NEXT-UI Link colors, tooltips and clickable actions

Let's work with the following application architecture :

```
-   index.html
----app
        -main.js
        -topology.js
----data
         topology_data.js
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" href="../next_libraries/css/next.css">
        <link rel="stylesheet" href="style.css">
        <script type="text/javascript" src="../next_libraries/js/next.js"></script>
</head>
<body>
        <div id="topology-container"></div>
        <!-- Application scripts -->
        <script type="text/javascript" src="data/topology_data.js"></script>
        <script type="text/javascript" src="app/topology.js"></script>
        <script type="text/javascript" src="app/main.js"></script>
</body>
</html>
```

app/main.js

```
(function(nx){
        // instantiate NeXt app
```

```
        var app = new nx.ui.Application();
        // instantiate Topology class
        var topology = new MyTopology();
        // load topology data from app/data.js
        topology.data(topologyData);
        // bind the topology object to the app
        topology.attach(app);
        // app must run inside a specific container. In our case this is the one with id="topology-container"
        app.container(document.getElementById("topology-container"));
})(nx);
```

app/topology.js

```
(function (nx) {
        nx.define('MyTopology', nx.graphic.Topology, {
                methods: {
                        "init": function(){
                                this.inherited({
                                        // width 100% if true
                                        'adaptive': false,
                                        // show icons' nodes, otherwise display dots
                                        'showIcon': true,
                                        // special configuration for nodes
                                        'nodeConfig': {
                                                'label': 'model.name',
                                                'iconType': 'router',
                                                'color': '#0how00'
                                        },
                                        // special configuration for links
                                        'linkConfig': {
```

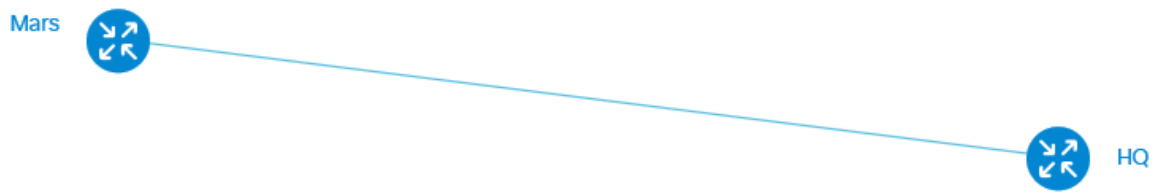
```
                                        'linkType': 'curve'
                                    },
                                    // property name to identify unique nodes
                                    'identityKey': 'id', // helps to link source and target
                                    // canvas size
                                    'width': 1000,
                                    'height': 600,
                                    // "engine" that process topology prior to rendering
                                    //'dataProcessor': 'force',
                                    // moves the labels in order to avoid overlay
                                    'enableSmartLabel': true,
                                    // smooth scaling. may slow down, if true
                                    'enableGradualScaling': true,
                                    // if true, two nodes can have more than one link
                                    'supportMultipleLink': true,
                                    // enable scaling
                                    "scalable": true
                                });
                        }
                }
        });
})(nx);
```

Data/topology_data.js

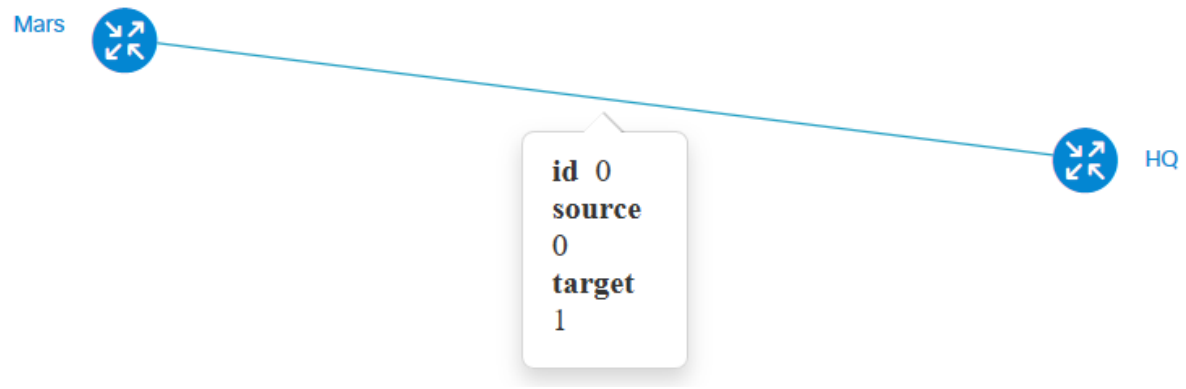
```
var topologyData = {
        "nodes": [
                {
                        "id": 0,
                        "x": 250,
                        "y": -25,
                        "name": "HQ"
```

```
        },
        {
            "id": 1,
             "x": 50,
            "y": -50,
            "name": "Mars"
        }
    ],
    "links": [
        {
            "id": 0,
            "source": 0,
            "target": 1
        }
    ]
};
```

