WebOS-Hackathon

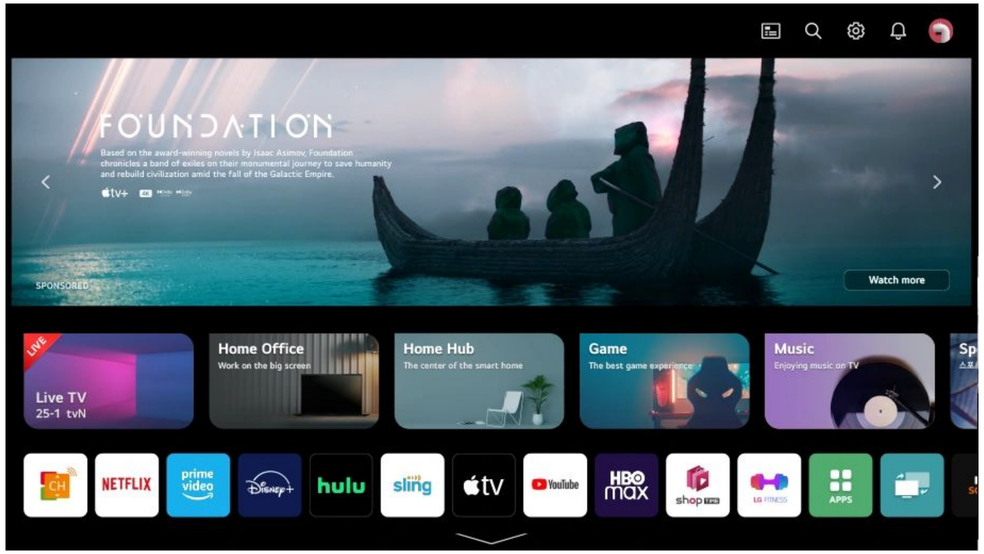
WebOS-Hackathon

- This guide will show you how you can create your own webos-tv app.

- Written guides and example app codes are provided in our Github link https://github.com/youngheoncho/webos-hackathon

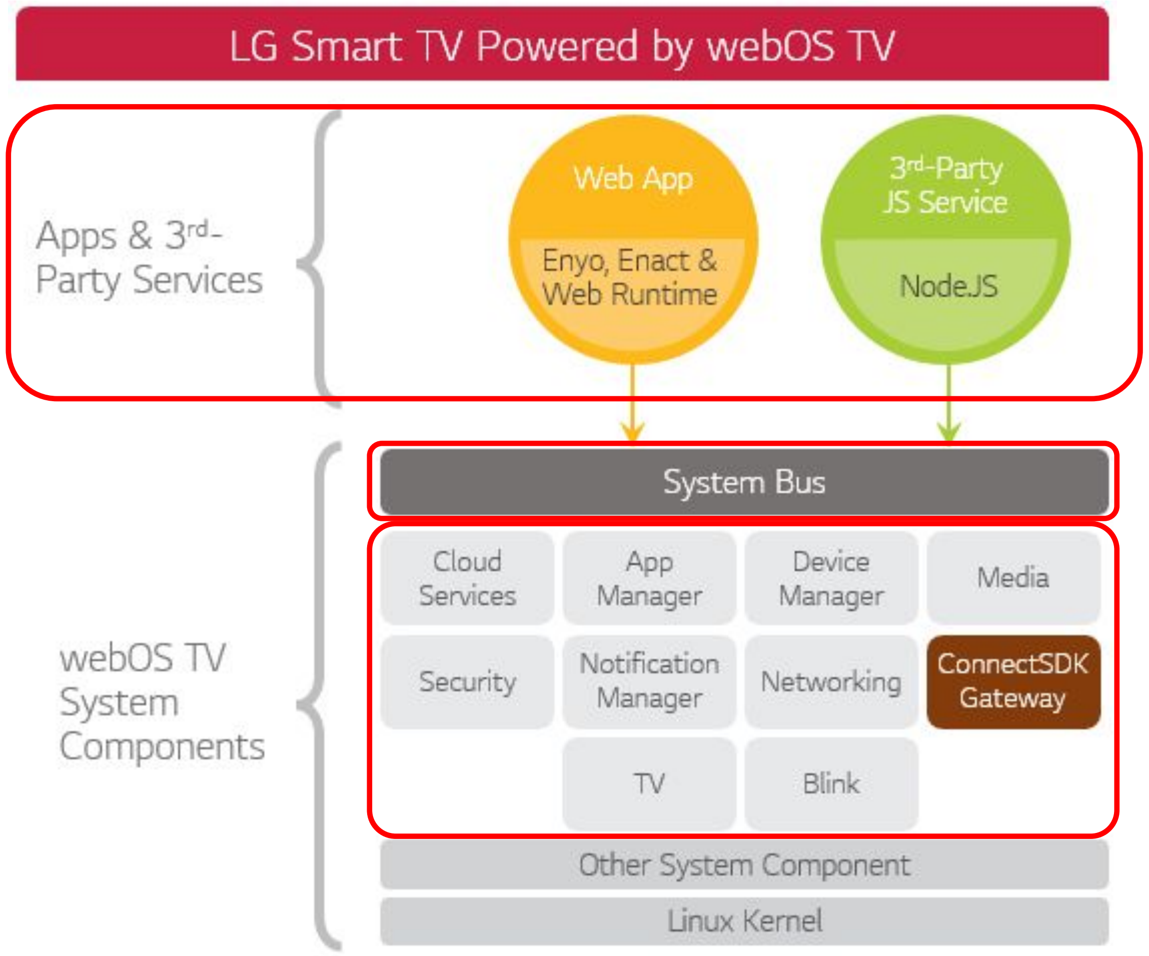
Table Of Contents

What is WebOS-TV



- WebOS-TV is a web-centric smart TV platform that has powered LG Smart TVs for over a decade.
- Linux-based smart display platform with reliable and stable performance

WebOS-Architecture



WebOS TV System Components

- WebOS TV is based on Linux, and its core contains a rich set of essential services that enable apps to manage media, devices, security, networking, TV functionalities, and more.
- System Bus(Luna Bus) is a channel through which apps communicate with WebOS services, such as App Manager, Media, and Notification Manager
- Apps can also access custom services, provided by 3rd party developers, through the System Bus.

