
pluginDocs

Release 0.0.1

Jul 21, 2020

Contents:

1	How to use this Plug-In correctly	1
1.1	Step 1: Groups and elementboxes	1
1.2	Step 2: Name your elements	3
1.3	Step 3: Select your Items	4
1.4	Step 4: Use the plugin	4
2	How to deal with the exported files	11
2.1	How to import the exported files into your web application	11
2.2	Handle the .js file	13
2.3	Handle the .css file	15
2.4	Why this index.html?	15
2.5	Extra .html file	15
3	Web components	17
3.1	What are web components?	17
4	FAQs	19
4.1	1. Why do I not see the filepath?	19
4.2	2. Why does my export looks different than in Adobe XD?	19

How to use this Plug-In correctly

1.1 Step 1: Groups and elementboxes

A. Make necessary groups

To make sure, that **buttons** and **inputfields with a placeholder** will implemented correctly, you have to group the *rectangle* and the *textfield* together:

1. Select the two elements you want to group

- select the first element
- press `Shift`
- select the other element

2. Group them together

- right click on one of the selected elements
- choose *Group*
- OR: use the shortcut: `Ctrl + G`

3. Name the group

- double click on the name of your group (e.g Group 1)
you find your elements in the list on the left hand side
- – choose a unique name that includes the function of this group - button or input
(e.g: `my_input` - for an inputfield OR `my_button` - for a button)

Hint: If you want to design your own logos or icons you also have to group all elements the logo/icon consists and the name of the group must also include the keyword: **img**

B. Resize the elementboxes

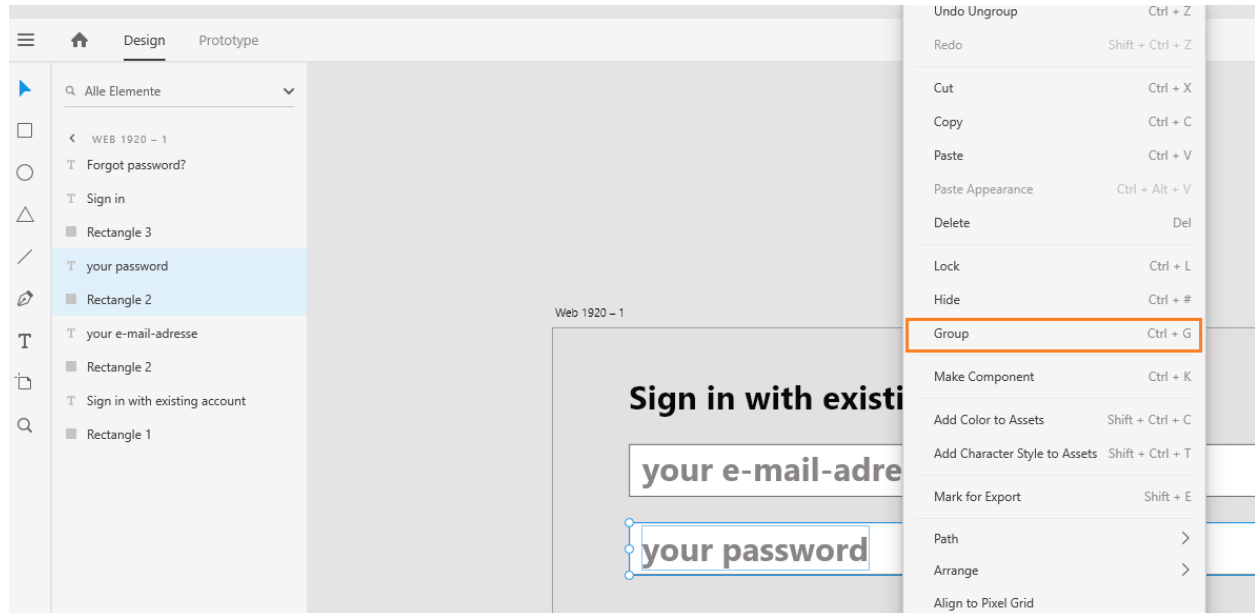


Fig. 1: Selecting and grouping the elements (ref: 1. + 2.)

