# Dokumentace

## Frontend

Důležitou částí je zde propojení javascriptu s html, což se děje pomocí HTML atributu.

### Propojovací HTML attribute

**shop-id="{name}::{type}@{event}#{id}"**

**{name}**

název daného modulu (input, carousel, modal)

**{type}**

typ daného modulu (input::number, input::select)

**{event}**

javascript event po jehož spuštění se provede určitá akce pro daný modul. V závorkách za eventem lze specifikovat nějaký parameter  (modal@click(close))

**{id}**

identifikátor pro určitou instanci daného modulu. Nemusí být unikátní a dá se použít jako propojovací prvek dvou instancí daného modulu (modal#12 -> modal@click(open)#12)

# Komponenty

## Carousel

**name:** carousel

**type:** —

**event:** click(left), click(right)

**id:** jakékoliv

Carousel komponent slouží pro komplexní práci se sliderama projektů, bannerů…

### Pomocné HTML atributy

**data-options** - pro nakonfigurování carouselu přímo v html

#### *Důležité options carouselu pro konfiguraci*

**type** - typ carouselu (slide, loop, fade) **string, default: slide**

**showArrowsOnHover** - zobrazí šipky jen po najetí na carousel **boolean, default: false**

**showPaginationOnHover** - zobrazí paginaci jen po najetí na carousel **boolean, default: false**

**rewind** - určuje zdali má docházet k přetočení carouselu **boolean, default: true**

**perPage** - určuje počet počet slidů zobrazených na jedné stránce **number, default: 1**

**perMove** - určuje počet o kolik slidů se má carousel posouvat **number, default: 1**

**gap** - určuje mezeru mezi jednotlivými slidy **number | string, default: podle gridu**

**arrows** - určuje zdali se mají zobrazit šipky **boolean, default: false**

**pagination** - určuje zdali se má zobrazit paginace **boolean, default: false**

**autoplay** - určuje zdali má docházek k automatickému posouvání **boolean, default: false**

**interval** - určuje interval, po kterém se carousel posune **number, default: 5000**

**breakpoints** - objekt pro změnu konfigurace při responsivu

### Struktura HTML

Podle struktury HTML javascript ví jak má s carouselem pracovat. Je tedy nutné strukturu HTML zachovat

```
<div  shop-id=”carousel” data-options=’{“type”: “loop”}’ class=”splide”>
        <div class=”splide__track”>
                <ul class=”splide__list”>
                        <li class=”splide__slide”></li>
                        <li class=”splide__slide”></li>
                        <li class=”splide__slide”></li>
                        <li class=”splide__slide”></li>
                </ul>
```

```
        </div>
</div>
```

### Příklady

**shop-id="carousel#landing"** - samotný carousel

**shop-id="carousel@click(left)#landing"** - po kliknutí na element se carousel s id landing posune do prava

## Modal

**name:** modal

**type:** gallery, visible

**event:** click(close), click(open), click(toggle)

**id:** jakékoliv

Modal komponent jednoduše slouží k základní práci s madalovými okny. Zobrazí se zatmavené pozadí a samotný modal, po kliknutí na zatmavené pozadí se modal zavře.

### Gallery

- možnost použití vlastních ikonek s použitím **data-gallery** attributu na body
    - **data-gallery**='{"icon_close": "cross", "icon_left": "left", "icon_right": "right"}'

### Struktura HTML

v DOMu musí být element s id #modal, který se stará o začernění obrazovky. Jednotlivé modaly uvnitř tohoto elementu, jsou pak samostatné karty modalu bez zatemnění pozadí. Modal se může dát i mimo #modal. Každý modal musí na sobě mít **shop-id="modal#identifikator"**

### Příklady

**shop-id="modal#minicart"** - samotný modal

**shop-id="modal@click(open)#minicart"** - po kliknutí na element se otevře modal s id #minicart

**shop-id="modal::visible#minicart"** - po otevření modalu #minicart bude element s typem visible nad ztmavenou částí

**shop-id="modal::gallery#products", shop-id="modal::gallery#products", shop-id="modal::gallery#products", shop-id="modal::gallery#products"** - na obrázky, které

se mají přidat do společné galerie #products. Po kliknutí na obrázek se otevře modal s galerii #products

## Input

**name:** input

**type:** select, number, file, range

**event:** —

**id:** jakékoliv

Input komponent slouží pro vytváření komplexních custom inputů jako select, multiselect, datepicker….

Pomocné HTML atributy

**data-icon** (type=file) - classa pro ikonku, která bude použita na tlačítku upload inputu

**data-button** (type=file) - text, který bude použit na tlačítku upload inputu

**data-img** (type=select) - na optionu selectu pro zobrazení obrázku pro danou možnost

**data-icon** (type=select) - na optionu selectu pro zobrazení ikonky pro danou možnost

**data-label** (type=range) - na input[type=range] pro text labelu daného inputu

**data-unit** (type=range) - jednotka zobrazených hodnot (EUR, Kč)

### Struktura HTML

**type=file**

- je nutné, aby element s shop-id=”input::file” obsahoval input[type=file].
- je jedno na jaké pozici bude v elementu label nebo input
- label není v elementu povinný

**type=select**

- label není povinný
- je jedno na jaké pozici bude v elementu label nebo input
- je nutné, aby element s shop-id=”input::select” obsahoval select element.

