Refine element selection with the code selector

Some applications require code selector fine-tuning. This depends on the complexity of your application. With the code selector option, you can hook into the automatically recorded code selector to make sure that it always finds the right element.

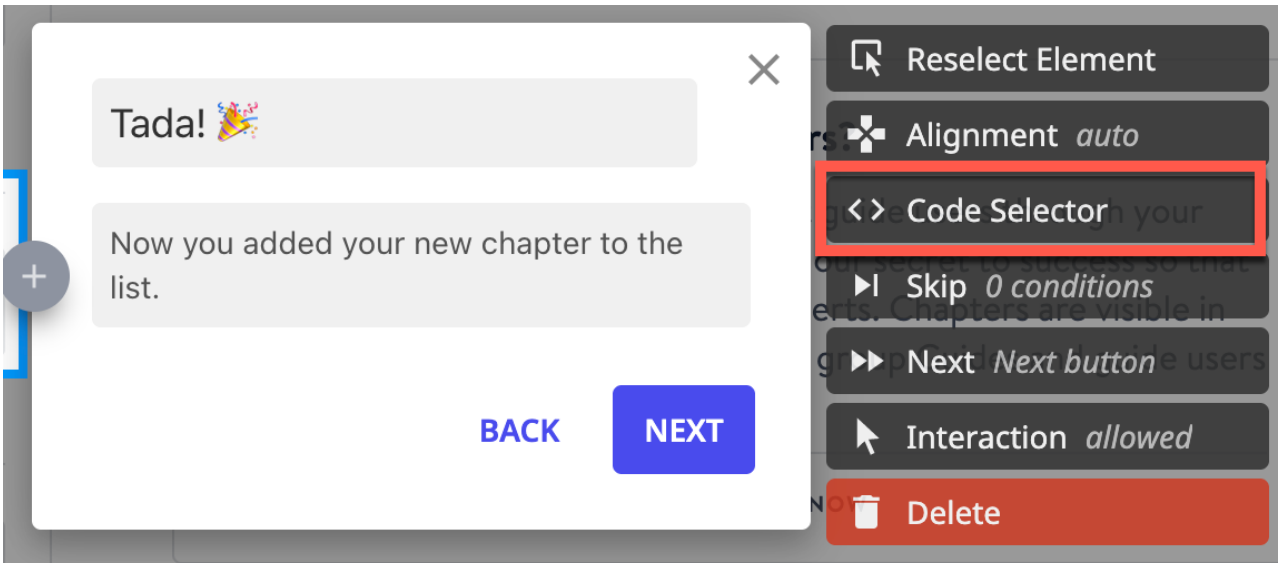
How it works

You can define a code selector by manually adding

- An element class or an element ID or an element attribute
- A content selector
- A subSelector (e.g. for iFrames)
- ParentNum for a bigger element frame

Where to find it

You can find and manually edit the automatically recorded code selector of an element within the Editor in the black step options of the respective recorded step:



Userlane Code Selector structure

Structure basics:

	HTML code	Respective code in Userlane selector

Recorded element	<div class="userlane-slide-base userlane-base"
	<input
	id="userlane-search"
		{
		"selector":
		"DIV.userlane-slide-base.userlane-base > INPUT#userlane-search",
		"orderNum": 0
		}

General structure	<div class="element class"
	<input id="elementid"
		{
		"selector":
		"DIV.element.class > INPUT#elementid",
		"orderNum": 0
		}

- **Red:** The value for the code selector defines the highlighted element, e.g. a DIV box or an INPUT element.
- You can **separate multiple elements** by space or greater than character (>). By doing so, you go one level deeper in the code structure (you select a more specific element within the bigger element).
- **Blue:** The class of an element is added to the element with a dot ('.'). If the element class has a space in its value, use another dot instead of the space. A space would mean to go an undefined level deeper in the code structure.
- **Yellow:** The ID of an element is added to an element with a '#' (see *Example 1*)
- You can refine element selection with = or ***=.** If you apply `=`, the class, for example, must be exactly the value that follows `=`. If you add `*=`, the class must *contain* the value that follows `*=` (see *Example 3*).
- orderNum defines the **number of elements that match the code selector value** It starts with 0, so if the orderNum is 2, the third element that matches the code selector value will be chosen.

Example 1: General code selector structure with class and ID:

```
{
"selector": "DIV.element.class > INPUT#elementid",
"orderNum": 0
}
```

Example 2: Code selector structure with attributes

```
{
"selector": "INPUT[readonly=\"true\"].main-element",
"orderNum": 0
}
```