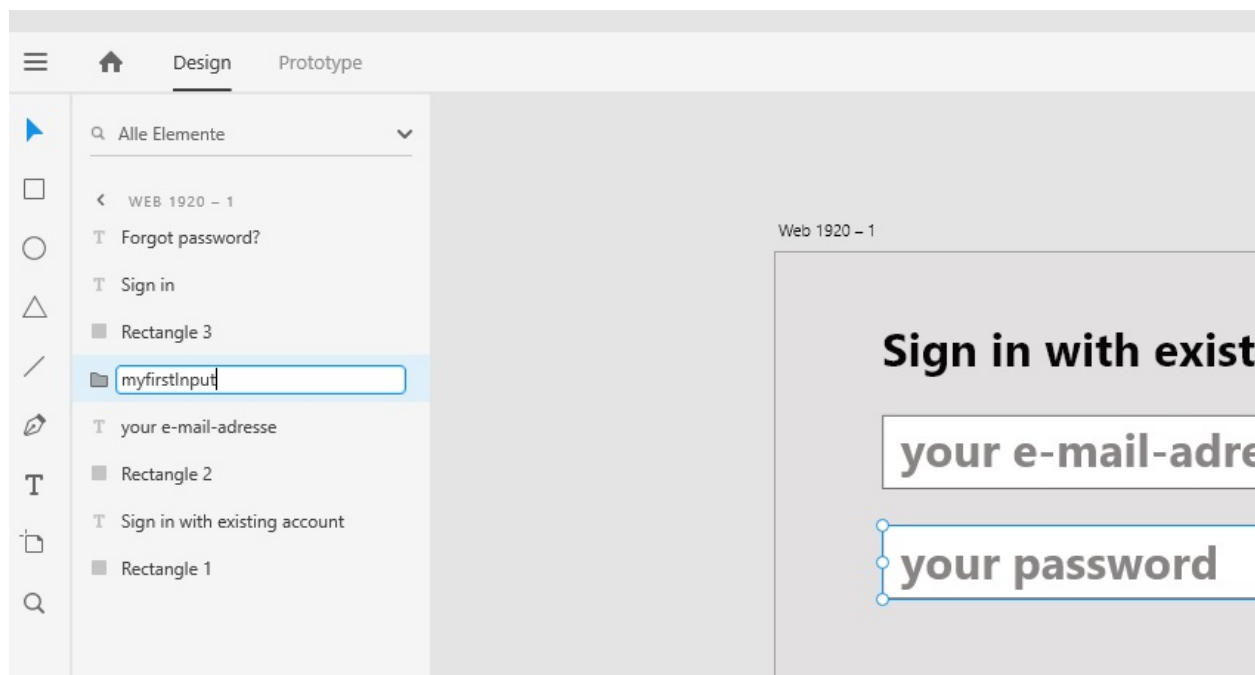


Fig. 2: Rename the group (ref: 3.)

Make sure that the elementboxes from your elements - especially from the textfield are as big as the text it self. Otherwise the margin will not calculate correctly.

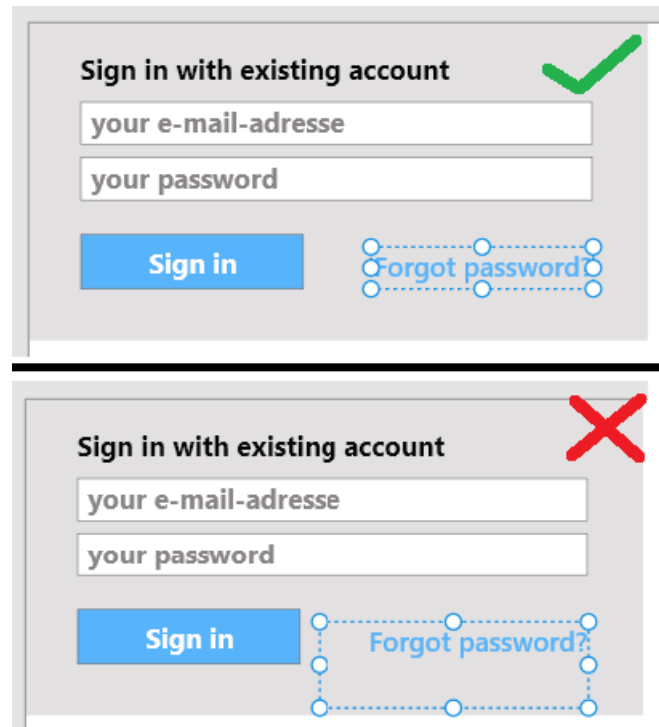


Fig. 3: Adjust the elementboxes.

1.2 Step 2: Name your elements

It is necessary to name specific elements the right way, so they will be implemented correctly. There for you will find all elements, that will be supported in the plugin and how to name them, in the table below.

Important: all names have to be one word or seperated with a _ !

Adobe element	HTML element	keyword ¹
rectangle + text (group)	button	button
rectangle + text (group)	input with placeholder	input
rectangle	inputfield	input
rectangle	checkbox	checkbox
rectangle	radiobutton	radiobutton
line	line	line
text ²	list	list
text	link	link
images/logos ³	image	img

Hint:

Make sure you give all your other elements an unique name as well.

Especially all your text-elements, because the name will be used as an **ID** in the CSS File.

Examples:

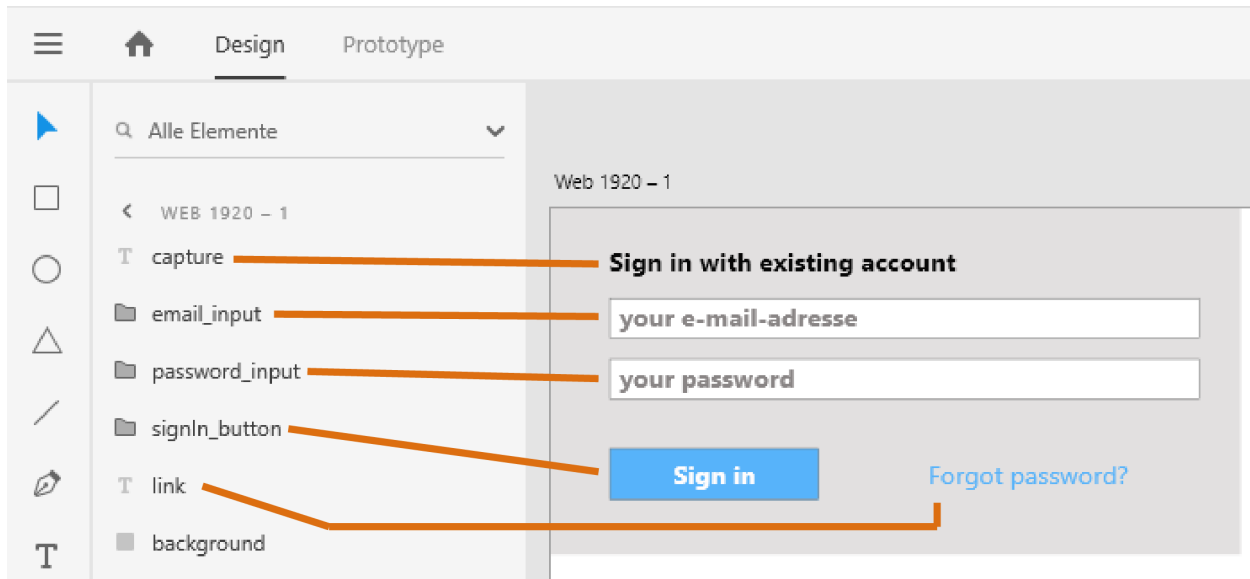


Fig. 4: Example how to name your elements correctly.

1.3 Step 3: Select your Items

For now it is important to **select the elements from top left to bottom right**:

1.4 Step 4: Use the plugin

After selecting all elements, go to your plugin selection and choose “Export as Webcomponent”.

A pop-up will appear and you first have to enter a name for your component.

Hint: Make sure you don’t forget the “-” in the name.

Then you are able to select a folder to save your component.

If everything is correct you can see the file path in the inputfield and the export will be saved automatically in the chosen folder.

¹ the keyword has to be included in the elementname

² Illustration: how to make a list:

³ Illustration: How to name a image:

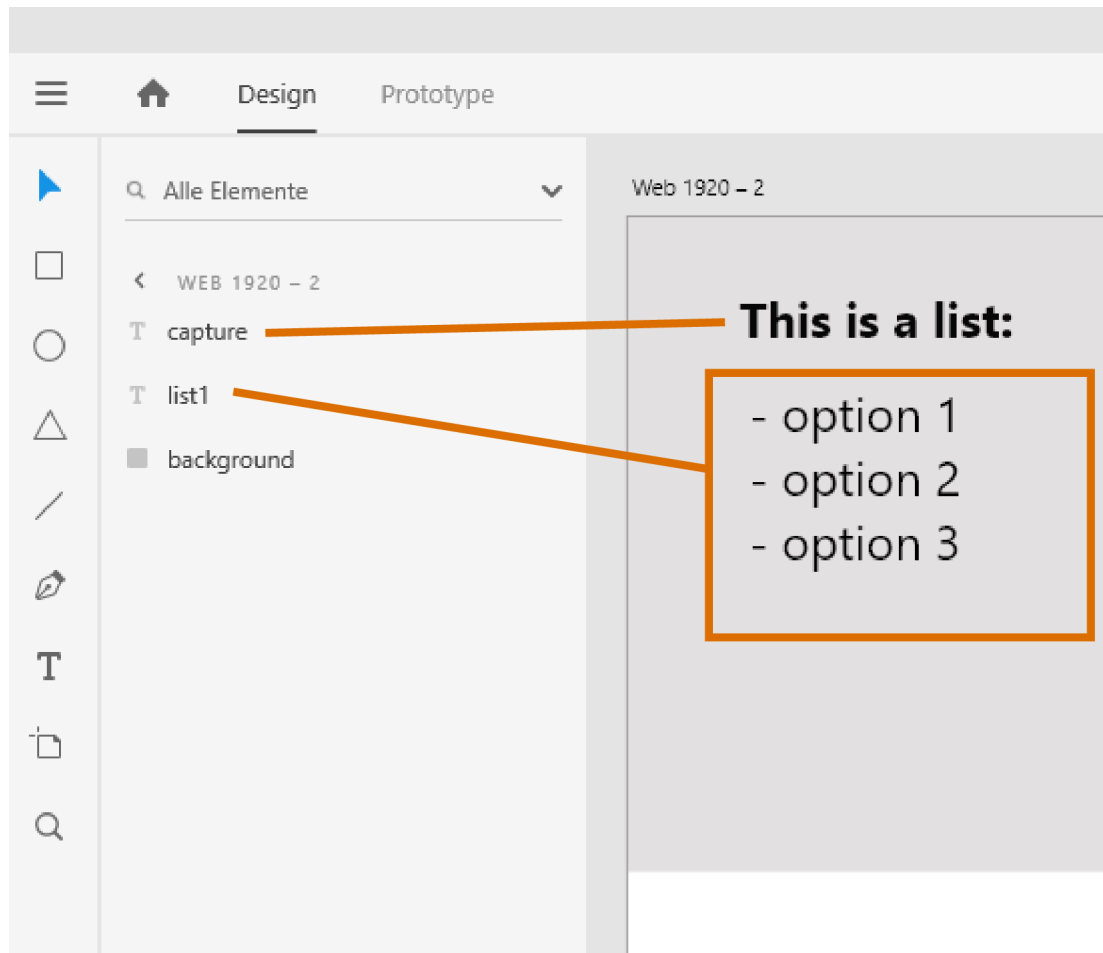


Fig. 5: Example how to make a list.

