



LIQUID.JS

The Web Conference 2018
Developers' Track

Andrea Gallidabino Cesare Pautasso

Software Institute
Faculty of Informatics
USI Lugano, Switzerland

LIQUID.JS

ONE USER, MANY DEVICES

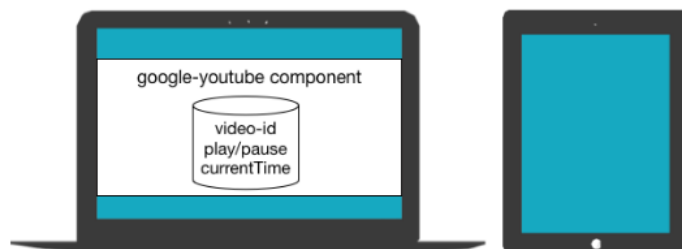


Liquid = Flow + Adapt

LIQUID.JS

LIQUID USER EXPERIENCE

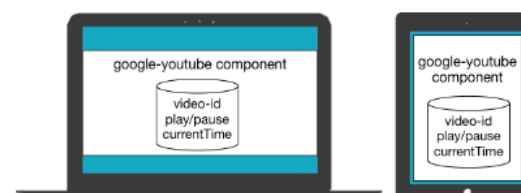
Initial Configuration



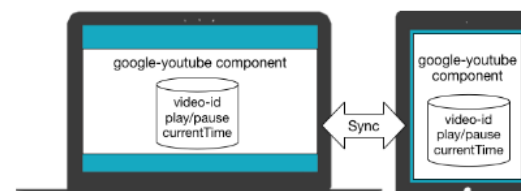
Migrate



Fork



Clone



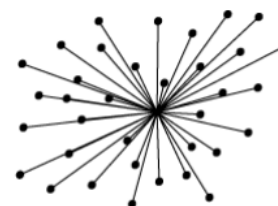
LIQUID.JS

CROSS-DEVICE WEB APPS

Multiple device dynamic
deployment



Decentralised



centralised



decentralised



Component-based



DEMO

Liquid google-youtube component

STEP 1: REUSE WEBCOMPONENTS REPOSITORY



WEBCOMPONENTS.ORG

Discuss & share web components

Getting started

Community

Chat

Publish element

Q google-youtube

EMOJI

FORM

BUTTON

IMAGE

LAYOUT

MEDIA



by GoogleWebComponents



105



46



20



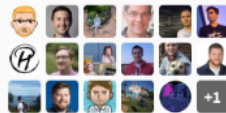
16

LICENSED UNDER [APACHE-2.0](#), LAST
UPDATED 8 MONTHS AGO, &
INSTALLED VIA [BOWER](#)

DOWNLOAD

```
bower install --save  
GoogleWebComponents/google-  
youtube
```

19 CONTRIBUTORS



LAST UPDATED 8 MONTHS AGO

6/2017

4/2018

google-youtube v2.0.0

YouTube video playback web component

Element <google-youtube>

class extends HTMLElement

Path: google-youtube.html

Description

`google-youtube` encapsulates the YouTube player into a web component.

```
<google-youtube  
  video-id="..."  
  height="270px"  
  width="480px"  
  rel="0"  
  start="5"  
  autoplay="1">  
</google-youtube>
```

STEP 2: LIQUID BEHAVIOR

Load liquidAPI.js script and configure the library:

```
<script src="/corescripts/liquidAPI.js"></script>
<script>
  Liquid.configure({
    host: 'localhost',
    secure: true
  })
</script>
```

Import the Liquid Behavior into the component:

```
behaviors: [LiquidBehavior],
```



DEMO

STEP 3: API CONSOLE

Create components:

```
Liquid.createComponent('google-youtube')
```

Get component URIs:

```
Liquid.getAllComponentURIs()
```

Get component instance:

```
Liquid.getComponentInstance('liquid://device/componentType/componentRef') // 'URI'
```

Get connected devices:

```
Liquid.getDeviceList()  
Liquid.getDeviceList(true)
```

Migrate Component:

```
Liquid.migrateComponent('componentURI', Liquid.getDeviceList(true)[0])
```