**type=number**

- label není povinný
- je jedno na jaké pozici bude v elementu label nebo input
- je nutné, aby element s shop-id=”input::number” obsahoval input[type=number]

**type=range**

- label není povinný
- je jedno na jaké pozici bude v elementu label nebo input
- je nutné, aby element s shop-id=”input::range” obsahoval 2x input[type=range]

### Příklady

** **

```
        <label for="country">Stát</label>
        <select name="country" id="country" placeholder="Vyberte stát...">
            <option value="option-1" data-img="/images/cz.png">Možnost 1</option>
            <option value="option-2" data-img="/images/sk.png">Možnost 2</option>
            <option value="option-3" data-img="/images/usa.png">Možnost 3</option>
        </select>
</div>
```

## Dropdown

**name:** dropdown

**type:** –

**event:** click, hover

**id:** jakékoliv

Dropdown se po spuštění určitého eventu změní z is-hidden stavu na is-visible stav a naopak. Klikne-li uživatel mimo dropdown nebo opět na element, který otevírá dropdown, tak se dropdown zavře.

### Příklady

**shop-id="dropdown#myaccount"** - samotný dropdown (musí na sobě ve výchozím stavu mít is-hidden nebo is-visible)

**shop-id="dropdown@click#myaccount"** - Po kliknutí na tento element se otevře/zavře dropdown s id **myaccount**

## Tabs

**name:** tab

**type:** –

**event:** click

**id:** jakékoliv

Jednotlivé taby spadají do společné skupiny, která je propojena s id. Propojení tlačítka, který daný tab otevírá, spočívá v nastavení stejného typu na obou elementech.

### Příklady

**shop-id="tab::1#product"** - samotný tab skupiny product (jestli má být ve výchozím stavu viditelný, musím mít classu is-visible, jinak classu is-hidden)

**shop-id="tab::1@click#product"** - Po kliknutí na tento element se otevře tab s id product a s typem 1. Ostatní taby se zavřou.

## Menu

**name:** menu

**type:** recursive

**event:** —

**id:** jakékoliv

Menu komponent slouží pro konkrétnější a komplexnější práci s komponenty u menu např. rekurzivní akordeonové otevírací menu.

### Struktura HTML

**type=recursive**

```
<ul shop-id="menu::recursive#menu">
    <li>
      <a href="">
          <span>Sekce 1</span>
          <i class="bi bi-chevron-down"></i>
      </a>
      <ul>
          <li>
            <a>
                <span>Sekce 1.1</span>
            </a>
          </li>
      </ul>
    </li>
    <li>
      <a href="">
          <span>Sekce 2</span>
      </a>
    </li>
</ul>
```

**shop-id="menu::recursive#products"** - na ul elementu, který se má chovat jako rekurzivní akordeonové menu

## Header

**name:** header

**type:** sticky

**event:** —

**id:** jakékoliv

Header komponent slouží pro konkrétnější a komplexnější práci s Headerem např. sticknutí headeru při scrollování

Příklady

**shop-id=”header::sticky”** - jakmile se doskroluje k tomuto elementu, nastaví se na něm classa is-sticky

# Akce

**name:** action

**type:** –

**event:** show, remove, collapse

**id:** jakékoliv

Akce slouží pro jednoduché funkce jako odstranění elementu, zobrazení elementu na základě určité hodnoty nějakého inputu apod.

## Jednotlivé akce

- ke každé akci lze přidat parametr cache, který výsledek dané akce uloží do session storage
- **<button shop-id="action@click(collapse,cache)#accordion12"> otevřít </button>**

**show**

- show akce střídá na elementu is-visible a is-hidden

**<input type="checkbox" shop-id="action@change(show,true)#company_info" />**

** **

- jakmile se hodnota checkboxu změní, spustí se akce show, která na příslušném elementu nastaví is-visible, jestliže je je checkbox checked (true)

**remove**

- po kliknutí na button se spustí akce remove, která action#alert12 element kompletně vymaže z DOMu

```
<div class="alert" shop-id="action#alert12">
        <div>
                <p>lorem jdfjlsd jsdlf sdf</p>
        </div>
        <button shop-id="action@click(remove)#alert12">close</button>
</div>
```

**collapse**

- po kliknutí na button se spustí akce collapse, která action#accordion2 element otevře/zavře jako akordeon
- is-collapsed = ve výchozím stavu je rozevřený
- is-collapse = ve výchozím stavu je zavřený

**<button shop-id="action@click(collapse)#accordion12"> otevřít </button>**

** **

**fb_track**

- pro trackování přes facebook pixel
- **<button shop-id="action@click(fb_track,AddToCart)"> Přidat do košíku</button>**

