Hard-coding Floating Action Buttons

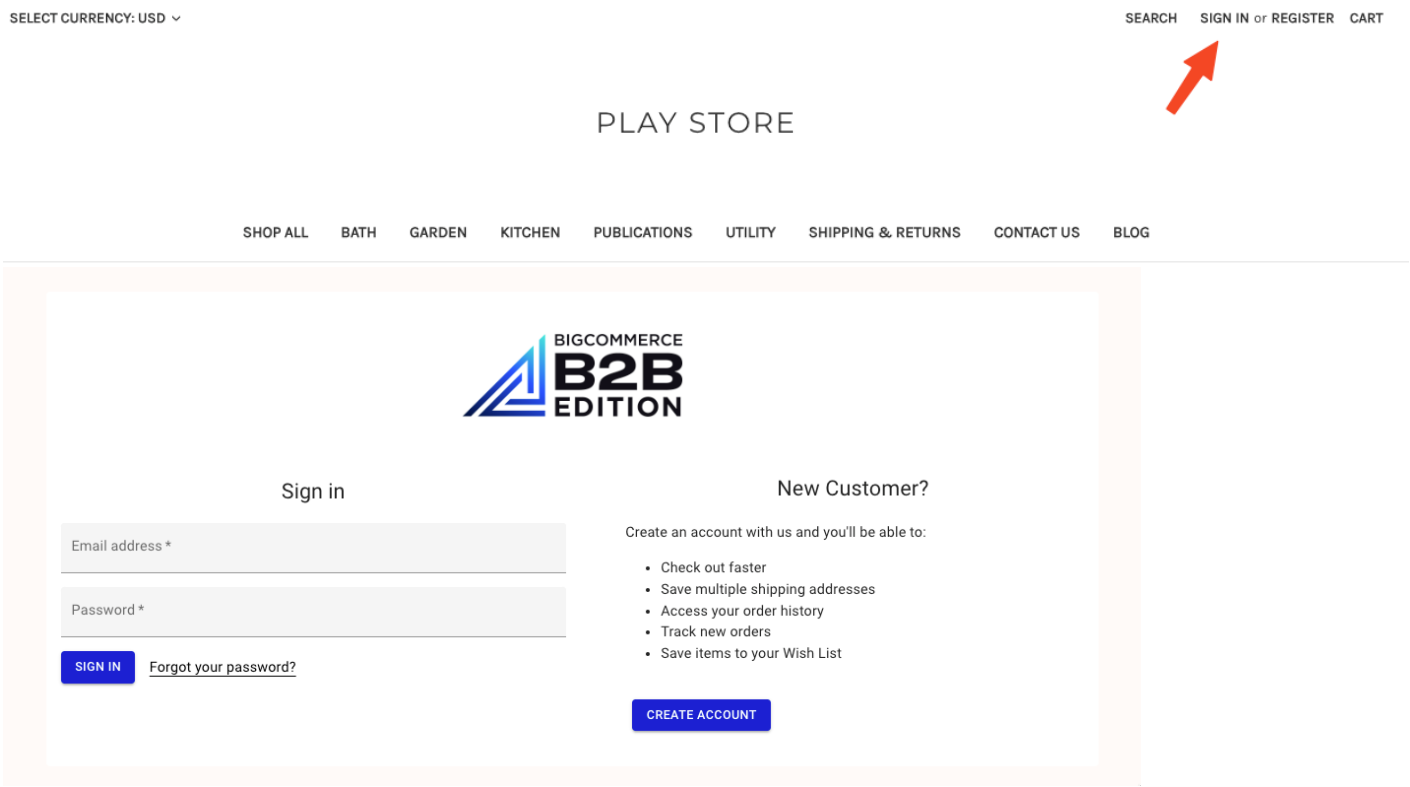
Note: Links to B2B Edition documentation in this PDF are no longer active. Search the BigCommerce Help Center for relevant articles.

This article assumes you have development skills. It is not a step-by-step tutorial, and vague in details on actual implementation.

Buyer Portal Entry Points

Entrance Points are the different points at which a buyer enters the Buyer Portal. The Buyer Portal automatically overrides these entry points on the top BigCommerce themes. If you are using an alternative, or custom theme you may want/need to directly point your theme to the buyer portal. You can use these functions to open specific pages within the Buyer Portal.