If you open the index.html file with your browser you will see the following.



And if you click on the link you will get the following



Let's change de link color

For doing this we must first add a property in the linkConfig definition of the **app/topology.js** file

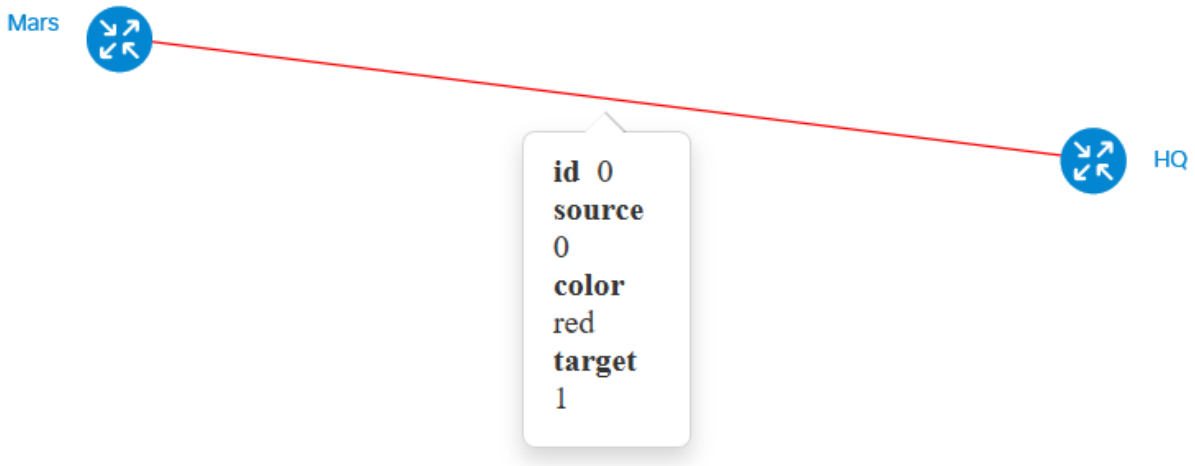
Modify the file this way :

```
// special configuration for links
'linkConfig': {
        'linkType': 'curve',
        'color': "model.color"
},
```

And then add property values in the **data/topology_data.js** file

```
"links": [
        {
                "id": 0,
                "source": 0,
                "color": "red",
                "target": 1
        }
]
```

Then we got the following when we open the **index.html** file



The link's color is now red and when we click on the link we see that the **color** property is displayed.

That means that by default all properties we add in the link definition will be displayed the same way.