WebOS-Architecture

Web Apps

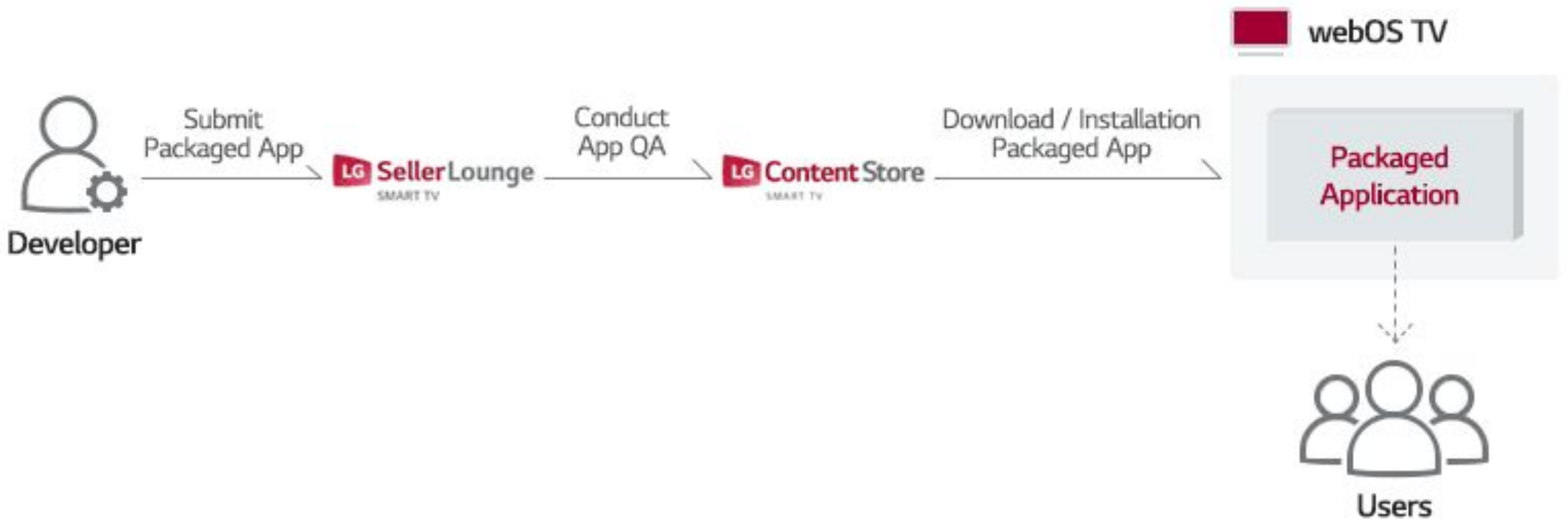
- Web apps built for WebOS TV are basically similar to standard web apps.
- You can create web apps for WebOS TV using standard-based web technologies like HTML, CSS, and JavaScript.
- There are two types of web apps
  - Packaged Web Apps(Basic web app)
  - Hosted Web Apps.

JavaSript Services

- WebOS TV JavaScript services are created using Node.js.
- JavaScript services can perform background processing, low-level networking, access to the file system, and more.
- Apps can access the WebOS JavaScript services through the WebOS TV System Bus.

1. First WebOS-TV App

Basic Web App



Basic Web App

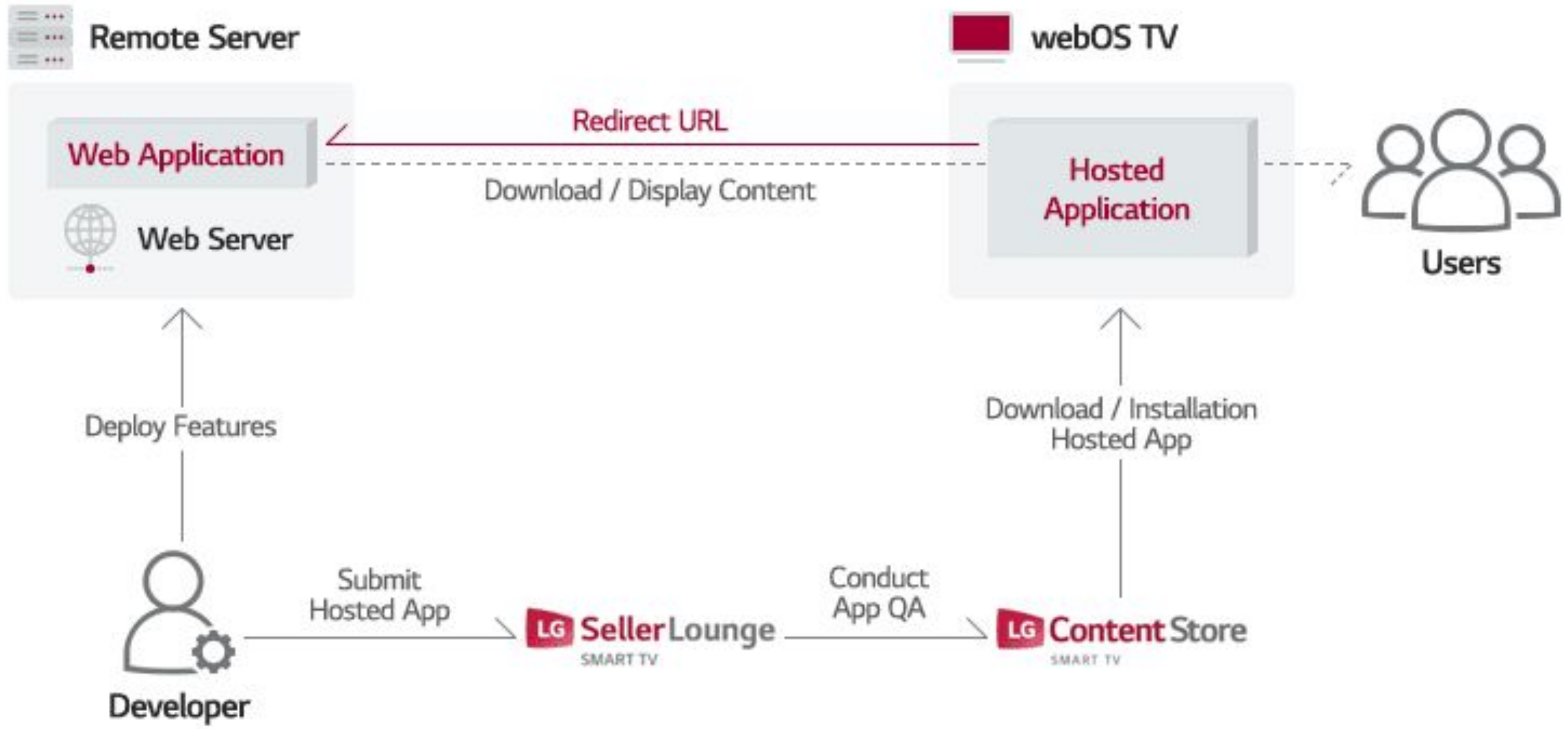
- Basic Web App(Packaged Web app) is the most basic type of an app for webOS TV

