JavaScript

Block Group:	Browser API
Block Icon:	

Enables loading and running JavaScript and accessing JavaScript and HTML APIs.

This block is similar to the Script block, but it has two main differences:

- The JavaScript block uses regular JavaScript, which allows it to access JavaScript and HTML APIs directly. The JavaScript block uses `meta` to interact with the JavaScript of the page. `meta` can have two properties, `element` and `gmap`, both of which apply to the JavaScript block's parent object. In contrast, the Script block uses DGLux script, including DGLux5 page model syntax such as `@parent`. Because of this, the JavaScript block is a better choice for accessing APIs and the Script block is a better choice for accessing the DGLux5 page model in script.
- To access the code, the JavaScript block takes a path to a JavaScript file in the project, and the Script block takes script input as a multi-line text property.

The JavaScript is loaded in the global namespace of the page and must meet the following requirements:

- It must include a special `init` function, because the `init` function creates an instance of the script that is differentiated from any other instances that might be running on the page.
- The special `init` function must return a function to be executed when the block is invoked.
- The function returned by the `init` function must take `meta`, `getValue`, and `setValue` as parameters.

If you make changes to the JavaScript code, these changes will take effect only if you refresh the browser—not simply if you close and reopen the `.dg5` file.

In the example:

- `initTestJs` is the special `init` function.
- The `onInvoke` function is returned by the `init` function and is executed when the block is invoked.
- The function to be executed takes `meta`, `getValue`, and `setValue` as parameters.

You can use `getValue` and `setValue` to interact with the custom parameters of the block. These custom parameters appear as properties that you can bind to other properties in DGLux5. The syntax to use `setValue` is `setValue('a', newValue)`. This sets the value of the DGLux5 property **a** to `newValue`. The syntax to use `getValue` is `getValue('a')`. This returns the current value of the DGLux5 property **a**.

Input/Output Properties

These properties can take input and give output.

- invoke *(trigger)*
- autoRun *(boolean)*
- scriptPath *(string)*
- initFunction *(string)*
- custom parameters

invoke causes the JavaScript to be loaded in the global source for the page and runs the `init` function.

autoRun specifies whether the script runs automatically.

- **TRUE**: The script runs every time the script is changed, and every time any property is changed. This includes the first time that the script is initialized.
- **FALSE**: The script runs only when the **invoke** property is triggered.

scriptPath specifies the path to the JavaScript file. It can be a DGLux5 file path relative to the root, or it can be some other URL. The file extension must be `.js`.

initFunction specifies the name of the `init` function in the file.