Colors

The table here under gives the names you can use

HTML name	HEX	decimal
Red colors		
IndianRed	CD 5C 5C	205 92 92
LightCoral	F0 80 80	240 128 128
Salmon	FA 80 72	250 128 114
DarkSalmon	E9 96 7A	233 150 122
LightSalmon	FF A0 7A	255 160 122
Red	FF 00 00	255 0 0
Crimson	DC 14 3C	220 20 60
FireBrick	B2 22 22	178 34 34
DarkRed	8B 00 00	139 0 0
Pink colors		
Pink	FF C0 CB	255 192 203
LightPink	FF B6 C1	255 182 193
HotPink	FF 69 B4	255 105 180
DeepPink	FF 14 93	255 20 147
MediumVioletRed	C7 15 85	199 21 133
PaleVioletRed	DB 70 93	219 112 147
Orange colors		
LightSalmon	FF A0 7A	255 160 122
Coral	FF 7F 50	255 127 80
Tomato	FF 63 47	255 99 71
OrangeRed	FF 45 00	255 69 0
DarkOrange	FF 8C 00	255 140 0
Orange	FF A5 00	255 165 0
Yellow colors		
Gold	FF D7 00	255 215 0
Yellow	FF FF 00	255 255 0
LightYellow	FF FF E0	255 255 224
LemonChiffon	FF FA CD	255 250 205
LightGoldenrodYellow	FA FA D2	250 250 210
PapayaWhip	FF EF D5	255 239 213
Moccasin	FF E4 B5	255 228 181
PeachPuff	FF DA B9	255 218 185
PaleGoldenrod	EE E8 AA	238 232 170
Khaki	F0 E6 8C	240 230 140
DarkKhaki	BD B7 6B	189 183 107
Purple colors		
Lavender	E6 E6 FA	230 230 250
Thistle	D8 BF D8	216 191 216
Plum	DD A0 DD	221 160 221
Violet	EE 82 EE	238 130 238
Orchid	DA 70 D6	218 112 214
Fuchsia	FF 00 FF	255 0 255
Magenta	FF 00 FF	255 0 255
MediumOrchid	BA 55 D3	186 85 211
MediumPurple	93 70 DB	147 112 219
BlueViolet	8A 2B E2	138 43 226
DarkViolet	94 00 D3	148 0 211
DarkOrchid	99 32 CC	153 50 204
DarkMagenta	8B 00 8B	139 0 139
Purple	80 00 80	128 0 128
Indigo	4B 00 82	75 0 130
DarkSlateBlue	48 3D 8B	72 61 139
SlateBlue	6A 5A CD	106 90 205
MediumSlateBlue	7B 68 EE	123 104 238

HTML name	HEX	decimal
Green colors		
GreenYellow	AD FF 2F	173 255 47
Chartreuse	7F FF 00	127 255 0
LawnGreen	7C FC 00	124 252 0
Lime	00 FF 00	0 255 0
LimeGreen	32 CD 32	50 205 50
PaleGreen	98 FB 98	152 251 152
LightGreen	90 EE 90	144 238 144
MediumSpringGreen	00 FA 9A	0 250 154
SpringGreen	00 FF 7F	0 255 127
MediumSeaGreen	3C B3 71	60 179 113
SeaGreen	2E 8B 57	46 139 87
ForestGreen	22 8B 22	34 139 34
Green	00 80 00	0 128 0
DarkGreen	00 64 00	0 100 0
YellowGreen	9A CD 32	154 205 50
OliveDrab	6B 8E 23	107 142 35
Olive	80 80 00	128 128 0
DarkOliveGreen	55 6B 2F	85 107 47
MediumAquamarine	66 CD AA	102 205 170
DarkSeaGreen	8F BC 8F	143 188 143
LightSeaGreen	20 B2 AA	32 178 170
DarkCyan	00 8B 8B	0 139 139
Teal	00 80 80	0 128 128
Blue/Cyan colors		
Aqua	00 FF FF	0 255 255
Cyan	00 FF FF	0 255 255
LightCyan	E0 FF FF	224 255 255
PaleTurquoise	AF EE EE	175 238 238
Aquamarine	7F FF D4	127 255 212
Turquoise	40 E0 D0	64 224 208
MediumTurquoise	48 D1 CC	72 209 204
DarkTurquoise	00 CE D1	0 206 209
CadetBlue	5F 9E A0	95 158 160
SteelBlue	46 82 B4	70 130 180
LightSteelBlue	B0 C4 DE	176 196 222
PowderBlue	B0 E0 E6	176 224 230
LightBlue	AD D8 E6	173 216 230
SkyBlue	87 CE EB	135 206 235
LightSkyBlue	87 CE FA	135 206 250
DeepSkyBlue	00 BF FF	0 191 255
DodgerBlue	1E 90 FF	30 144 255
CornflowerBlue	64 95 ED	100 149 237
RoyalBlue	41 69 E1	65 105 225
Blue	00 00 FF	0 0 255
MediumBlue	00 00 CD	0 0 205
DarkBlue	00 00 8B	0 0 139
Navy	00 00 80	0 0 128
MidnightBlue	19 19 70	25 25 112