- provided as a package where all the resources are included

- when a user downloads the package and installs the app, the resources will also be installed in their device

- when there is a change to the package, code or resources, a new package needs to be submitted to the app store

Hosted Web App



Hosted Web App

- In a hosted web app, the actual content of the app is hosted on a remote web server
- when launched, the URL of the app is redirected to the web app on the web server, and the resources are downloaded from the server to the device.
- content, including new features can be updated any time without pushing update to device
- performance depends on the quality of connection to the server

Prerequisite

- We will use the following tools
    - CLI(Command Line Interface)
    - WebOS TV Simulator
    - Developer Mode App(testing on TV)

- Installation link and guide found on WebOS-TV Developer site https://webostv.developer.lge.com/

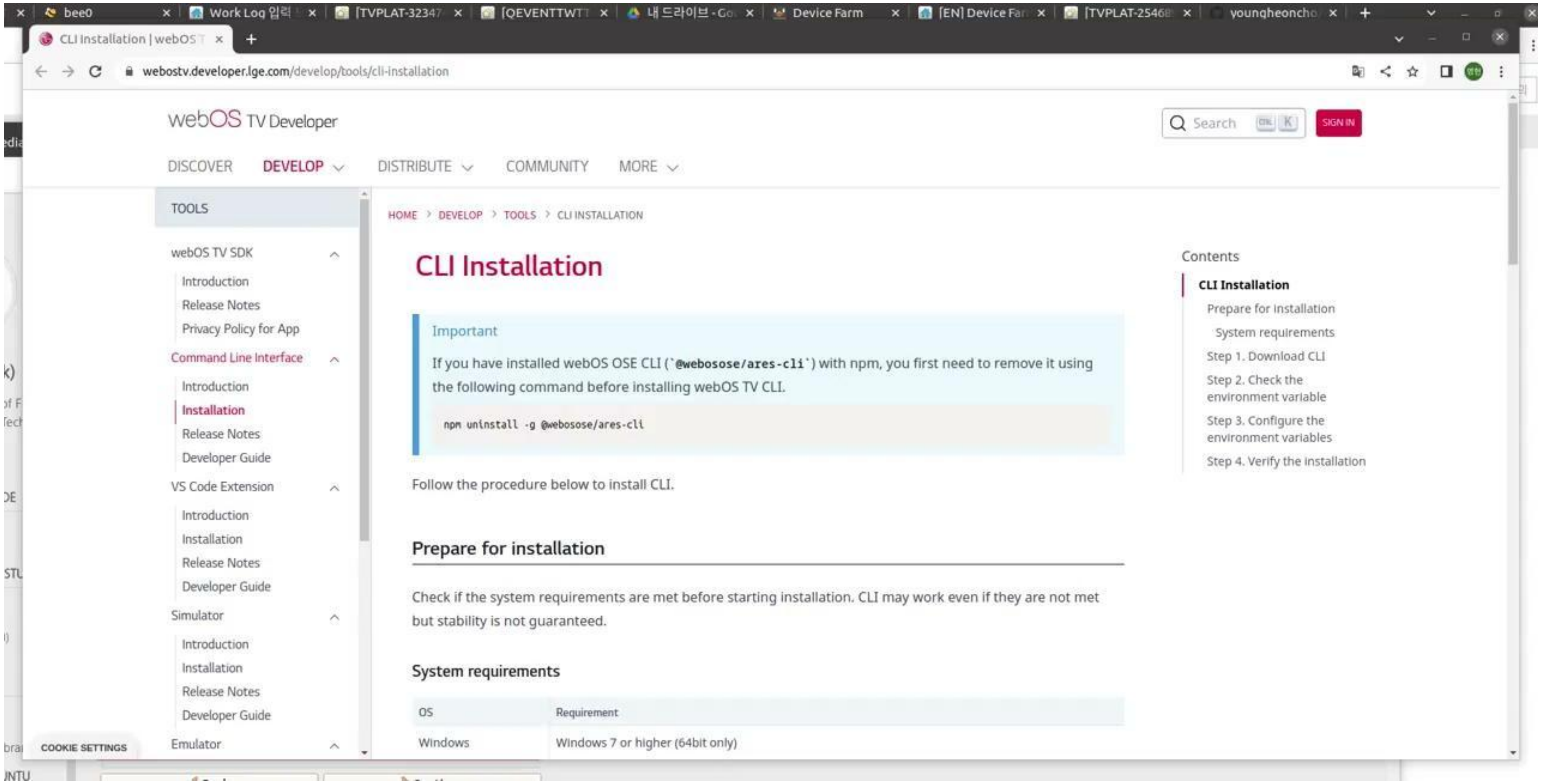
CLI(Command Line Interface)

- Interface providing collection of commands used for creating, packaging, installing, and launching web apps.
- Download from WebOS-TV developer site https://webostv.developer.lge.com/develop/tools/cli-installation

CLI(Command Line Interface)

CLI Command	Features
ares-generate	● Creating an app from templates ● Creating JS service files ● Creating an appinfo.json file
ares-package	● Packaging an app in minify or non-minify mode ● Packaging an app and js service into a pakage file (.ipk)
ares-setup-device	● Adding/modifying/removing/setting target device
ares-install	● Installing/uninstalling an app on a target device ● Listing apps that are installed on a target device
ares-launch	● Launching an app on a target device ● Launching an app on a target device with parameters ● Listing apps that are running on a target device ● Launching an app on the Simulator
ares-device-info	● Retrieving the system information of a target device ● Monitoring resource usage of a target device

CLI(Command Line Interface)



CLI Installation

Important

If you have installed webOS OSE CLI (`@webosose/ares-cli`) with npm, you first need to remove it using the following command before installing webOS TV CLI.

```
npm uninstall -g @webosose/ares-cli
```

Follow the procedure below to install CLI.