Example 3: Code selector structure with (stable) parts of attributes or classes

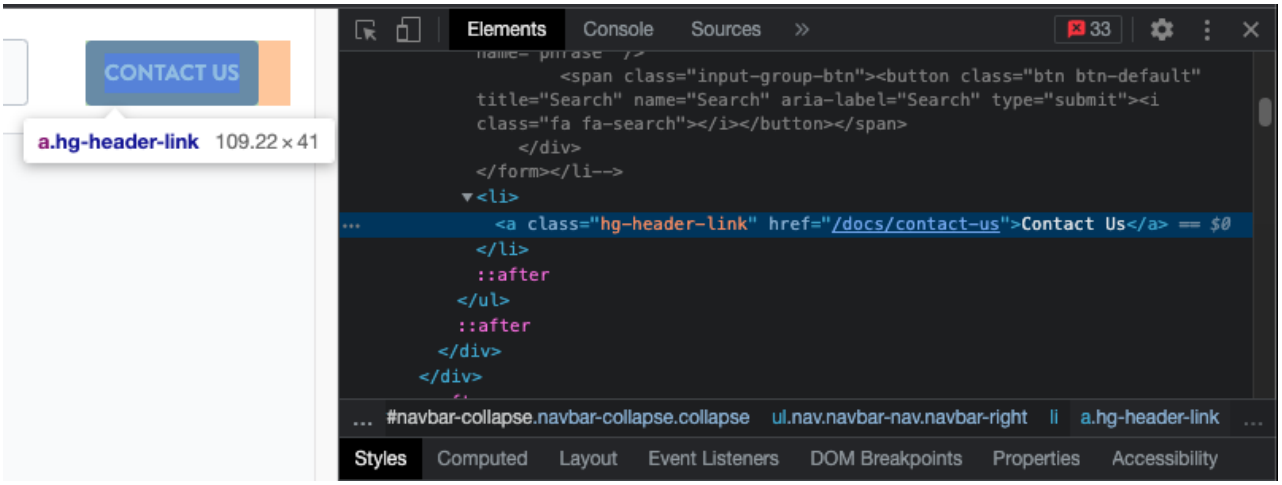
```
{
“selector”: “DIV[class*=CustomerSearch__InputContainer] INPUT[class*=SearchInput__StyledInput]“,
“orderNum”: 0
}
```

How to manually adjust the code selector

Change the value of an element

You can look up any element in your browser’s developer tools and define the code selector value by yourself with the rules above. To get to the HTML code, you can right-click on the element and click at ‘Inspect’.

Example:



Add a content selector

You can also modify the selector to make sure that it chooses an element that has specific content. The detailed process is described in Target specific text for an element with the content selector.

Alternatively:

1. **Open the code selector** in the step options of the respective recorded step.

2. Add the content selector with the **“Add content selector” button**. If this button is not available this doesn’t work,
   - **Copy-paste the following selector code** in the selector input field (beneath the old selector).
   - For label please insert the label code of the respective element (**copy-paste the “selector” part from the old/recorded selector**).
   - For *text-to-look-for* **insert the text/content** which is in the element you would like to highlight.
   - **Delete the old code** of the recorded selector.

3. **Save your selector changes** with the respective button. Check your changes by opening the selector pop up again because if the syntax of your code is not correct, the new selector will not be saved.

Content selector structure:

```
{
"selector": "label",
"orderNum": 1,
"content": {
"type": "contains",
"value": ["text-first-language", "text-second-language"]
}
}
```

This is a content selector with "type": "contains". Therefore, only a part of the text of the element has to be typed into the content selector. If you want to make sure that an element with the exact same content written in the element is selected, you can use a content selector with "type": "matches":

Adding a subSelector (e.g. for iFrames)

SubSelectors are necessary for iFrames (you cannot have a normal selector that goes inside an iFrame) or for other elements where you want to find a smaller 'child' element within a defined bigger 'parent' element.

Each subSelector has the same general syntax structure as a normal selector:

```
{
"selector": "IFRAME#same-origin-iframe-id",
"orderNum": 0,
"subSelector": {
"selector": "DIV.example.class",
"orderNum": 0
}
}
```

With this selector code, the Userlane selector would search for a DIV element with the class 'example class' (subSelector), which is inside the iFrame element with the ID 'same-origin-iframe-id' (selector). You can insert and adapt this general code in the same way as described in the content selector above.

Adding parentNum

ParentNum will take an ascendant ('parent') of the selected element. This is useful when you need to enlarge the highlighted area of a specifically defined element. This is also helpful to select elements bigger than the target element you want to highlight, especially when you want your Guide to appear visually more appealing. Here is a code example with parentNum value of 2, which therefore would select the 'grandparent' element (two levels up) of the targeted DIV-box:

```
{
"selector": "DIV.element-class",
"orderNum": 2,
"parentNum": 2
}
```

Ignoring Opacity

This is useful when some elements in your page have oppacity:0 until a user hovers over them. You can add this to your selector if you wish to have Userlane find the element even if it's not visible for the end user:

```
{
 "selector": "DIV > DIV > DIV > DIV > BUTTON > SVG",
 "orderNum": 0,
 "ignoreOpacity": true
}
```

Good to know

- Manually editing selectors can be pretty complex. Please don't hesitate to contact us via our contact form in case of any issues you run into and we'll help you to make it work :)
- Always make sure that the syntax of the edited selector is correct. We suggest you to always copy-paste the examples from this article and then adapt the new selector so that you do not forget a comma or quotes. You can also always check your changed selector by simply opening it again after you saved it.