HTML name	HEX	decimal
Brown colors		
Cornsilk	FF F8 DC	255 248 220
BlanchedAlmond	FF EB CD	255 235 205
Bisque	FF E4 C4	255 228 196
NavajoWhite	FF DE AD	255 222 173
Wheat	F5 DE B3	245 222 179
BurlyWood	DE B8 87	222 184 135
Tan	D2 B4 8C	210 180 140
RosyBrown	BC 8F 8F	188 143 143
SandyBrown	F4 A4 60	244 164 96
Goldenrod	DA A5 20	218 165 32
DarkGoldenrod	B8 86 0B	184 134 11
Peru	CD 85 3F	205 133 63
Chocolate	D2 69 1E	210 105 30
SaddleBrown	8B 45 13	139 69 19
Sienna	A0 52 2D	160 82 45
Brown	A5 2A 2A	165 42 42
Maroon	80 00 00	128 0 0
White colors		
White	FF FF FF	255 255 255
Snow	FF FA FA	255 250 250
Honeydew	F0 FF F0	240 255 240
MintCream	F5 FF FA	245 255 250
Azure	F0 FF FF	240 255 255
AliceBlue	F0 F8 FF	240 248 255
GhostWhite	F8 F8 FF	248 248 255
WhiteSmoke	F5 F5 F5	245 245 245
Seashell	FF F5 EE	255 245 238
Beige	F5 F5 DC	245 245 220
OldLace	FD F5 E6	253 245 230
FloralWhite	FF FA F0	255 250 240
Ivory	FF FF F0	255 255 240
AntiqueWhite	FA EB D7	250 235 215
Linen	FA F0 E6	250 240 230
LavenderBlush	FF F0 F5	255 240 245
MistyRose	FF E4 E1	255 228 225
Gray colors		
Gainsboro	DC DC DC	220 220 220
LightGrey	D3 D3 D3	211 211 211
Silver	C0 C0 C0	192 192 192
DarkGray	A9 A9 A9	169 169 169
Gray	80 80 80	128 128 128
DimGray	69 69 69	105 105 105
LightSlateGray	77 88 99	119 136 153
SlateGray	70 80 90	112 128 144
DarkSlateGray	2F 4F 4F	47 79 79
Black	00 00 00	0 0 0

Let's customize the tooltip

The goal now is to customize the tooltips. We want it to have a nice look. Let's insert into it some html formatting stuff.

We will do this into the **app/topology.js** file.

Modify it into the following way.

```
(function (nx) {
        nx.define('MyTopology', nx.graphic.Topology, {
                methods: {
                        "init": function(){
                                this.inherited({
                                        // width 100% if true
                                        'adaptive': false,
                                        // show icons' nodes, otherwise display dots
                                        'showIcon': true,
                                        // special configuration for nodes
                                        'nodeConfig': {
                                                'label': 'model.name',
                                                'iconType': 'router',
                                                'color': '#0how00'
                                        },
                                        // special configuration for links
                                        'linkConfig': {
                                                'linkType': 'curve',
                                                'color': "model.color"
                                        },
                                        // property name to identify unique nodes
                                        'identityKey': 'id', // helps to link source and target
                                        // canvas size
                                        'width': 1000,
                                        'height': 600,
                                        // "engine" that process topology prior to rendering
```