Sign In



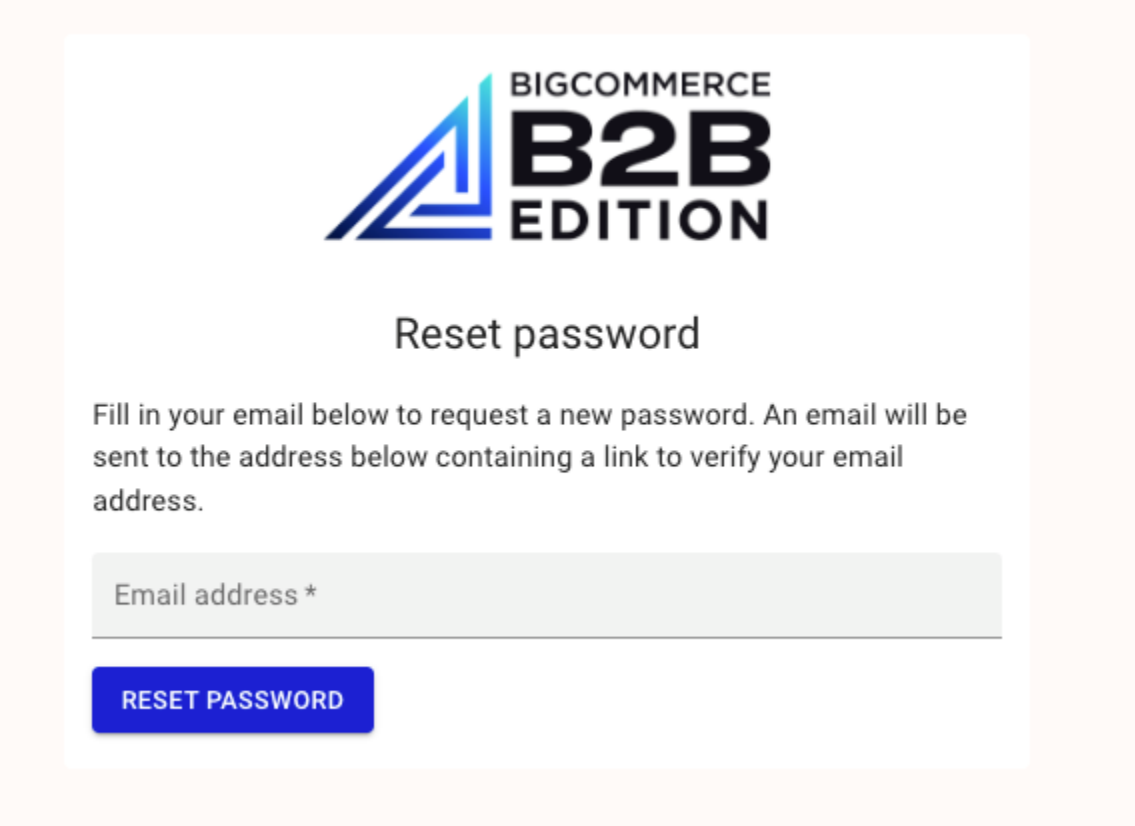
The Important Part:

onclick=”b2b.utils.openPage("SIGN_IN")”

Sample Code:

```
// link
<a onclick="b2b.utils.openPage('SIGN_IN')">SIGN IN</a>
// button
<button onclick="b2b.utils.openPage('SIGN_IN')">SIGN IN</button>
```

