featurereduction.org

Feature Usage Explorer User Guide

Version 1.0

1. System Requirements

Operating System

Feature Explorer supports any operating system, which has a web browser.

Programming language

JavaScript HTML, CSS.

Additional system requirements

Feature Usage Explorer JavaScript library requires 27 kb space.

Dependencies

jquery 1.6: https://ajax.googleapis.com/ajax/libs/jquery/1.6/jquery.js
fabric 0.9.35: https://github.com/kangax/fabric.js/tree/v0.9.35

2. Initializing Feature Usage Explorer

Feature Usage Explorer can be used in any HTML website to monitor and visualize feature usage by adding the feature.usage.explorer.js library in the website of interest. First, jquery and fabric has to be added to the head of the website as shown bellow:

```
<head>
    <script type="text/javascript" src="jquery. js"></script>
    <script type="text/javascript"  src="fabric.js"></script>
</head>
```

Finally, feature.usage.explorer.js library has to be added in body of the website after the canvas element where the feature usage diagram will be visualized:

```
<body>
    <canvas id="canvas" width="2000" height="1000"></canvas>
    <script src="feature.usage.explorer.js"></script>
</body>
```

Note: make sure that canvas has id=”canvas” width and height attributes defined.

Feature Usage Explorer functionalities can be called only after DOM is initialized. Therefore, it is advisable to call all functions inside **$(function(){…})**, which is executed when DOM is ready. There are tree main functionalities implemented by Feature Usage Explorer.

To obtain a complete feature set **explorer.getFeatures(events)** function can be used, where event is any standard JavaScript event (see Section 3 for more details).

To monitor actual feature usage the **explorer.monitorFeatures(events)** has to be binded to any HTML element of interest (see Section 3 for more details).

To fill feature usage diagram with data **explorer.inidDiagram(url)** has to be called (see Section 3 for more details).

The complete working example is as follows:

```
<html>
    <head>
        <script type="text/javascript" src="jquery. js"></script>
        <script type="text/javascript"  src="fabric.js"></script>
    </head>
    <body>
        <canvas id="canvas" width="2000" height="1000"></canvas>
        <script src="feature.usage.explorer.js"></script>
        <script type="text/javascript">
        $(function(){
            var features = explorer.getFeatures ("click dblclick");
            for (var i=0;i<features.length;i++)
                console.log("Feature name: "+ features[i].name);

            $(window).bind("click dblclick", function(event){
                if (explorer.monitorFeatures(event)) {
                    var feature = explorer.monitorFeatures(event);
```

```
                    console.log(" Feature name: "+ feature.name);
                }
            });

            explorer.initDiagram('http://featurereduction.org/json-demo/getjson.php');
        });
        </script>
    </body>
</html>
```

3. Feature Usage Explorer Documentation

Events

Events is a string, which has to be combined by any JavaScript events separated by space. It can contain the following events: click, dblclick, mousedown, mousemove, mouseover, mouseout, mouseup, keydown, keypress, keyup, abort, error, load, resize, scroll, unload, blur, change, focus, reset, select, submit.

User has to specify the events manually in order to save computation power in cases, where only specific events are important. For example, if user is interested only in features, which are clickable, then only **"click"** should be assigned to the Events.

explorer.getFeatures(Events)

This function takes Events as in input and returns an array of features, which have any of the specified event attached to them.

Each feature contains path, name, and element properties. Where path is a full DOM path to the feature, name is a name of the feature, element is a HTML element, which corresponds to the feature.

explorer.monitorFeatures (event)

This function takes triggered event as in input and returns a feature, which was used.

The feature contains path, name, eventType and element properties. Where path is a full DOM path to the feature, name is a name of the feature, eventType is an event type, which was triggered on the feature, element is a HTML element, which corresponds to the feature.

explorer.initDiagram(url)

This function takes url as in input. The url has to contain a json input, which is used to draw a feature usage diagram. The json file has to have the following structure:

The input json file has to be provided by users of Feature Usage Explorer. The json file must have the following structure:

```
[{
        "id": 1,
        "name": "Feature name string",
        "usage": "Number of times feature was used string",
        "type": "Feature",
        "siblings": [],
        "linksOut": [{
                "id": the id of a feature to which this link is pointing to,
                "cardinality": "Number of times a feature was accessed using this path"
        },
        {...}],
```

"linksIn": []

},

{...}]

Below we provide a more detailed description of the aforementioned fields:

id. This field represents a unique identifier of a feature, or a feature group. It can have any integer value.

name. This field represents the name of a feature, or a feature group. It can have any string value.

usage. This field shows how many times a feature was used. It can have any string value.

type. This field can have only two values "feature", or "feature group". If a type is "feature group" then "siblings":[] attribute must be present, which represents features that are in the feature group. The siblings field must have the same attributes as feature: id, name, usage, type, linksOut, linksIn.

linksOut. This field represents outgoing links from a feature ,or a feature group. It can have a multiple links outgoing links.

linksIn. This field represents incoming links to a feature, or a feature group. It can have a multiple incoming links.