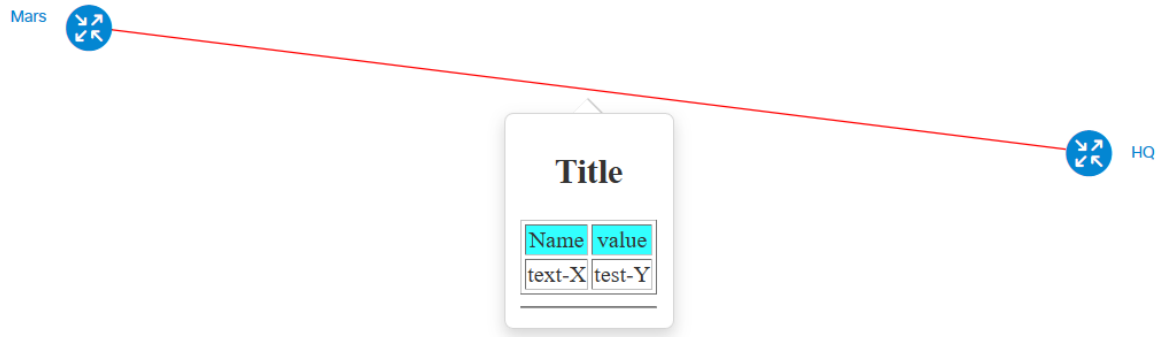
```
                                        //'dataProcessor': 'force',
                                        // moves the labels in order to avoid overlay
                                        'enableSmartLabel': true,
                                        // smooth scaling. may slow down, if true
                                        'enableGradualScaling': true,
                                        // if true, two nodes can have more than one link
                                        'supportMultipleLink': true,
                                        // enable scaling
                                        "scalable": true,
                                        tooltipManagerConfig: {
                                                linkTooltipContentClass: 'LinkTooltip',
                                        }
                                });
                        }
                }
        });
        nx.define('LinkTooltip', nx.ui.Component, {
                        properties: {
                                link: {},
                                topology: {}
                        },
                        view:
                        {
                                content:
                                [
                                        {
                                                // display a centered title
                                                tag:"h2",
                                                props:
                                                {
                                                        "align":"center",
```

```
            },
            content:"Title"
        },
        {
            // display a table
            tag: "table",
            props :
            {
                // html tag table properties
                "border" : "1"
            },
          content: [
             {
                tag: "thead",
                props :
                {
                    // html tag thead properties
                    "align": "center",
                    "bgcolor" : "aqua"
                },

                content:
                {
                  tag: "tr",
                  content: [
                    {
                      tag: "td",
                      content:"Name"
                    },
                    {
                      tag: "td",
```

```
                                        content: "value"
                                    }
                                ]
                            }
                        },
                        {
                            tag: "tbody",
                            name : "tbody",
                            tag: "tr",
                            content:
                            [
                                {
                                    tag: "td",
                                    content: "text-X"
                                },
                                {
                                    tag: "td",
                                    content: "test-Y"
                                }
                            ]
                        }
                    ]
                },
                {
                    // display an horizontal line
                    tag:"hr"
                ]
        });
})(nx);
```

Then open the **index.html** file. Click on the You should have the following result



Now Let's add a clickable button into the tooltip

Modify the **app/topology.js** file at the bottom of this file.

```
(function (nx) {
        nx.define('MyTopology', nx.graphic.Topology, {
.....
.....
......
        });

        nx.define('LinkTooltip', nx.ui.Component, {
                properties: {
                        link: {},
                        topology: {}
                },
                view:
                {
                        content:
                        [
                                {
                                        // display a centered title
                                        tag:"h2",
                                        props:
                                        {
                                                "align":"center",
```

```
            },
            content:"Title"
        },
        {
            // display a table
            tag: "table",
            props :
            {
                // html tag table properties
                "border" : "1"
            },
          content: [
             {
                tag: "thead",
                props :
                {
                    // html tag thead properties
                    "align": "center",
                    "bgcolor" : "aqua"
                },

                content:
                {
                  tag: "tr",
                  content: [
                    {
                      tag: "td",
                      content:"Name"
                    },
                    {
                      tag: "td",
```