Forgot Password



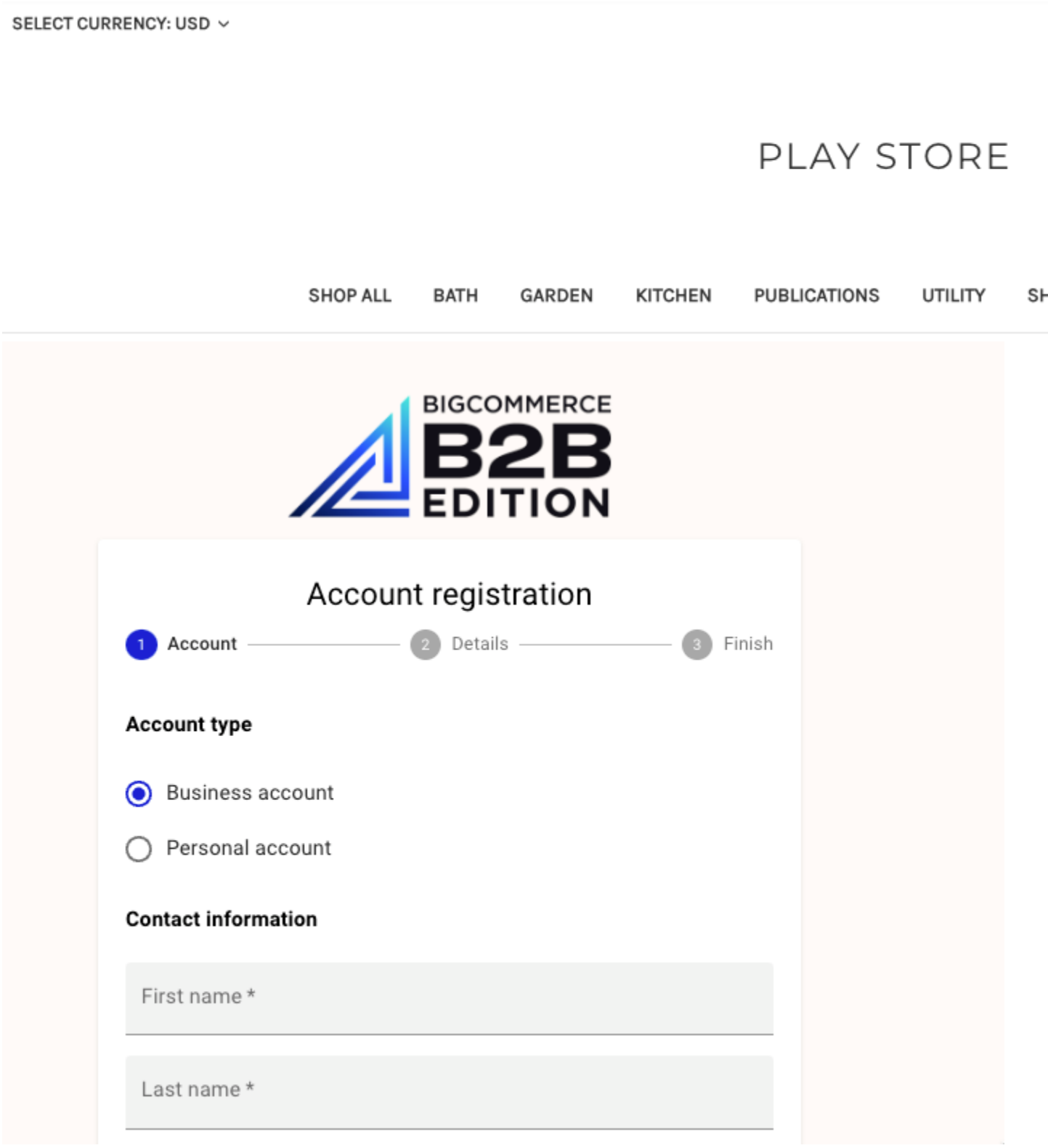
The Important Part:

onclick=”b2b.utils.openPage(‘FORGOT_PASSWORD’)”

Sample Code:

```
// link
<a onclick="b2b.utils.openPage('FORGOT_PASSWORD')">RECOVER ACCESS</a>
// button
<button onclick="b2b.utils.openPage('FORGOT_PASSWORD')">RECOVER
ACCESS</button>
```

Register / Create Account



The Important Part:

onclick=”b2b.utils.openPage(‘REGISTER_ACCOUNT’)”

Sample Code:

```
// link
<a onclick="b2b.utils.openPage('REGISTER_ACCOUNT')">REGISTER</a>
// button
<button onclick="b2b.utils.openPage('REGISTER_ACCOUNT')">REGISTER</button>
```

B2B and B2C Navigation

The Buyer Portal overrides the standard theme navigation to display and redirect available navigation links. It however does not include all navigation options. Navigation menus are unique to each store as they differ for B2C and B2B users, and depend on the feature set being utilized.

