Custom JS and HTML

Allows you to complement the functionality of the platform using JS or HTML code.

This lesson explains features of the element and gives an example of using JS for Web Share API in the app.

IMPORTANT NOTES

(1) THE JS SCRIPTS YOU PLACE GO INTO <BODY>

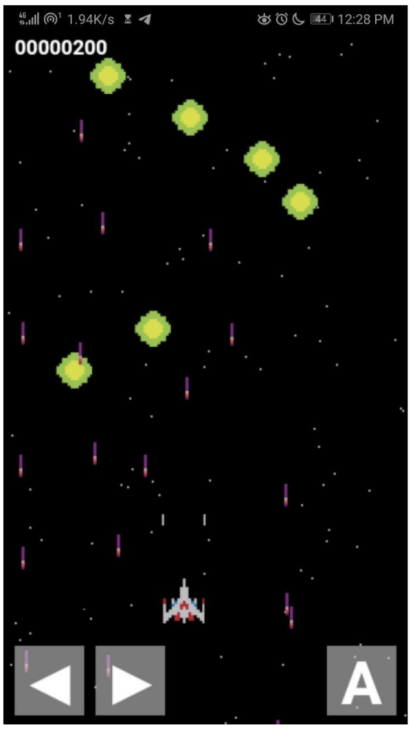
(2) DO NOT USE TAGS *<script type="text/javascript">* AND *</script>*, AS THE SCRIPT ELEMENT ITSELF OPENS AND CLOSES THESE TAGS - ONLY PASTE THE CODE ITSELF

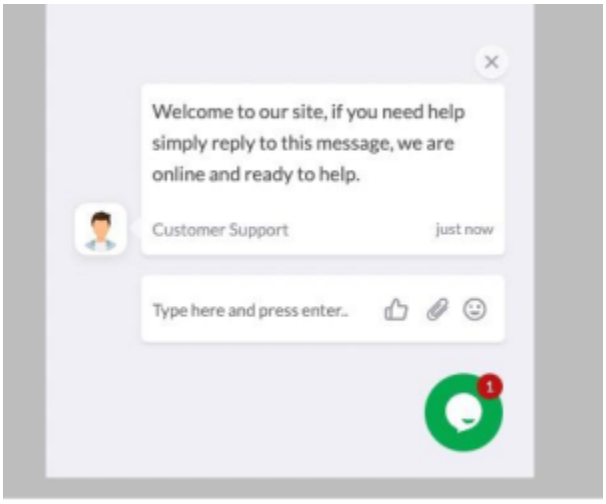
Use cases:

You can use JS code to interact/change/modify/update all of platform's entities, like screen elements, like data points. Use it to achieve ANY result needed for your application.

You can also use it for:

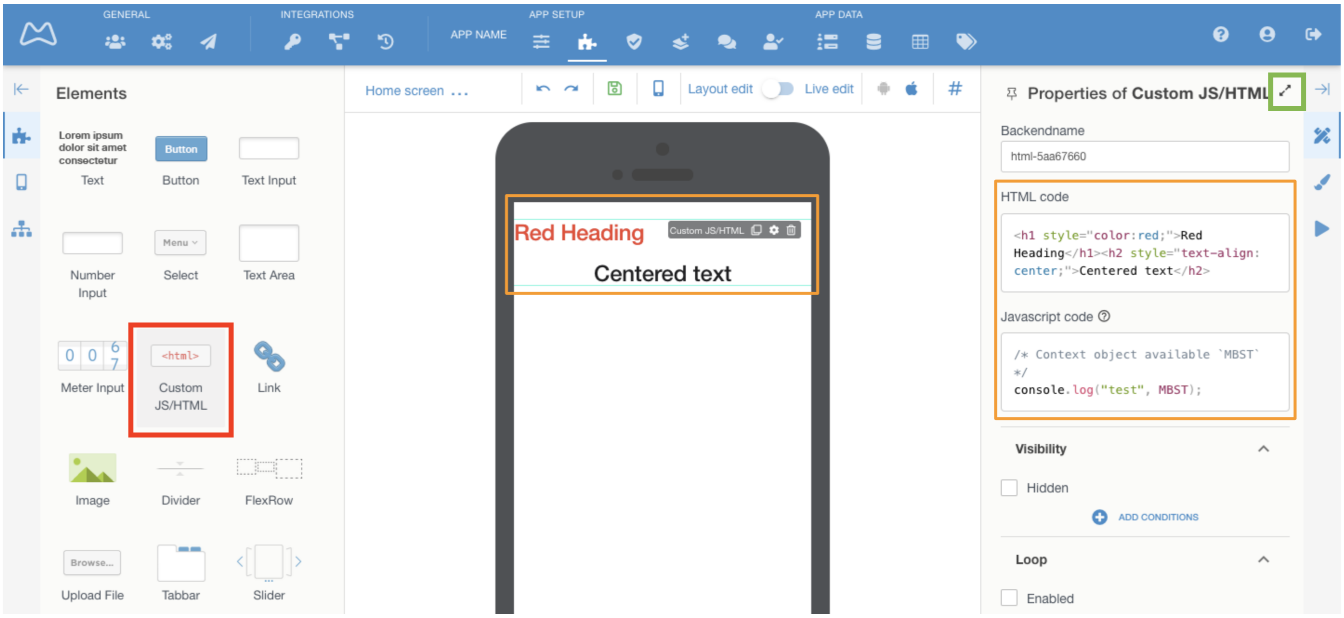
- calling Web APIs, for direct access to hardware
- embedding games or special visual behaviour
- connecting any JS libraries, for example for charts, online chats, counters, calendars etc etc etc.
- etc





How to Use Custom JS/HTML

ADD IT TO THE SCREEN



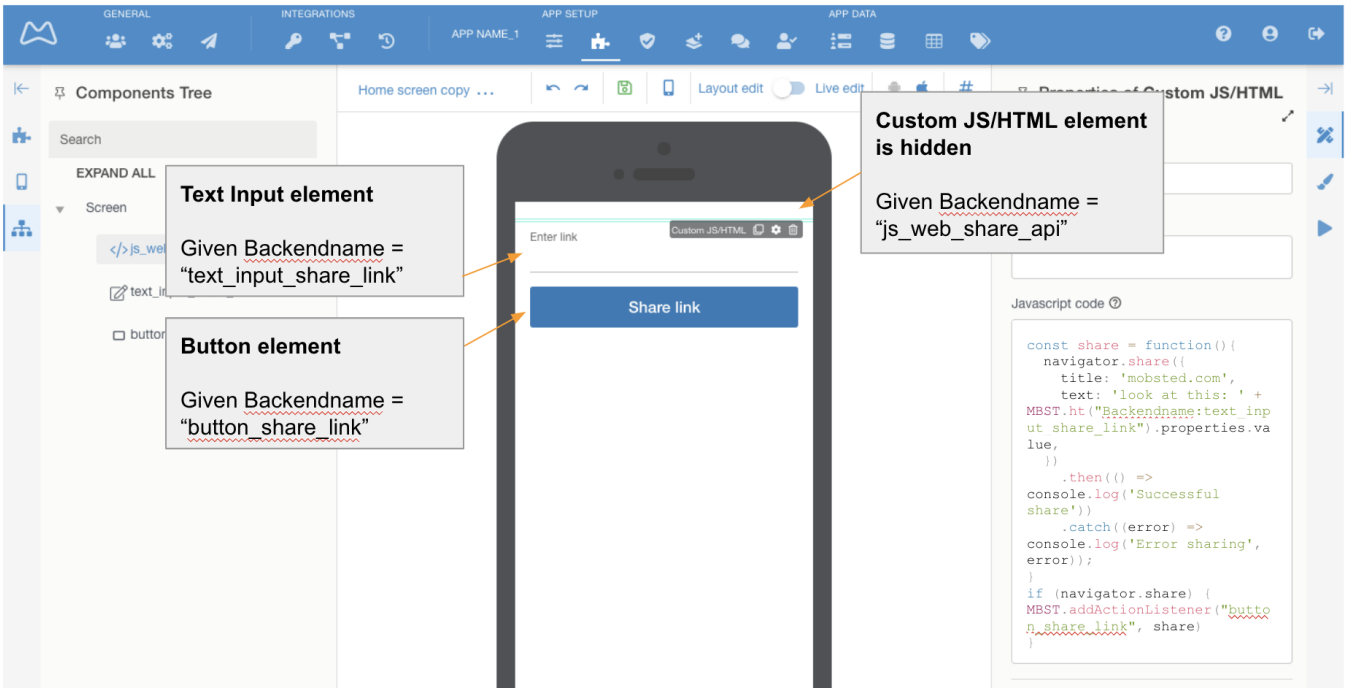
The HTML or JavaScript Element is marked RED, and code inputs, marked ORANGE, located in Properties of the element Custom JS/HTML.