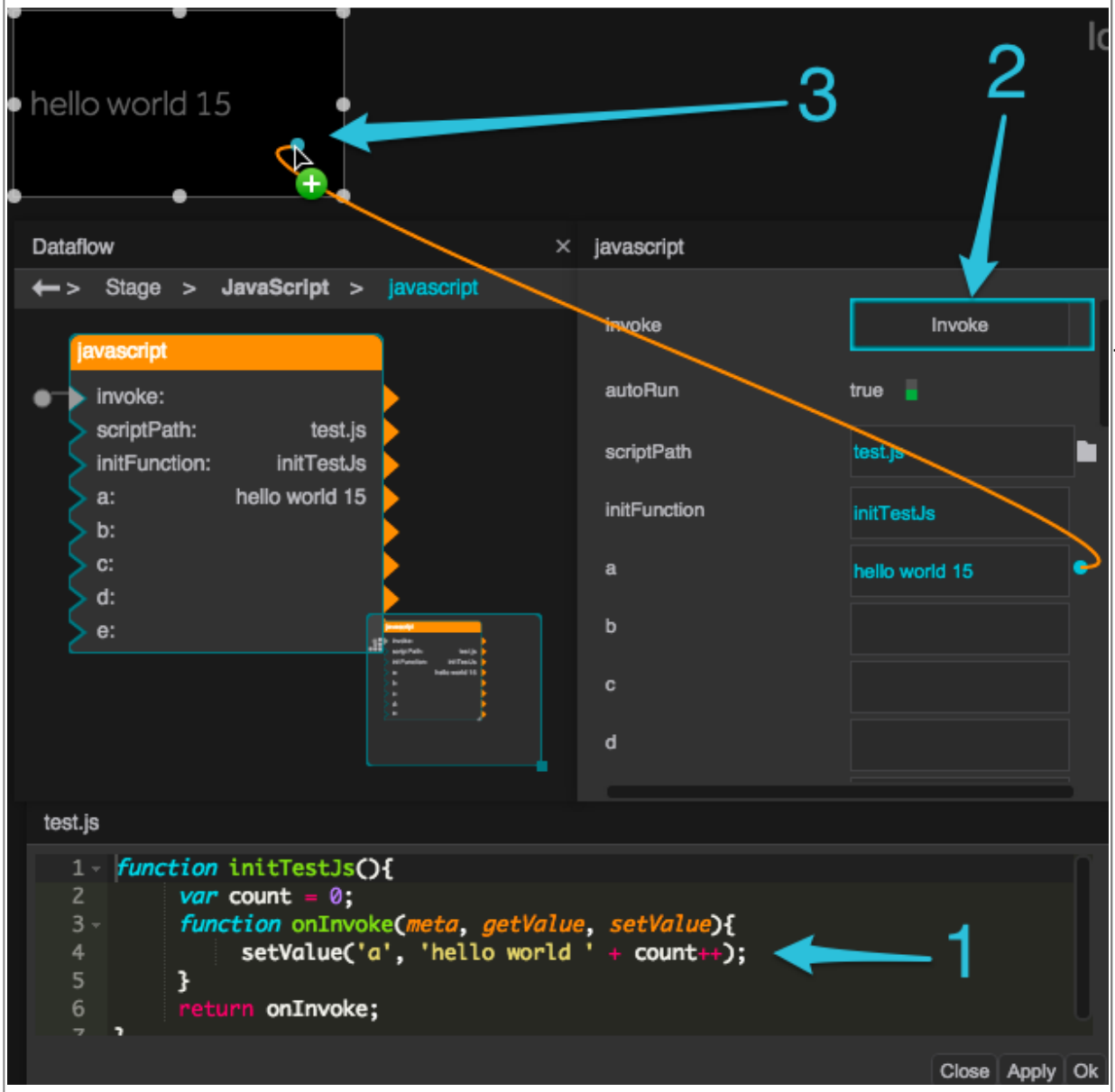
The custom parameters set or return the values of properties that you can use in your script as needed.

Remarks

- You can add a blank `.js` file to your project, and then click  **Edit in Window** in the Details panel to edit the file within DGLux5.
- To test your JavaScript, you must save the DG5 file and reload DGLux5—don't merely close and open the `.dg5` file. Changes to JavaScript take effect only when the page is reloaded.
- If you use the Script block to access Google Maps API, the map's initialization time can cause `meta.gmap` to be `null`. To ensure that this issue is avoided, bind the **onMapInit** event in the map widget's Advanced properties to the **invoke** property of the JavaScript block.
- To create a custom parameter, click the plus sign in the block.
- To delete a custom parameter, right-click it in the block property inspector (the right-hand panel of the dataflow window), and select **Remove Parameter**.
- Bind an event to **invoke**, and bind the custom parameters to other properties in DGLux5.

Model

This is a basic use of the block.

Dataflow Model	Description
	The number displayed in the text component goes up by one each time the component is clicked. To achieve this effect: 1. In your JavaScript file, set the value of the **a** property to the string "hello world *n*", with the value of *n* increasing by one each time the block is invoked. 2. Bind the onClick event for the text component to **invoke**. 3. Bind the value of **a** to the Text property of the text component.

Sample Code

The example on this page uses the following code:

```
function initTestJs(){
     var count = 0;
     function onInvoke(meta, getValue, setValue){
          setValue('a', 'hello world ' + count++);
```

```
     }
     return onInvoke;
}
```

Google Maps API Example

Use this code in the dataflow of a Google Map component to add a marker in Australia on the Google Map component. For **initFunction**, use `initMapForDGLux`.

```
function initMap(map) {
  var myLatLng = new google.maps.LatLng(-25.363, 131.044);

  var marker = new google.maps.Marker({
    position: myLatLng,
    map: map,
    title: 'Hello World!'
  });
}

function initMapForDGLux() {
     function runScript(meta, getValue, setValue) {
          if (meta.gmap != null) {
           initMap(meta.gmap);
          }
     }
     return runScript;
}
```

More Resources

These videos show the JavaScript block in use:

- Using the JavaScript Dataflow Block
- Using the JavaScript Dataflow Block to Access the Google Maps API

Previous: Get View Property

Next: Close Warning Dialog

From:
http://wiki.dglogik.com/ - **DGLogik**

Permanent link:
http://wiki.dglogik.com/dglux5_wiki:dataflow:dataflow_blocks_reference:browser_api:javascript

Last update: **2019/07/17 19:17**