You can also use these functions anywhere else on your theme to link to specific buyer portal pages.

- B2B Menu Navigation
- My orders
- Company orders
- Invoice
- Quotes
- Shopping lists
- Quick order
- Addresses
- User management
- Account Settings

B2C Menu Navigation

- My orders
- Quotes (if enabled for B2C/Individual users in Settings > Quotes)
- Shopping lists (if enabled for B2C/Individual users in Settings > Shopping list)
- Quick order
- Addresses
- Account Settings

The Important Part:

onclick=”b2b.utils.openPage(‘EnterPageTypeHere’)”

Buyer Portal Page Types:

- "SIGN_IN"
- "FORGOT_PASSWORD"
- "REGISTER_ACCOUNT"
- "DRAFT_QUOTE"
- "LOG_OUT"
- "SHOPPING_LISTS"
- "DASHBOARD"
- "ORDERS"
- "COMPANY_ORDERS"
- "QUOTES"
- "PURCHASED_PRODUCTS"
- "ADDRESSES"
- "USER_MANAGEMENT"
- "ACCOUNT_SETTINGS"
- "INVOICE"

Sample Code:

```
<script>
  // Menu items
  const routes = {
    "DASHBOARD": "Dashboard",
    "ORDERS": "My orders",
    "COMPANY_ORDERS": "Company orders",
    "INVOICE": "Invoice",
    "QUOTES": "Quotes",
    "SHOPPING_LISTS": "Shopping lists": ,
    "PURCHASED_PRODUCTS": "Quick order",
    "ADDRESSES": "Addresses",
    "USER_MANAGEMENT": "User management",
    "ACCOUNT_SETTINGS" :"Account settings",
  };

  // Valid routes per kind of session
  const SUPERADMIN_WITHOUT_MASQUERADE = ["DASHBOARD", "ORDERS",
"ACCOUNT_SETTINGS"]
  const SUPERADMIN_WITH_MASQUERADE = ["DASHBOARD", "ORDERS",
"COMPANY_ORDERS", "INVOICE", "QUOTES", "SHOPPING_LISTS",
"PURCHASED_PRODUCTS", "ADDRESSES", "USER_MANAGEMENT", "ACCOUNT_SETTINGS"]
  const GUEST_USER = ["ORDERS", "PURCHASED_PRODUCTS", "ADDRESSES",
"ACCOUNT_SETTINGS"]
  const B2C_USER = ["ORDERS", "ADDRESSES", "ACCOUNT_SETTINGS"]
  const B2B_USER = ["ORDERS", "COMPANY_ORDERS", "INVOICE", "QUOTES",
"SHOPPING_LISTS", "PURCHASED_PRODUCTS", "ADDRESSES", "USER_MANAGEMENT",
"ACCOUNT_SETTINGS"]

  const {role,email} = b2b.utils.user.getProfile()
  const {current_company_id: currentCompanyId} = await
b2b.utils.user.getMasqueradeState()
```