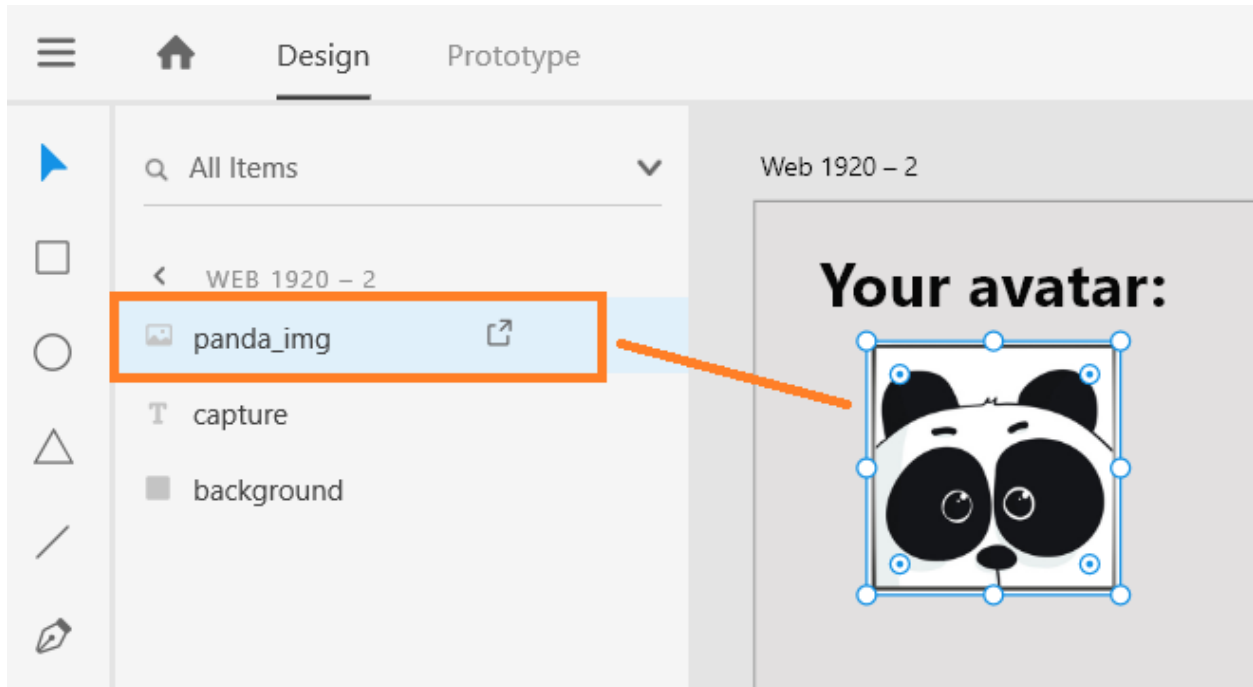


Fig. 6: Example how to name a image

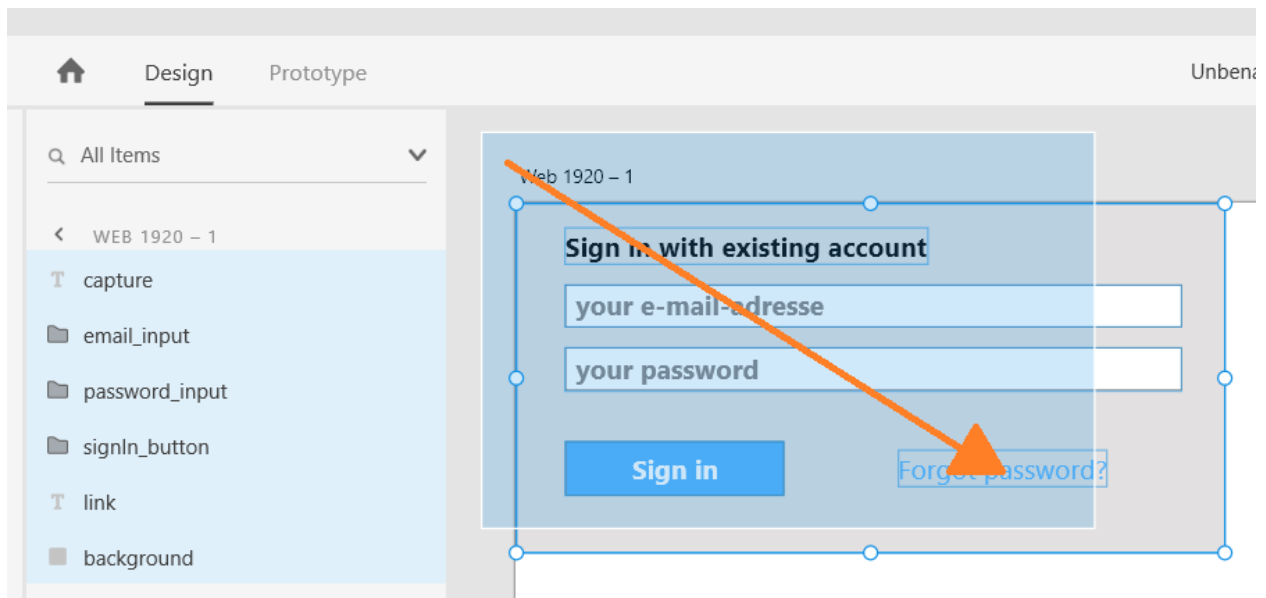


Fig. 7: How to select your elements

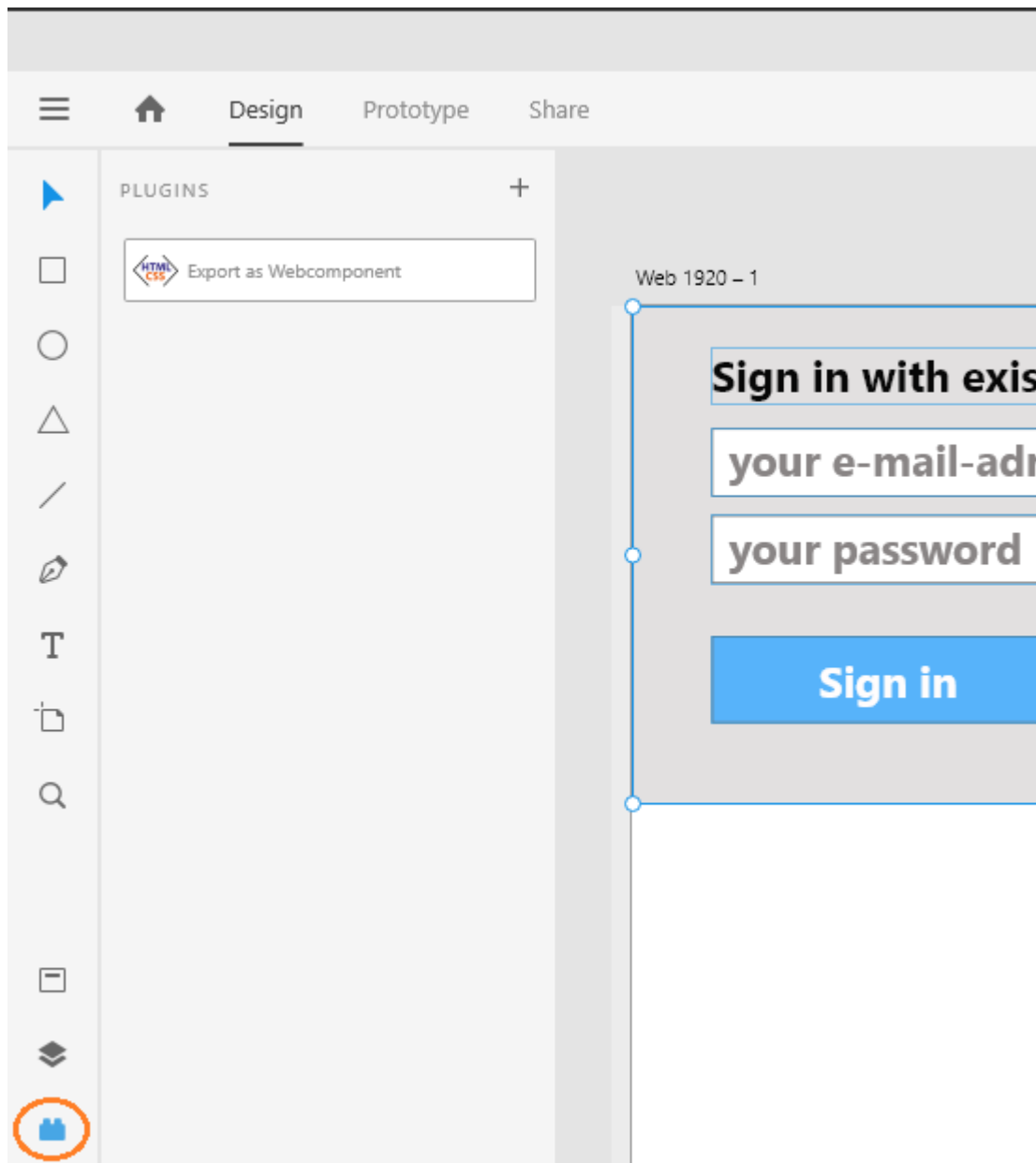


Fig. 8: Where to find the plugin

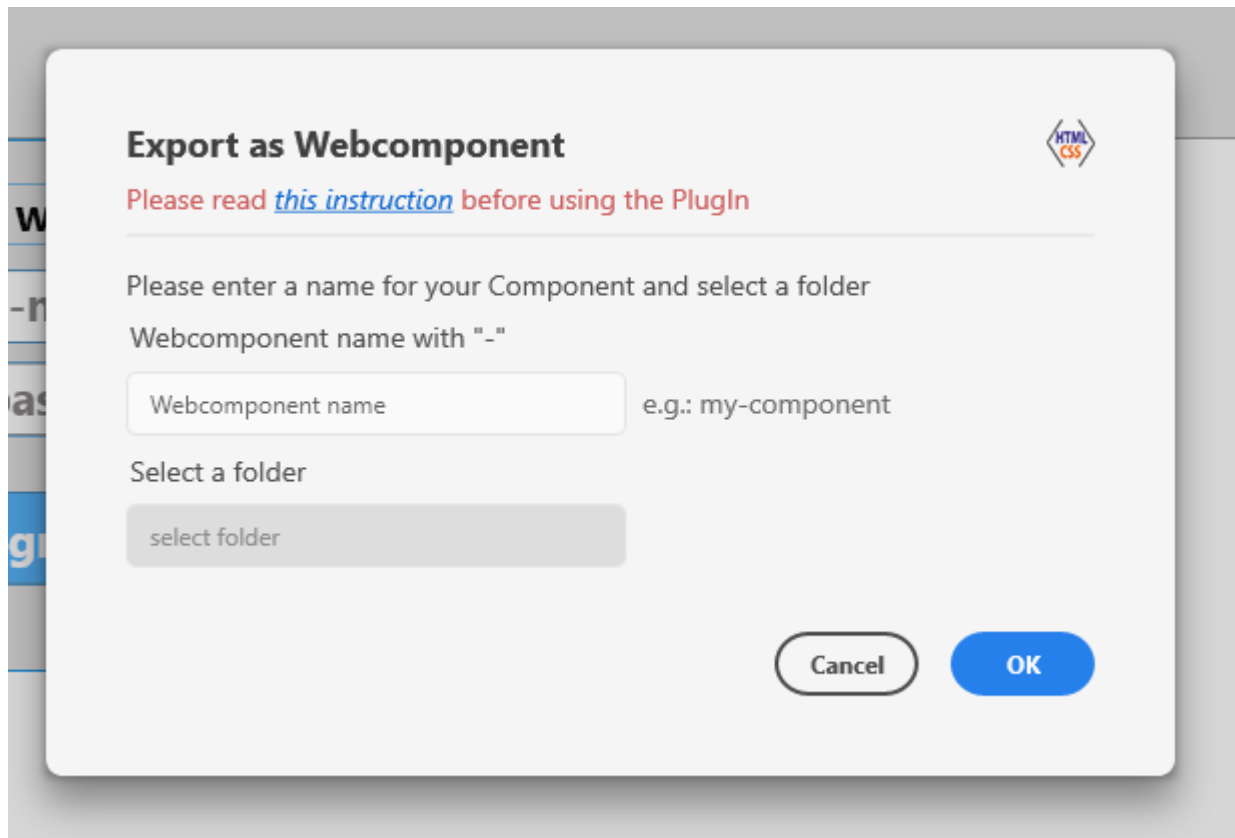


Fig. 9: Plugin pop-up

You will get a short message and you can close the pop-up.

If anything went wrong to will also get a message and you have to check if you made a mistake.
To find common mistakes faster read the FAQs.

How to deal with the exported files

2.1 How to import the exported files into your web application

In your folder, you selected for saving the export, you will find the .js file, the .css file, a index.html file, an other .html file and a material folder, where all your images will be saved.

1. Copy those three things:

the js file, the css file and the material folder

2. Paste them into your web application

ca-
tion

3. Make
sure
you
copy
the
im-
ages
as
well
and not only the folder (otherwise drag the image in the copied folder)

4. Read
the
TO-
DO
in
the
js-
file

5. Place
your
HTML
tag¹
on
the
po-
si-
tion
you
want
to
have
your
com-
po-
nent
on
your
Web-
site

Hint:

If you have the material folder or the .css file in another folder than the .js file
you have to adjust the source link, so that the .js file can finde everything.

¹ with a webcomponent you will create your own tag, so you may have to add the tag to your custom tags before it will work correctly.

2.2 Handle the .js file

In the .js file you will finde:

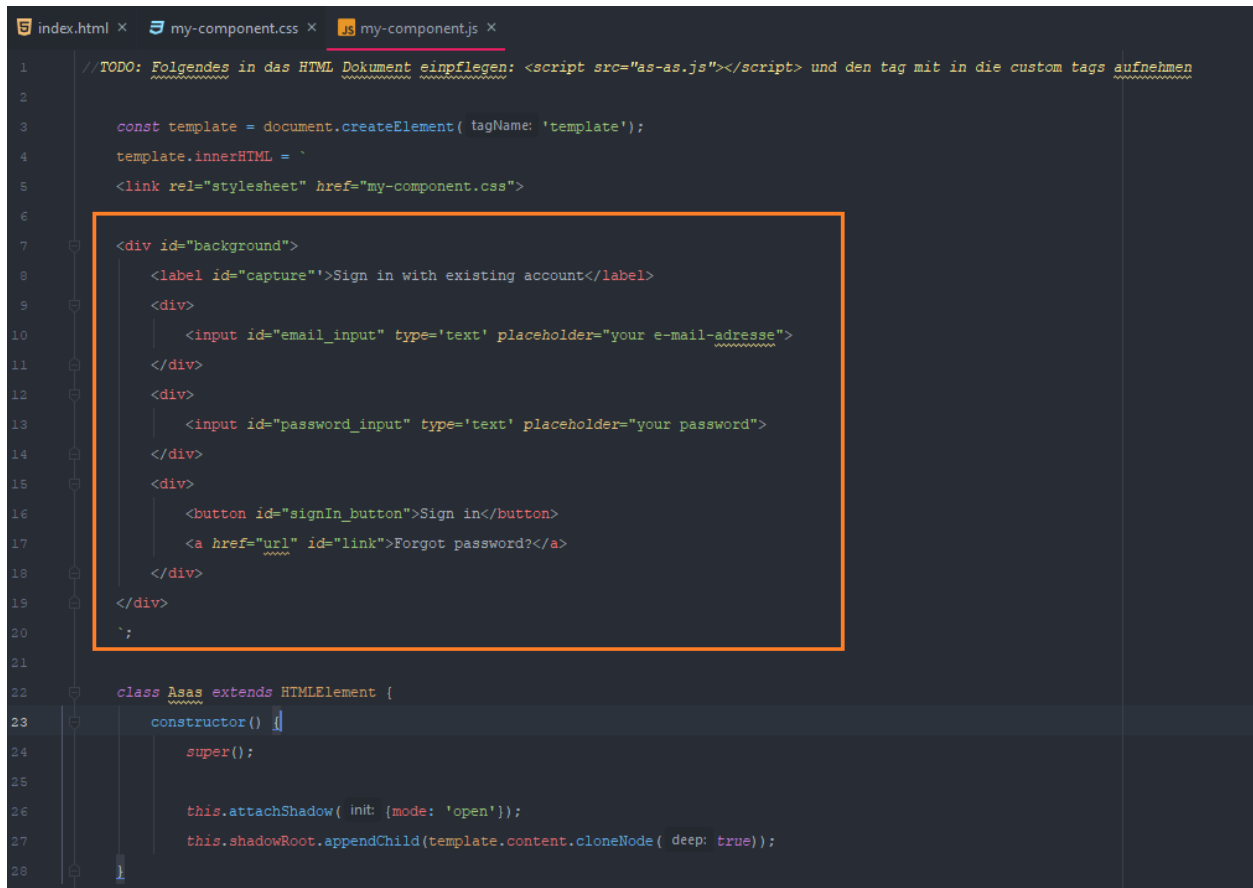
1. The HTML structur of your component

Here you can make changes that will effect the construction and the order of the elements.

Unfortunately you have to format the HTML structur by hand.

Otherwise the indentations aren't correct.

(That would not effect the functionallity, but it's easier to read and simply looks nicer)



```

1 //TODO: Folgendes in das HTML Dokument einpflegen: <script src="as-as.js"></script> und den tag mit in die custom tags aufnehmen
2
3 const template = document.createElement( tagName: 'template');
4 template.innerHTML = `
5   <link rel="stylesheet" href="my-component.css">
6
7   <div id="background">
8     <label id="capture">Sign in with existing account</label>
9     <div>
10       <input id="email_input" type='text' placeholder="your e-mail-adresse">
11     </div>
12     <div>
13       <input id="password_input" type='text' placeholder="your password">
14     </div>
15     <div>
16       <button id="signIn_button">Sign in</button>
17       <a href="url" id="link">Forgot password?</a>
18     </div>
19   </div>
20 `;
21
22 class Asas extends HTMLElement {
23   constructor() {
24     super();
25
26     this.attachShadow( {mode: 'open'});
27     this.shadowRoot.appendChild(template.content.cloneNode( {deep: true}));
28   }

```

Fig. 2: HTML structur in the js file.

2. The component class

Below the html-template you will find the component class, where you can add eventlistener to your buttons or other elements you want to interact with.

- a) call the choosen element by the ID-name and add an eventlistener in the provided connectedCallback-function:

```
connectedCallback() {
  this.shadowRoot.querySelector('#id-name').addEventListener('click', () => {
    this.function();
  });
}
```

b) define your function you called in den lambda-expression:

```
function() {
  // write your code here...
}
```

c) remove the eventlistener in the disconnectedCallback-function.

```
disconnectedCallback() {
  this.shadowRoot.querySelector('#id-name').removeEventListener();
}
```