Note: you can expand this area for easier editing with two arrows (GREEN mark)

EXAMPLE - Link sharing using Web API.

How it works in the app:

- User inputs a link,
- clicks Share button,
- chooses an application to share it to.



This is the code, you can copy/paste it for own tests:

```
const share = function(){
  navigator.share({
    title: 'mobsted.com',
    text: 'look at this: ' + MBST.ht("Backendname:text_input_share_link").properties.value,
  })
    .then(() => console.log('Successful share'))
    .catch((error) => console.log('Error sharing', error));
}
if (navigator.share) {
MBST.addActionListener("button_share_link", share)
}
```

RULES FOR JS

Referencing data points using Hashtags:

You can use all data points on the Mobsted platform in your Custom JS. One of the ways to get the value of "text_input_share_link" element is our standard #hashtag# function.

Just change #name# for “name” in JS. Other functions to get the value are described at the end of this lesson.

MBST.ht command is the enactor to the hashtag function. A hashtag can reference:

- element on this screen MBST.ht("Backendname:name")
- object MBST.ht("Object:InviteUrl"
- app MBST.ht("Application:Name")
- variable in a session MBST.ht("Variable:Task_ID")
- filter MBST.ht("EventsFilter:Name_Filter:Data")
- etc

Listener for actions

You can set JS to communicate to elements placed on screen using backend names given:

MBST.****** command allows this.

In our example: MBST.addActionListener("button_share_link", share) - tracks the click on the button called "button_share_link" and calls the share function.

Code usage features:

The code specified in the `**Javascript code**` field will be executed in strict mode (**ES5 "use strict"**).

A JavaScript object is available for interaction with the **Platform**:

```
MBST = {

platform: (string) 'ios|android',

ht: function (path[, value]),

watch: function (path, callback),

addActionListener: function(name, callback),

component: function (name),

action: function ([index], [actionsList])

}
```

FULL LIST OF MBST ACCESS FUNCTIONS

Again, at the constructor you use notation #Variable:test#, but in JS: MBST.ht("Variable:test")

`MBST.platform` - the **'ios'** or **'android'** string indicates the type of device the application is running on.

`MBST.ht(path[, value])` - platform **hashtag read/write** function.

- **Read** - `MBST.ht("Variable:test")`.
- **Write** - `MBST.ht("Variable:test", "new value")`.

`MBST.watch(path, callback)` - tracks changes in a hashtag. The new hashtag value is passed to **callback**.

For example, `MBST.watch("Backendname:text-field", (value) => console.log(value));`.

`MBST.addActionListener(name, callback)` - assigning an event handler of the component on the current app screen, i.e. “button press”.

For example, `MBST.addActionListener("button_share_link", share)`.

`MBST.component(name)` - function to return available component parameters. Available parameters can be viewed using **`console.log(MBST.component("button-test"))`**

`MBST.action([index], [actionsList])` - function that performs a component actions.

- **index** - determines the sequence number of the action to be performed. If no number is set, all actions are performed.
- **actionsList** - defines a list of actions. If not, the list of actions of component 'Custom JS/HTML' itself is used.

⚠ NOTE - To perform actions of a component of the current screen, you need to pass them using the second parameter - “actionsList”. The list of available actions for each component can be obtained by **`MBST.component("button-test").actions`**

- Filters for Objects, Events, Table Lists
- View and Manage Sub-Accounts
- Create Sub-Accounts Manually
- Send Push to Android and iOS
- Create Sub-Accounts by API