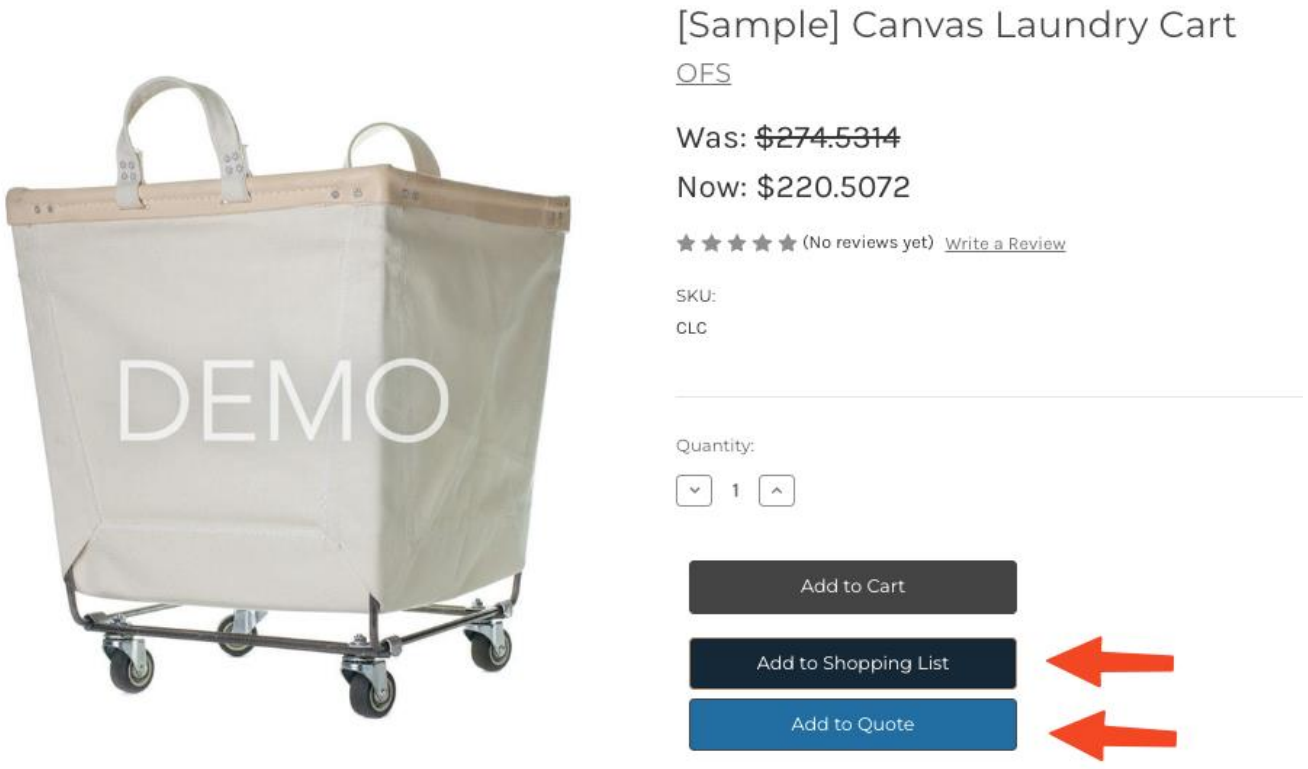
```
  const { enabled: addToQuoteBtn } = b2b.utils.quote.getButtonInfo()
  const { enabled: addToShoppingListBtn } =
b2b.utils.shoppingList.getButtonInfo()

  // filter valid routes
 const validRoutes = Object.entries(routes).filter(([route]) => {
    // if user is superadmin
    if(role === 3){
      // and it's not using masquerade feature
      if(currentCompanyId === 0){
        return SUPERADMIN_WITHOUT_MASQUERADE.includes(route)
      }
      return SUPERADMIN_WITH_MASQUERADE.includes(route)
    }

    // if it's guest user
    if(!emailAddress) {
      // Special cases
      if(route === "QUOTES" && addToQuoteBtn) {
        return true
      } else if (route === "SHOPPING_LISTS" && addToShoppingListBtn){
        return true
      }
      return GUEST_USER.includes(route)
    }

    // if user is b2c
    if(role === 99){
      // Special cases
      if(route === "QUOTES" && addToQuoteBtn) {
        return true
      } else if (route === "SHOPPING_LISTS" && addToShoppingListBtn){
        return true
      }
      return B2C_USER.includes(route)
    }

    // if non of other cases applies, then it's a b2b user
    return B2B_USER.includes(route)
  })

  // generate menu
  const menuItems = validRoutes.map(([route, text]) => `<button
onclick="b2b.utils.openPage('${route}')">${text}</button>`).join("")

  // render menu
  document.querySelector(".menu-container").innerHTML = menuItems
</script>
```

Quote and Shopping List Buttons

Product Display Page Buttons

We inject these using the BC script manager, but they can be hard-coded to the theme if you'd like for more precise placement, eliminate loading time, or create custom workflows. You can turn off the injected button on and off in Settings > Quotes or Settings > Shopping list respectively. ***(This assumes you've already migrated to the Buyer Portal.)***

For long-term maintenance, we find it's best for most merchants to use injected buttons. [You can change the container they are injected to if they are not injecting where you like, or you are using non-standard containers on your custom theme.]