Prepare for installation

Check if the system requirements are met before starting installation. CLI may work even if they are not met but stability is not guaranteed.

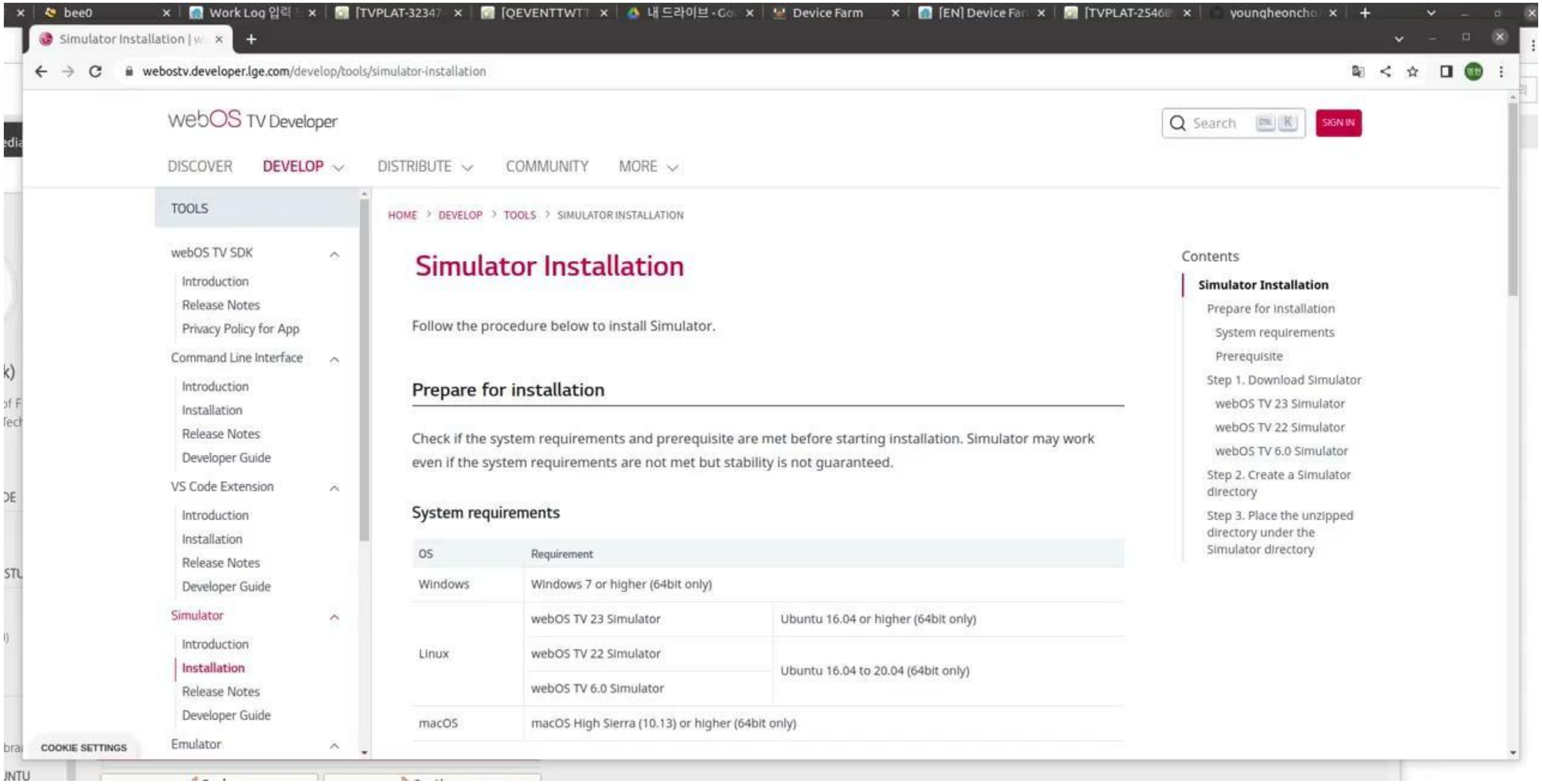
System requirements

OS	Requirement
Windows	Windows 7 or higher (64bit only)

WebOS TV Simulator

- Simulator for launching app on PC
- Download from WebOS-TV developer site https://webostv.developer.lge.com/develop/tools/simulator-installation

WebOS TV Simulator



webostv.developer.lge.com/develop/tools/simulator-installation

HOME > DEVELOP > TOOLS > SIMULATOR INSTALLATION

Simulator Installation

Follow the procedure below to install Simulator.

Prepare for installation

Check if the system requirements and prerequisite are met before starting installation. Simulator may work even if the system requirements are not met but stability is not guaranteed.