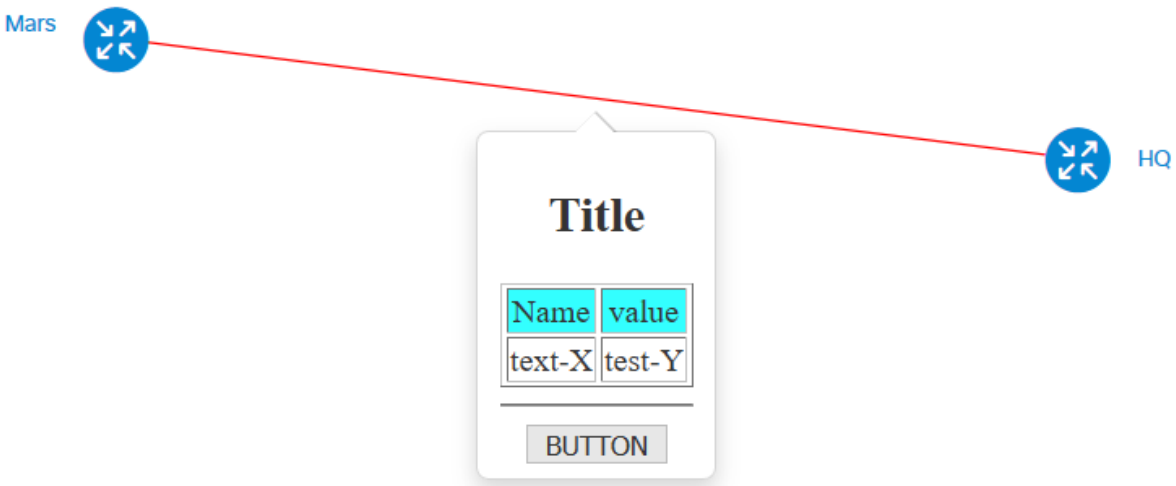
```
                content: "value"
                }
            ]
            }

        },
        {
            tag: "tbody",
            name : "tbody",
            tag: "tr",
            content:
            [
                {
                tag: "td",
                content: "text-X"
                },
                {
                tag: "td",
                content: "test-Y"
                }
            ]

        }
    ]
},
{
    // display an horizontal line
    tag:"hr"
},
{
    tag:'form',
    props:{
```

```
                                                "action":"https://www.google.com"
                                        },
                                        "content":
                                        {
                                                // display a centered button
                                                tag:"center",
                                                "content":{
                                                        tag:"button",
                                                        "props": {
                                                                "type": "submit",
                                                        },
                                                        "content":"BUTTON"
                                                }
                                        }
                                }
                        ]
                },
                methods: {
                        // we will see later why we need the following
                        _click: function (sender, events) {
                                var link = sender.model().edge;
                        }
                }
        });
})(nx);
```

Then the result becomes



If you click on the button, you will get access to www.google.com

Trigger an action on click on the link

Now let's see how we can trigger an action when on click on the link.

First modify the **app/main.js** file :

```
(function(nx){
        nx.define('ExtendedScene', nx.graphic.Topology.DefaultScene, {
                methods: {
                        clickLink: function(sender, link){
                                this.inherited(sender, link);
                                alert("hello!"+ link.id() );
                                popup = window.open( 'https://www.google.com', '1', 'toolbar=no,scrollbars=no,location=no,statusbar=no,menubar=no,resizable=yes,width=500,height=550' );
                                popup.focus();
                        }
                }
        });
        // instantiate NeXt app
        var app = new nx.ui.Application();

        // instantiate Topology class
        var topology = new MyTopology();
```

```
        topology.on("ready", function(){
                // load topology data from app/data.js
                topology.data(topologyData);
                topology.registerScene('extended-scene', 'ExtendedScene');
                topology.activateScene('extended-scene');
        });

        // load topology data from app/data.js
        topology.data(topologyData);

        // bind the topology object to the app
        topology.attach(app);

        // app must run inside a specific container. In our case this is the one with id="topology-container"
        app.container(document.getElementById("topology-container"));

})(nx);
```