If you would like to hard-code the button, you'd do something similar to the following:

The important part for the add to quote button:

onclick="b2b.utils.quote.addProductFromPage(productObj)"

Sample Code:

```
// simplest button
<button id="add-to-quote">Add to quote</button>

<script>
  /* ProductOption type
    interface ProductOption {
      optionEntityId: number
```

```
      optionValueEntityId: number
    }
  */
  const addToQuoteButton = document.getElementById("add-to-quote");
  addToQuoteButton.addEventListener("click", () => {
    b2b.utils.quote.addProductFromPage({
      quantity: number
      productEntityId: number
      selectedOptions?: ProductOption[]
      sku?: string
      variantEntityId?: number
    })
  )};
</script>

// render button by using getButtonInfo function
<script>
  const { enabled, text, customCss, color, customTextColor, classSelector } =
b2b.utils.quote.getButtonInfo();
  if(!enabled) return

  const addToQuoteButton = document.createElement('button');

  addToQuoteButton.innerHTML = text || 'Add to Quote'
  addToQuoteButton.setAttribute('style', customCss)
  addToQuoteButton.style.backgroundColor = color
  addToQuoteButton.style.color = customTextColor
  addToQuoteButton.setAttribute('class', classSelector)
  addToQuoteButton.addEventListener("click", () => {
    b2b.utils.quote.addProductFromPage({
      quantity: number
      productEntityId: number
      selectedOptions?: ProductOption[]
      sku?: string
      variantEntityId?: number
    })
  )};

  document.appendChild(addToQuoteButton);
</script>
```

The important part for the add to shopping list button:

onclick="b2b.utils.quote.addProductFromPage(productObj)"

Sample Code:

```
// simplest button
<button id="add-to-shopping-list">Add to shopping list</button>

<script>
  /* ProductOption type
    interface ProductOption {
      optionEntityId: number
      optionValueEntityId: number
```

```
    }
  */
  const addToShoppingListButton = document.getElementById("add-to-shopping-
list");
  addToShoppingListButton.addEventListener("click", () => {
    b2b.utils.shoppingList.addProductFromPage({
      quantity: number
      productEntityId: number
      selectedOptions?: ProductOption[]
      sku?: string
      variantEntityId?: number
    })
  )};
</script>

// render button by using getButtonInfo function
<script>
  const { enabled, text, customCss, color, customTextColor, classSelector } =
b2b.utils.shoppingList.getButtonInfo();
  if(!enabled) return

  const addShoppingListButton = document.createElement('button');

  addShoppingListButton.innerHTML = text || 'Add to Shopping List
  addShoppingListButton.setAttribute('style', customCss)
  addShoppingListButton.style.backgroundColor = color
  addShoppingListButton.style.color = customTextColor
  addShoppingListButton.setAttribute('class', classSelector)
  addShoppingListButton.addEventListener("click", () => {
    b2b.utils.shoppingList.addProductFromPage({
           quantity: number
           productEntityId: number
           selectedOptions?: ProductOption[]
           sku?: string
           variantEntityId?: number
    })
 )};

  document.appendChild(addShoppingListButton);
</script>
```

A note about maintaining functionality controls as defined by app settings.

- When using B2B Edition services, it is important to note that additional layers of functionality are added to role/user/buyer-based configurations and settings. You can maintain the app-specific functionality by including the default class of the specific button type so that the including that button is appropriately hidden if that user type does not have access to that functionality.

- This reduces the need to write additional logic to know when to display a specific functionality to a user, but does not prevent you from adding additional logic, such as restrictions by brand, value, meta-data, etc.