DEMO

STEP 4: MAKE STATE LIQUID

Annotate videoID property:

```
videoID: {  
  value: '848QV4H0IDE',  
  liquid: true  
},
```

DEMO

STEP 5: LIQUID UI ACTIVATION

Add a component header:

```
<liquid-component-google-youtube liquidui="www"></liquid-component-google-youtube>
```

Instantiate the library core-toolbar component:

```
Liquid.configure({  
  host: 'localhost',  
  secure: true  
})  
.then(function() {  
  return Liquid.createComponent('core-toolbar')  
})
```

DEMO

STEP 6: STATE AND TIME SYNC

Create and annotate the properties:

```
state: {  
  value: 0,  
  liquid: true,  
  observer: "_onStateChange"  
},  
currenttime: {  
  value: 0,  
  liquid: true,  
  observer: "_onCurrentTimeChange"  
},  
duration: {  
  liquid: true  
}
```

Bind properties to google-youtube component:

```
<google-youtube  
  id="myVideo"  
  video-id="{{videoID}}"  
  height="320px" width="600px" chromeless="0"  
  state="{{state}}"  
  on-google-youtube-ready="_onPlayerReady"  
  currenttime="{{currenttime}}"  
  duration="{{duration}}"  
></google-youtube>
```





DEMO

STEP 7: SEND VIDEO TO SLIDES

Watch demo on screen



DEMO

STEP 8: COMPLEMENTARY VIEW

Add the liquid-component-slider:

```
<liquid-component-slider  
  liquidui="www"  
  currenttime={{currenttime}}  
  duration="{{duration}}">  
</liquid-component-slider>
```

Add the Liquid Container behavior:

```
behaviors: [LiquidBehavior, LiquidContainer],
```

Preload the component in the ready function:

```
ready: function () {  
  Liquid.loadComponent('slider')  
},
```



FRAMEWORK CONFIGURATION

Method Name	Return Value
<code>configure(options)</code>	<code>Promise()</code>
<code>getLoadableComponents()</code>	<code>Promise(componentTypes[])</code>
<code>getAllComponentURIs(DeviceURI)</code>	<code>Promise(componentURIs[])</code>
<code>getAllComponentInstances()</code>	<code>componentInstance[]</code>
<code>getComponentInstance(componentURI)</code>	<code>componentInstance</code>
<code>getDevicename</code>	<code>devicename</code>
<code>setDevicename(devicename)</code>	-

Events

`devicenameChange(devicename)`

Triggers when the server accepts the notification of the devicename change.

`loadableComponentsListUpdate(componentTypes[])`

Triggers when the available list of components on the server side changes and returns a list of `componentTypes[]`. The list keeps track only of the components available on the server, it does not check for components available on the clients.

LIQUID USER EXPERIENCE API

Method Name	Return Value
<code>migrateComponent(sourceCompURI, targetDevURI)</code>	<code>Promise(migratedComponentURI)</code>
<code>forkComponent(sourceCompURI, targetDevURI)</code>	<code>Promise(forkedComponentURI)</code>
<code>cloneComponent(sourceCompURI, targetDevURI)</code>	<code>Promise(clonedComponentURI)</code>

DEVICE DISCOVERY API

Method Name	Return Value
<code>getDeviceInfo()</code>	<code>deviceInfo</code>
<code>getDeviceURI()</code>	<code>deviceURI</code>
<code>getDevicesList()</code>	<code>availableDeviceURIs[]</code>
<code>getDevicesInfoList()</code>	<code>availableDeviceInfos[]</code>

Events

`devicesListUpdate(availableDeviceInfos[])`

Triggers when a new device connects to the server and is available to be paired. The callback returns the entire list of {deviceURI, clientjsFingerPrint, devicename, customData} of device connected.