System requirements

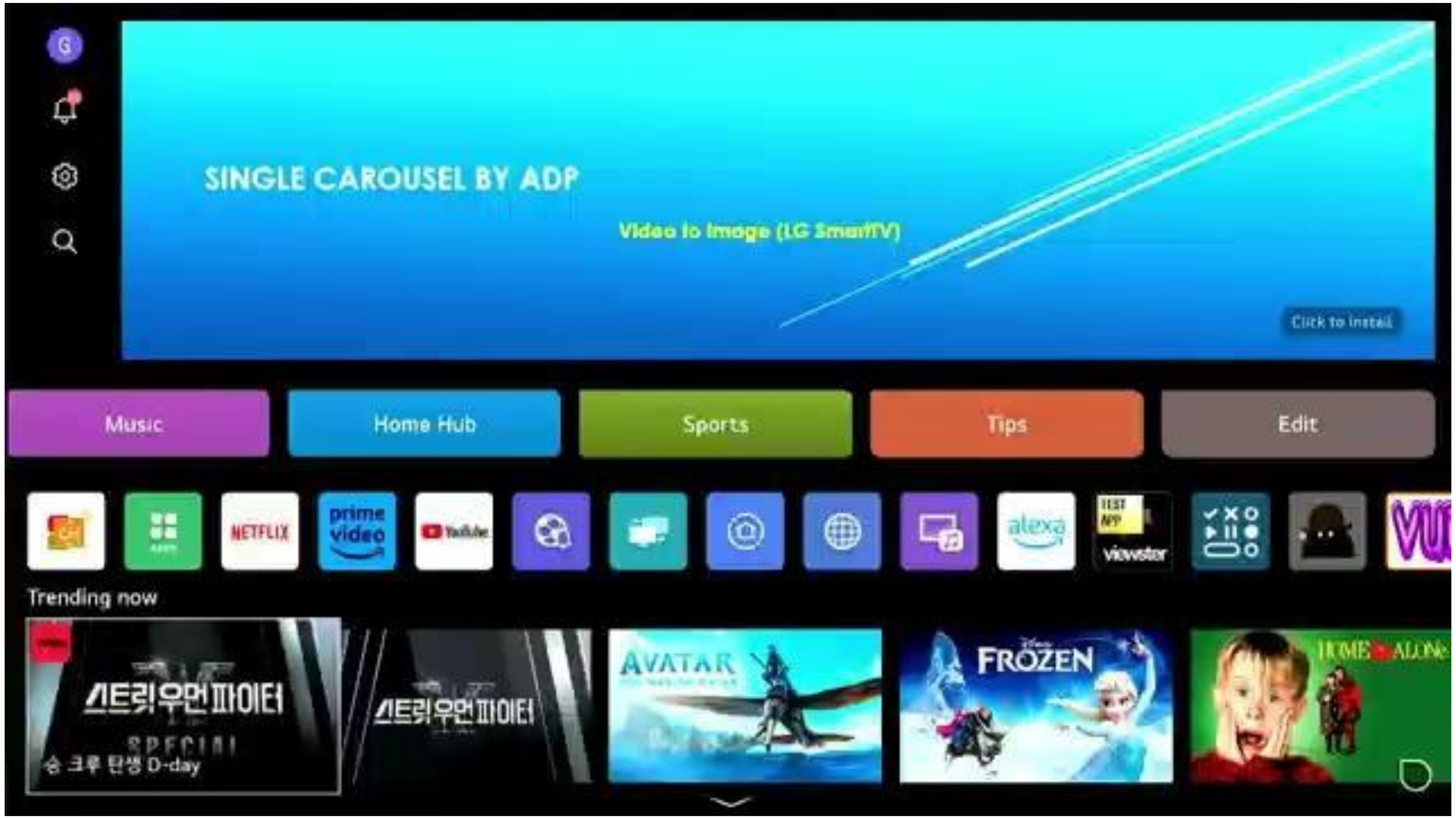
OS	Requirement	
Windows	Windows 7 or higher (64bit only)	
Linux	webOS TV 23 Simulator	Ubuntu 16.04 or higher (64bit only)
	webOS TV 22 Simulator	Ubuntu 16.04 to 20.04 (64bit only)
	webOS TV 6.0 Simulator	
macOS	macOS High Sierra (10.13) or higher (64bit only)	

Developer Mode App

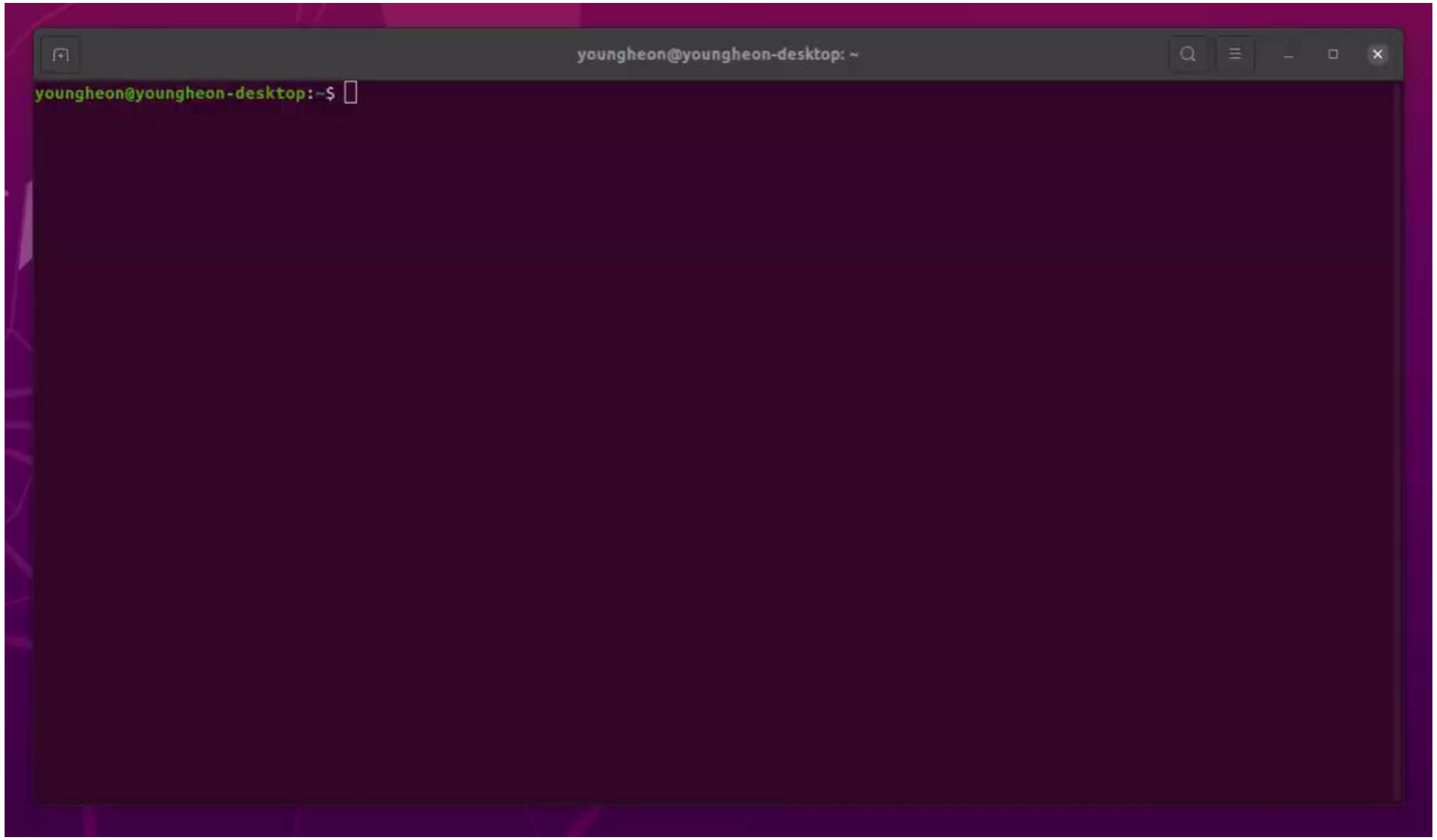
- TV app to help install your app.
- Install on your TV. Requires LG account.
- Guide for installing Developer mode app & preparing LG account is available on WebOS-TV developer site
  https://webostv.developer.lge.com/develop/getting-started/developer-mode-app

  https://webostv.developer.lge.com/develop/getting-started/preparing-lg-account
- no need for developer mode app when testing on PC with the simulator