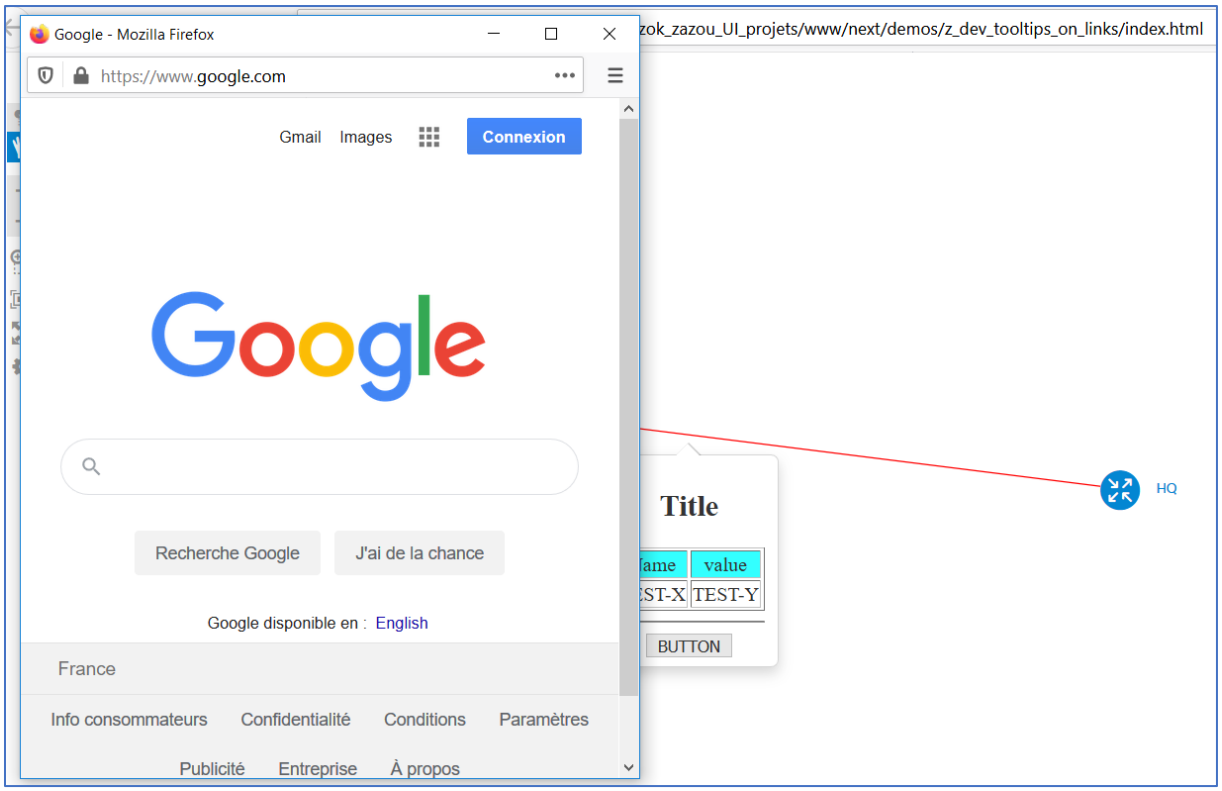
And then let's modify the **index.html** file in order to manage to open a new html windows as a popup

Index.html

```
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" href="../next_libraries/css/next.css">
        <link rel="stylesheet" href="style.css">
        <script type="text/javascript" src="../next_libraries/js/next.js"></script>
        <script type="TEXT/javascript">
```

```
        <!--
   function popup_window( url, id, width, height )
   {
      //extract the url parameters if any, and pass them to the called html
      var tempvar=document.location.toString(); // fetch the URL string
      var passedparams = tempvar.lastIndexOf("?");
      if(passedparams > -1)
         url += tempvar.substring(passedparams);
     popup = window.open( url, id,
'toolbar=yes,scrollbars=yes,location=no,statusbar=no,menubar=yes,resizable=yes,width=' + width
+ ',height=' + height + '' );
     popup.focus();
   }
        function goto(liste)
        {
                if(liste.action.selectedIndex != 0)
                location = liste.action.options[liste.action.selectedIndex].value;
        }
        -->
        </script>
</head>
<body>
        <div id="topology-container"></div>
        <!-- Application scripts -->
        <script type="text/javascript" src="data/topology_data.js"></script>
        <script type="text/javascript" src="app/topology.js"></script>
        <script type="text/javascript" src="app/main.js"></script>
</body>
</html>
```

Try it now. Click on the link. You should get the following result



We have now an interesting link clickable actions. We can replace https://www.google.com by any other location like a PHP / python script, or an api_path etc…

And the do some complex computation. But in this case, we will need a Backend Web Werver underneath our NEXT-UI application. We will see some examples in another tutorial

Additionnal resources

- https://d1nmyq4gcgsfi5.cloudfront.net/site/neXt/discover/demo/
  - Chapter : Highlight Link and Node
  - Chapter : Disable node or link
  - Chapter : Customize Node and Link style
  - Chapter : Link tooltip
  - Chapter : Extend Link