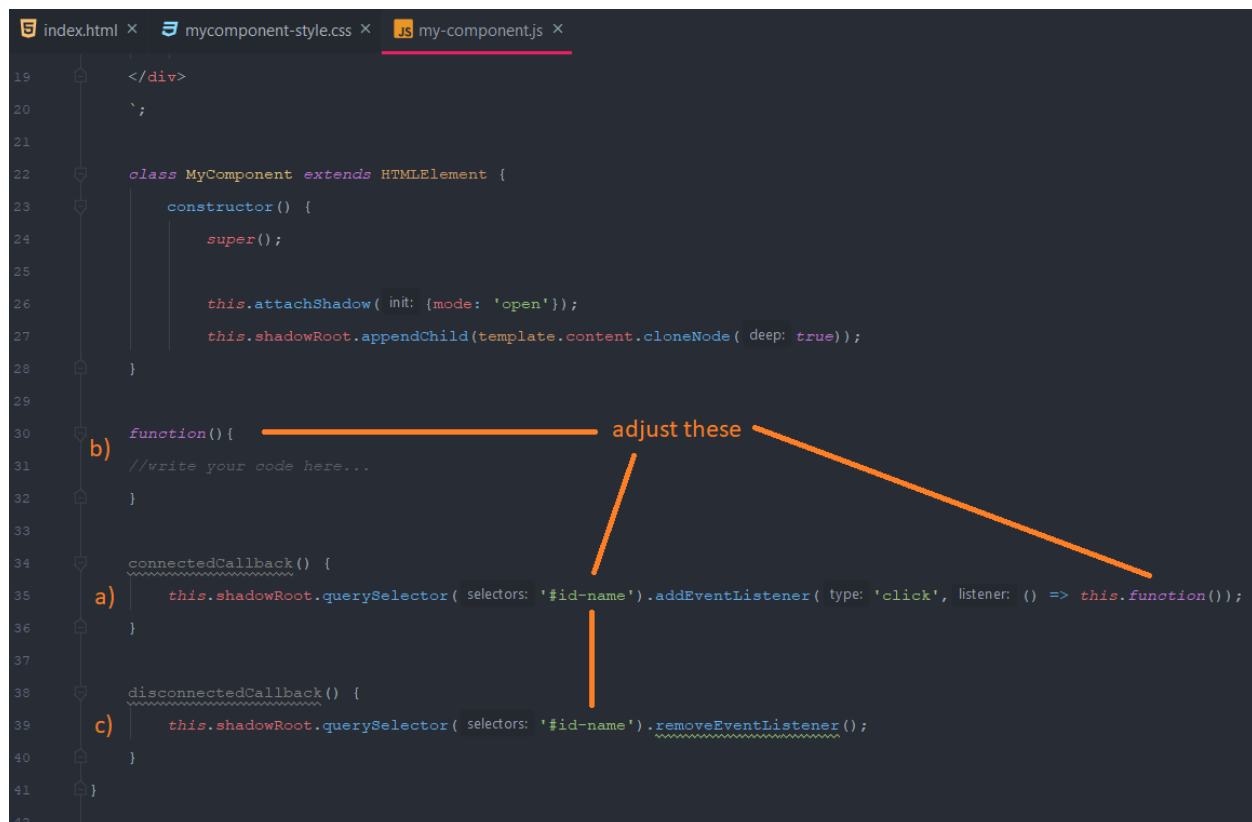


Fig. 3: explanation how to add events.

Hint:

In the exported .js file you will already find this code.

You only have to replace the id-name and the function name.

You can easily call multiple element in the connectedCallback() function.

Just make sure you name the used functions in the evenlistener differently.

2.3 Handle the .css file

The .css file includes all the information about the style for every element.

Here you can make changes, that will effect the appearance of the differen elements.
For format this file automatically press `Ctrlg + Alt + L`

Hint:

If you have elements that have the same look you can combine their style.
There for you have to:

1. change the `id="..."` into a `class="..."` (in the js file -> HTML template)
 2. use the same class name
 3. change the `.` into a `#` (in the css file)
 4. delete the duplicated styles
-

2.4 Why this index.html?

With this index.html file you can view your component before importing it into your application.
Simply doubleclick on it, than the browser will open and you can see your component.

You can also preview all changes you make in the .js or .css file.

2.5 Extra .html file

The extra .html file contains only the hmlt structur of the component.
So you can use this for example in Angular or other projects.

Hint:

This file does not contain the style file (.css file)
For that you have to integrate the css-file.
Simply paste following tag into the html-file and enter the correct css-file-name.
`<link rel="stylesheet" href="*css-file-name*">`

3.1 What are web components?

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps. They will work across modern browsers, and can be used with any JavaScript library or framework that works with HTML.

4.1 1. Why do I not see the filepath?

- Your element names aren't written correctly.
Please check step 2.
- You forgot to select the elements you want to export.
To do this in the right way, please check step 3.
- You didn't name your component correctly / The "-" is missing in the component name.
- Other reasons, in this case please write an e-mail to the support. Thank you.

4.2 2. Why does my export looks different than in Adobe XD?

- Little differences are common.
- Sometimes it's helpful to select the elements and do the export again.
- You selected the elements wrong.
Please check step 3.
- Your element names aren't written correctly, so the styling can't be taken over.
Please check step 2.
- The margin couldn't be calculated correctly.
May your elementboxes are bigger than for example the text.
Please check step 1 / B.
- Your elements overlapping or aren't in one line with eachother.
In this case there is no solution - this arrangement of the elements isn't supported yet.
Please write an e-mail to the support with the subject: "individual arrangement of elements".
- Other reasons, in this case please write an e-mail to the support. Thank you.