Developer Mode App

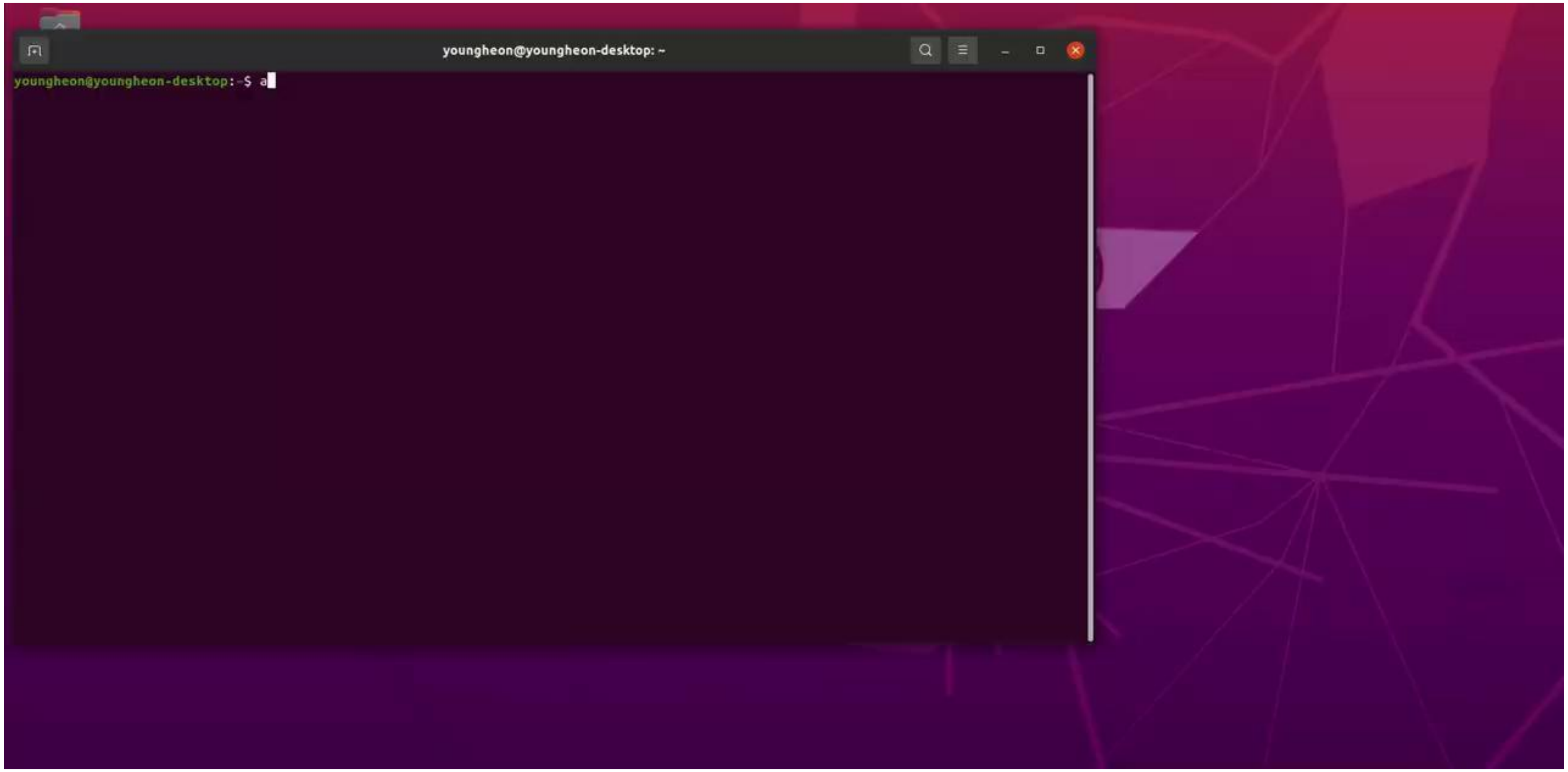


Creating a basic web-app



```
youngheon@youngheon-desktop:~$
```

Launching basic web-app on simulator



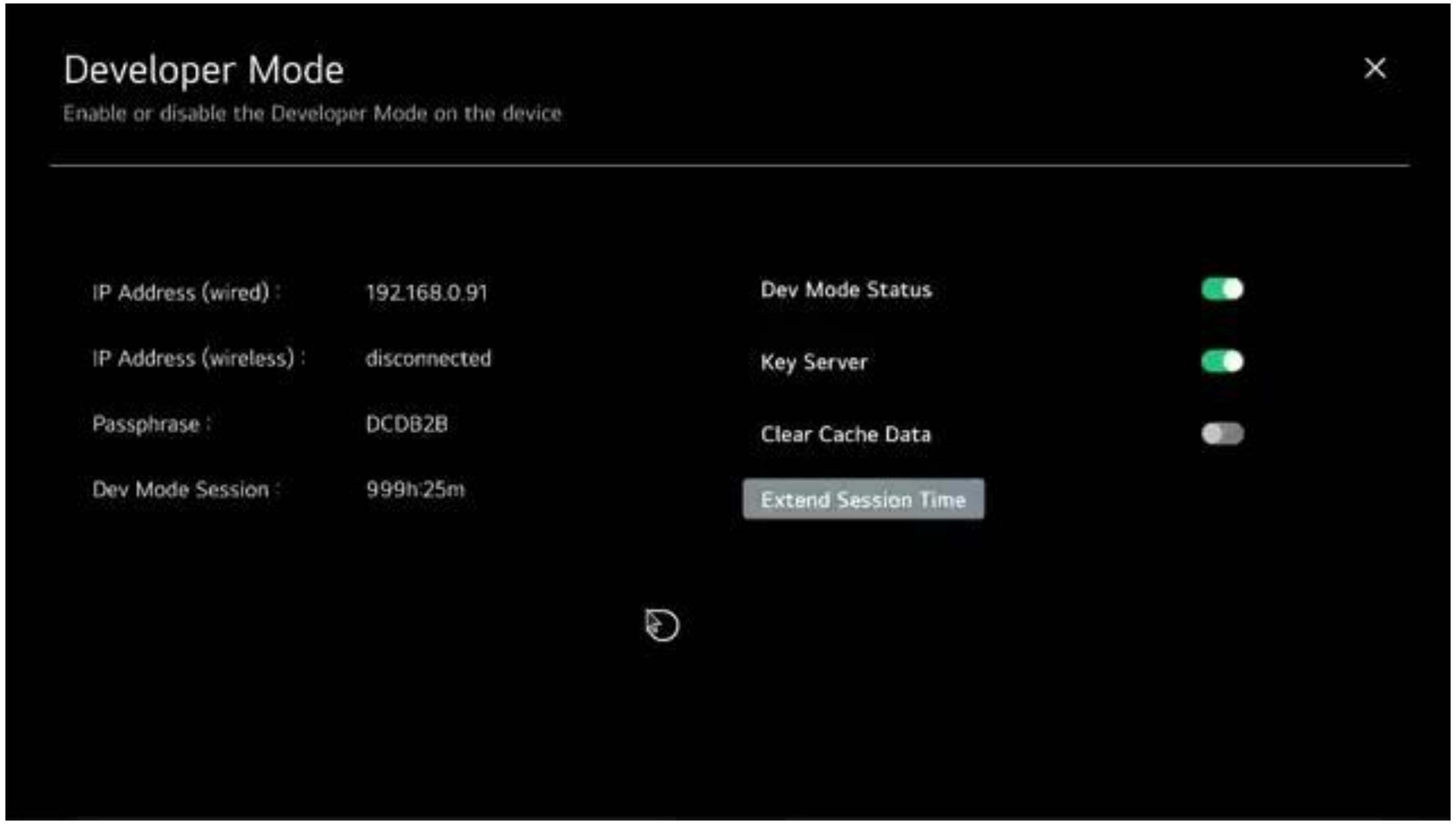
```
youngheon@youngheon-desktop:~$ a
```

Launching basic web-app on TV

Toggle Developer Mode

Launching basic web-app on TV

Connect TV to your workspace