    - *Ie: you may want to additionally hide the add-to-quote button based on meta-data stored on a product.*

Sample Setting scenario:

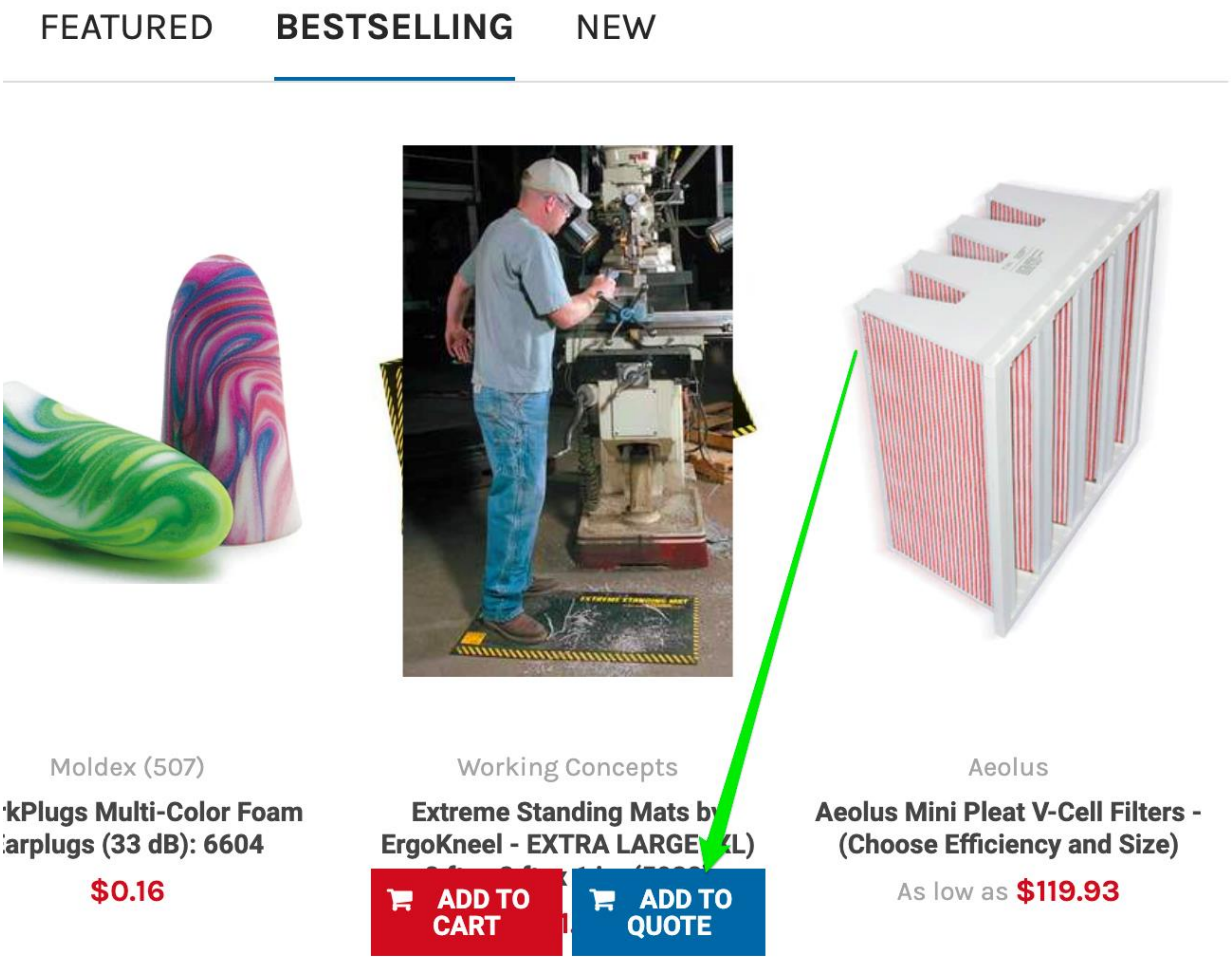
- Assuming above settings, and **WITHOUT CONSUMING** getButtonInfo function properties on the Add to Quote button:
    - Guest users (shoppers) will be able to request quotes

- Assuming above settings, and **CONSUMING** getButtonInfo function properties on the Add to Quote button:
    - Guest users(shopper) will NOT be able to request quotes and
    - **WILL NOT** see the Add to Quote button

Category Buttons

These will redirect to the product page, validate for options, inventory, etc.

It is possible to set these up to work from the category page, and submit the product directly to our quote cart from the category page.



This code is usually added to card.html or list-item.html files.