**writing_effect**

- pro efekt psaní
- pro input element bude text psát do placeholderu, pro všechny ostatní elementy bude text psát jako innerText
- v data-write atributu jsou fráze oddělené **;**, které se mají vypisovat
- **<input type="email" shop-id="action@load(writing_effect)" data-write="avast;call of duty;minecraft" />**

**popup_message**

- postupně v daném elementu zobrazí zprávy z **data-messages** atibutu a následně elemen schova
- ** **
- jestli se má po kliknutí na nějaký jiný element popup_message zavřít dříve, musí být paramater action funkce close a zároveň id elementu se musí shodovat s id elementu, který chci zavřít
- ** **
- **<button shop-id="action@click(popup_message,close)#watching"></button>**

## Template

**name:** template

**type:** název templatu

**event:** –

**id:** jakékoliv

## Ajax

**name:** ajax

**type:** –

**event:** jakýkoliv javascript event (ajaxresource, methoda ajax resource) (@click(cart.product, add))

**id:** jakékoliv

### Pomocné HTML atributy

**data-data** - json ve stringu, který se odešle v ajaxu jako data

*Použití*

```
<div
    shop-id=”ajax@click#product1”
    data-data=’{“product_id”: “12”, “quantity”: “{(input#quantity).value || 1}”}’ >
</div>
```

- po kliknutí na **div** se spustí ajax s metodou GET
- jako query parametry requestu se vloží data v **data-data** atributu
    - jeli nějaký property v data-data atributu string uzavřen mezi {}, najde se mezi propojenými elementy (elementy které mají shop-id=“ajax#product1”) element a vezme se jeho hodnota (value) nebo vnitřní text (innerText). Není-li element mezi propojenými elementy nalezen, použije se výchozí hodnota 1 (viz. quantity)

### Obecné použití

*Request*

```
<div
    shop-id=”ajax@click#product1”
    data-data=’{“product_id”: “12”, “quantity”: “{(input#quantity).value || 1}”}’ >
</div>
```

- po kliknutí na **div** se spustí ajax s metodou GET
- jako query parametry requestu se vloží data v **data-data** atributu

*Čekání na response*

- během čekání na response se nastaví na elementu, který ajax spustil, na jeho propojených elementech a na jeho propojených **template** elementech classa **is-loading**

*Response*

- po úspěšně ukončeném request se:
    - se ze všech elementů, na kterých se nastavila class **is-loading**, odstrání classa **is-loading**
    - přepíší se templaty s typem z objektu templates v responsu
    - obsahuje-li response **redirect_url,** přesměruje se přes javascript na danou url
    - obsahuje-li response **replace_url,** přepíše se přes javascript url

Použití s ajax resource

- slouží pro vykonávání ajaxů pro určité **resources** (přidávání/odebírání produktů v košíku, vyhledávání, wishlist, filterování v nabídce produktů, atd.)
- tyto ajaxy většinou potřebují zpracovávat response dle potřeby

*Request*

- stejné jako při obecném použití, ale zároveň může daný **resource** provést vlastní dedikované operace
    - vložit do data requestu data navíc
        - “add”: “” (metoda na daném resourcu [add, delete, edit, atd.])
        - “ajax”: identifikátor requestu (“cart.product”)

*Čekání na response*

- stejné jako při obecném použití, ale zároveň může daný **resource** provést vlastní dedikované operace

*Response*

- stejné jako při obecném použití, ale zároveň může použít daný **resource** provést vlastní dedikované operace
    - vyhodnotit určité properties z dat responsu individálně (otevření košíku)

*Resources*

- **ProductResource** (cart.product)
    - pro přidávání/odebírání/upravování produktů v košíku na celém webu
    - **metody:** add, delete, edit
    - **response**
        - cart.mini.open - otevře košík
        - cart.products.count - přepíše element, který zobrazuje množství produktů v košíku
- **GiftResource** (cart.gift)
    - pro přidávání/odebírání dárků na kroku dárky v košíku
    - **metody:** add, delete
- **BillingResource** (cart.billing-information)
    - pro vyplňování formuláře z adresáře na kroku fakturačních údajů v košíku
    - **metody:** add_delivery_address, add_billing_address
- **CheckoutResource** (cart.checkout)
    - pro změnu způsoby dopravy/platby na třetím kroku košíku
    - **metody:** add
- **WishlistResource** (wishlist)
    - pro přidávání/odebírání produktů do seznamu přání na celém webu
    - **metody:** add, delete
    - **response**
        - wishlist.products.count - přepíše element, který zobrazuje množství produktů v seznamu přání
- **SearchResource** (search)

    - pro vyhledávání produtků v eshopu
    - **response**
        - search.products.open - otevře box se zobrazenými výsledky vyhledávání
- **ShopResource** (shop)
    - pro filterování produktů v nabídce produktů
    - **response**
        -