Developer Mode

Enable or disable the Developer Mode on the device

IP Address (wired) : 192.168.0.91

IP Address (wireless) : disconnected

Passphrase : DCDB2B

Dev Mode Session : 999h:25m

Dev Mode Status

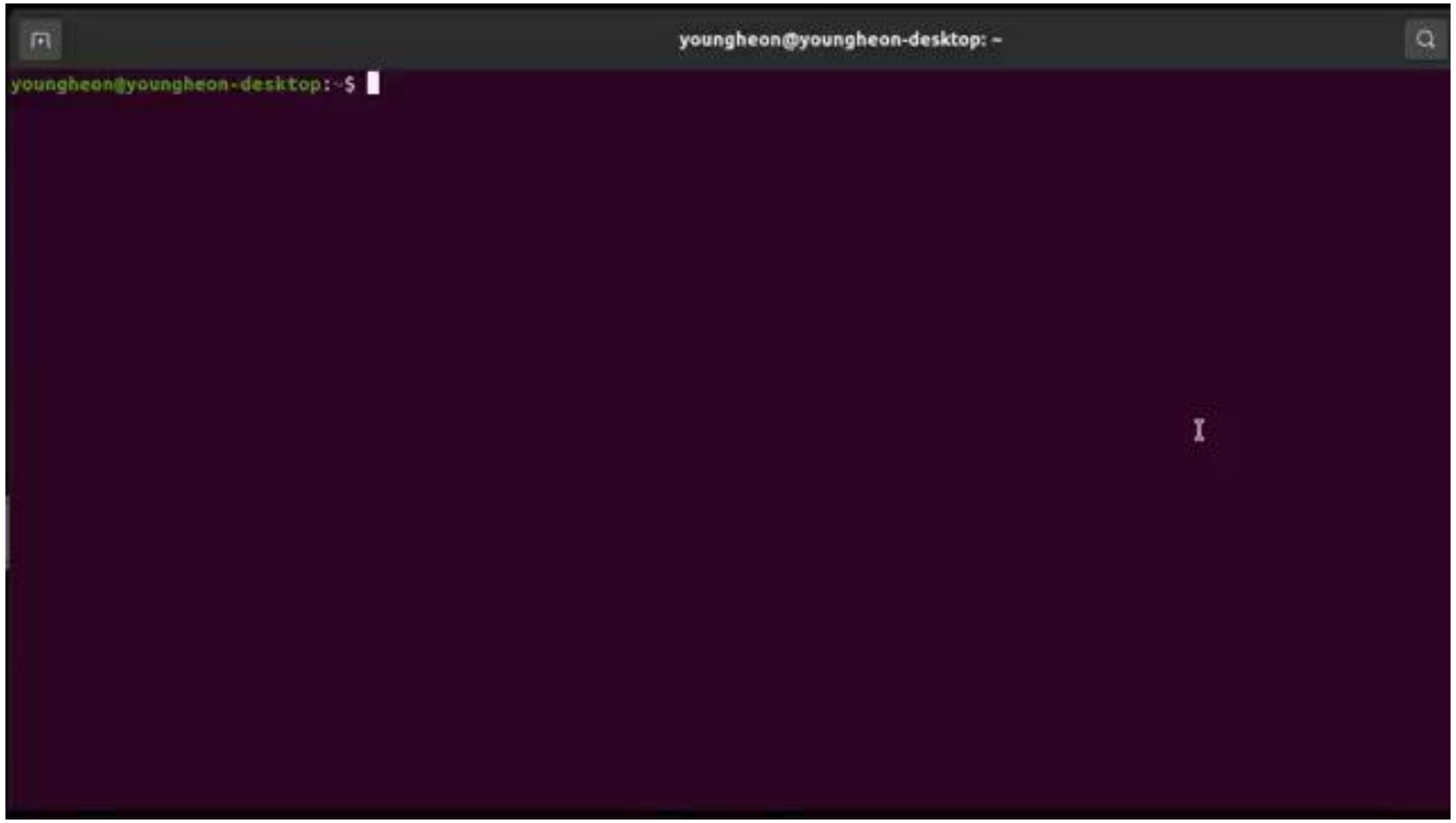
Key Server

Clear Cache Data

Extend Session Time

Launching basic web-app on TV

Install and launch app on TV.