The important part:

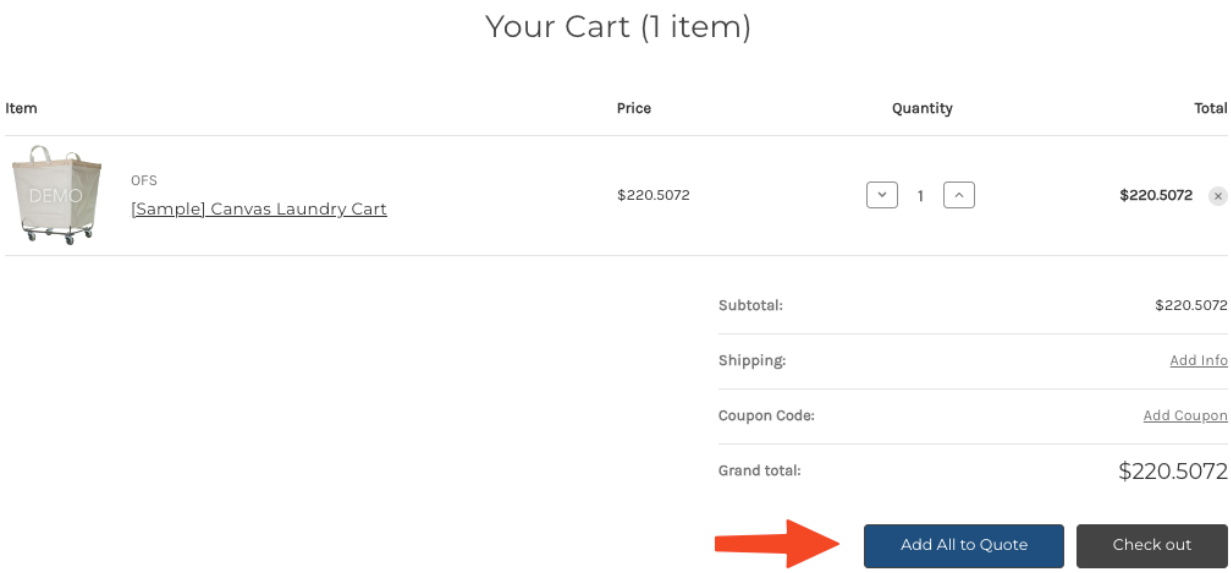
The example it's the same as quote button section, both goes inside product-view.html template file

```
// simplest button
<button id="add-to-quote">Add to quote</button>

<script>
  /* ProductOption type
    interface ProductOption {
      optionEntityId: number
      optionValueEntityId: number
    }
  */
```

```
  const addToQuoteButton = document.getElementById("add-to-quote");
  addToQuoteButton.addEventListener("click", () => {
    b2b.utils.quote.addProductFromPage({
      quantity: number
      productEntityId: number
      selectedOptions?: ProductOption[]
      sku?: string
      variantEntityId?: number
    })
  )};
</script>

// render button by using getButtonInfo function
<script>
  const { enabled, text, customCss, color, customTextColor, classSelector } =
b2b.utils.quote.getButtonInfo();
  if(!enabled) return

  const addToQuoteButton = document.createElement('button');

  addToQuoteButton.innerHTML = text || 'Add to Quote'
  addToQuoteButton.setAttribute('style', customCss)
  addToQuoteButton.style.backgroundColor = color
  addToQuoteButton.style.color = customTextColor
  addToQuoteButton.setAttribute('class', classSelector)
  addToQuoteButton.addEventListener("click", () => {
    b2b.utils.quote.addProductFromPage({
      quantity: number
      productEntityId: number
      selectedOptions?: ProductOption[]
      sku?: string
      variantEntityId?: number
    })
  )};

  document.appendChild(addToQuoteButton);
</script>
```

Cart to Quote Button

Used when adding the contents of the cart to a draft quote.



The important part:

onclick="b2b.utils.quote.addProductsFromCart()"

Sample Code:

```
<button onclick="b2b.utils.quote.addProductsFromCart()"> Add All to Quote</button>
```

Finish Quote Button

Use the open page function to show the current draft quote in progress.

The important part:

onclick=”b2b.utils.openPage(`DRAFT_QUOTE’)”

Sample Code:

<button onclick="b2b.utils.openPage('DRAFT_QUOTE')">Finish Quote</button>