframes scroll-left {
    0% {
        transform: translateX(0%);
    }
    100% {
        transform: translateX(-100%);
    }
}

@keyframes scroll-left2 {
    0% {
        transform: translateX(100%);
    }
    100% {
        transform: translateX(-100%);
    }
}
```



```html
<!--scrolling text from here: https://dev.to/kaja_uvais_a8691e947dd399/create-a-infinite-scrolling-marquee-with-html-and-css-39el-->
<nav class="navbar bg-body-tertiary">
    <div id="inner-scroll" class="container-fluid">
        <div class="scrolling-content">
            <div class="navText">
                Welcome to our organization's webpage!
            </div>
            <div class="navText">
                You can gift all of the items you see below to people in need
            </div>
            <div class="navText">
                25% Discount if you buy 3 or more items at once!
            </div>
        </div>

        <div class="scrolling-content">
            <div class="navText">
                Welcome to our organization's webpage!
            </div>
            <div class="navText">
                You can gift all of the items you see below to people in need
            </div>
            <div class="navText">
                25% Discount if you buy 3 or more items at once!
            </div>
        </div>
    </div>
</nav>
```

We have 2 times the text in one inner scroll div and we apply the 2 animations separately to the 2 separate divs so it becomes smooth and does not jump back to the start every time the animation ends.

Sources

Scrolling text: https://dev.to/kaja_uvais_a8691e947dd399/create-a-infinite-scrolling-marquee-with-html-and-css-39el

Bootstrap Card: https://getbootstrap.com/docs/5.3/components/card/

Bootstrap Button: https://getbootstrap.com/docs/5.3/components/buttons/

Popup Modal: https://getbootstrap.com/docs/5.3/components/modal/#how-it-works

Form layout and bootstrap validations:
https://getbootstrap.com/docs/5.3/forms/validation/

Bootstrap table: https://getbootstrap.com/docs/5.3/content/tables/

Color combination idea: https://htmlcolorcodes.com/color-chart/

Gradient navbar background colors: https://cssgradient.io/

Javascript reduce method: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce

Fading cards animation idea: https://www.youtube.com/watch?v=T33NN_pPeNI