COMPONENT LIFECYCLE API

Method Name	Return Value
<code>loadComponent(componentTypeURI)</code>	<code>Promise(componentTypeURI)</code>
<code>createComponent(componentType[,DeviceURI, DOMElem, UIType])</code>	<code>Promise(createdComponentURI)</code>
<code>registerComponent(componentURI)</code>	<code>Promise(componentURI)</code>
<code>instantiateComponent(compType[,DeviceURI, DOMElem, UIType])</code>	<code>Promise(instantiatedComponent)</code>
<code>deleteComponent(componentURI)</code>	<code>Promise()</code>

LIQUID STORAGE API

Method Name	Return Value
<code>getState(componentURI)</code>	<code>Promise(stateSnapshot)</code>
<code>setState(componentURI, stateSnapshot)</code>	<code>Promise(componentURI)</code>
<code>pairComponent(sourceCompURI1, targetCompURI2)</code>	<code>Promise()</code>
<code>unpairComponent(sourceCompURI1, targetCompURI2)</code>	<code>Promise()</code>
<code>pairProperty(sourcePropURI, targetPropURI)</code>	<code>Promise()</code>
<code>unpairProperty(sourcePropURI, targetPropURI)</code>	<code>Promise()</code>

COMPUTATION OFFLOAD API

Method Name	Return Value
<code>createLiquidWorker(workerName, workerURI)</code>	<code>Promise(worker)</code>
<code>createLiquidWorkerArray({workerName, workerURI}[])</code>	<code>Promise(workers[])</code>
<code>pairDeviceWorkers(DeviceURI)</code>	<code>Promise(DeviceURI)</code>
<code>postLiquidWorkerMessage(workerName, message)</code>	<code>Promise(callResponse)</code>
<code>terminateLiquidWorker(workerName)</code>	<code>Promise(workerName)</code>

LIQUID WORKER API

Method Name	Return Value
<code>postMessage(message)</code>	<code>Promise(callResponse)</code>
<code>_postMessage(message)</code>	<code>Promise(callResponse)</code>
<code>terminate()</code>	<code>Promise()</code>



ASSETS (P2P) API

Method Name	Return Value
<code>requestAsset(name, deviceURI, type)</code>	<code>Promise(script)</code>
<code>loadAsset(name, script)</code>	<code>Promise(name)</code>
<code>getAsset(name)</code>	<code>script</code>
<code>getLoadedAssets()</code>	<code>names[]</code>

Events

`loadingChange(status)`

Triggers when the Liquid.js library starts or ends fetching assets from the server or the client. The event carries the current loading status: true if Liquid.js is fetching or loading an asset, false otherwise

DEVICE CONNECTION API

Method Name	Return Value
<code>isSocketConnected()</code>	<code>isSocketConnected</code>
<code>sendSocketCustomMessage(msg)</code>	-
<code>socketDisconnect()</code>	-
<code>connectDevice(deviceURI)</code>	<code>Promise(deviceInfo)</code>
<code>disconnectDevice(deviceURI)</code>	<code>Promise()</code>
<code>sendMessage(message)</code>	-
<code>getConnectionList()</code>	<code>connectedList</code>

Events

`connect(deviceID)`

Triggers when the client connects to the server for the first time.

`disconnect()`

Triggers when the client disconnects from the server.

`reconnect()`

Triggers when the client reconnects to a server, this event is triggered instead of the connected event.

Custom Events

It is possible to use the socket connection opened by Liquid.js to communicate with the server side or allow the server to push custom data to the client. By sending a `socketCustom` message defined as `{type, payload}`, Liquid.js fires an event "type" with parameter "payload" on the receiving client.

LOCAL PERSISTENCE API

Method Name	Return Value
saveDeviceState(key)	Promise(key)
loadDeviceState(key)	Promise(key)
getAllDeviceState()	Promise(deviceStateSummaries[])
getDeviceState(key)	Promise(deviceStateSummary)
Method Name	Return Value
saveComponentState(key, componentURI)	Promise(key)
loadComponentState(key)	Promise(key)
getAllComponentState()	Promise(componentStateSummaries[])
getComponentState(key)	Promise(componentStateSummary)
Method Name	Return Value
savePropertyState(key, propertyURI)	Promise(key)
getAllPropertyState()	Promise(propertyStateValues[])
getPropertyState(key)	Promise(propertyStateValue)

TECHNOLOGIES

Client-side



Server-side



THANK YOU

liquid.inf.usi.ch

Contact me:



[Andrea Gallidabino](#)

[Software Institute](#)
Faculty of Informatics
USI Lugano, Switzerland

andrea.gallidabino@usi.ch