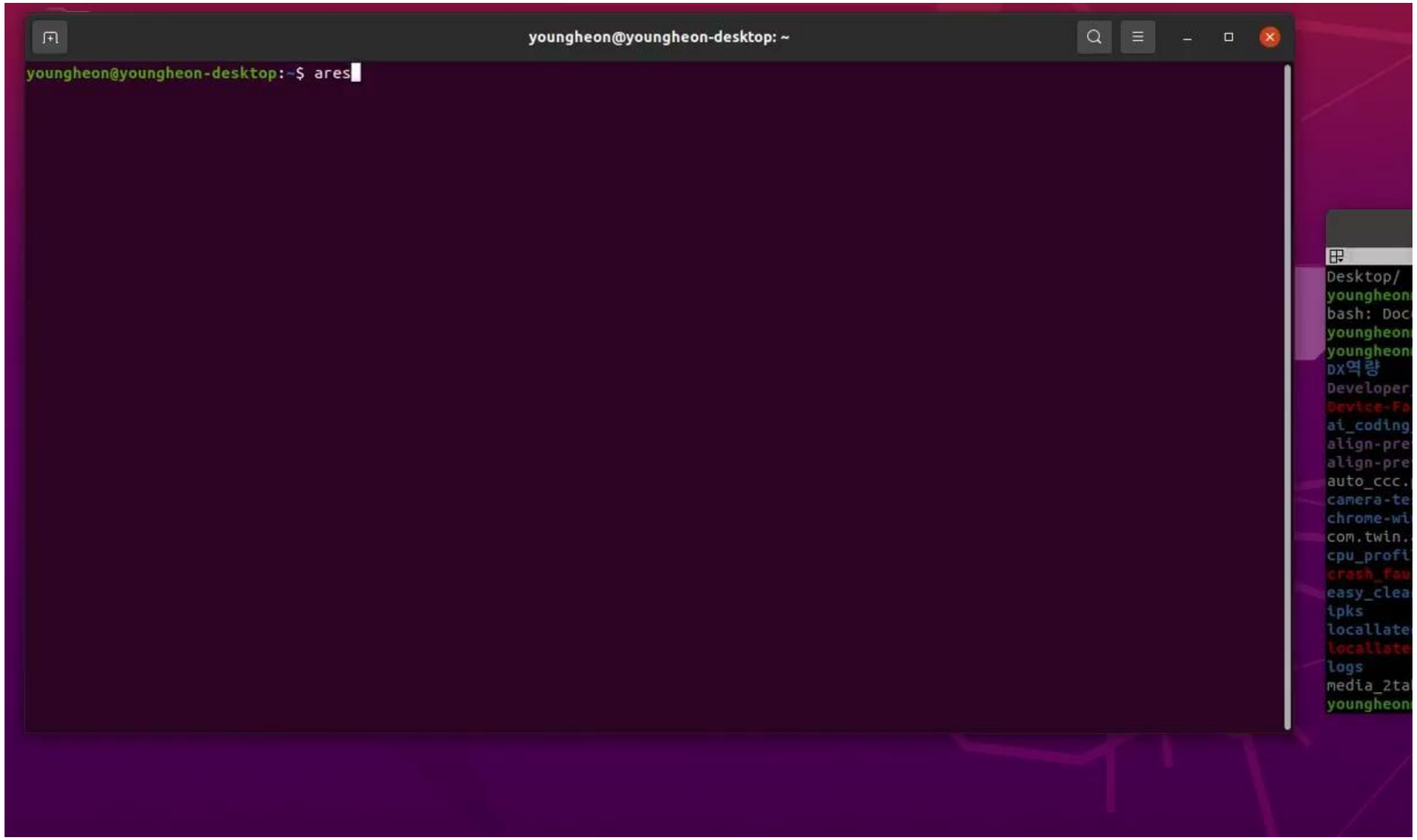
```
youngheon@youngheon-desktop:~$
```

Launching basic web-app on TV

- You can use ares-launch command to launch app on TV from PC

- ares-launch -d myTV com.domain.app

Creating a hosted web-app



youngheon@youngheon-desktop: ~

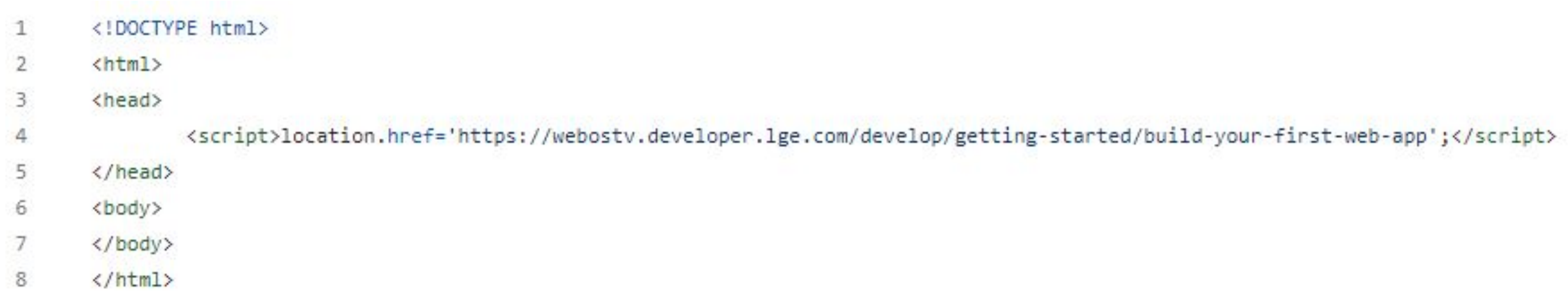
```
youngheon@youngheon-desktop:~$ ares
```

```
Desktop/
youngheon
bash: Doc
youngheon
youngheon
DX역량
Developer
Device-Fa
ai_coding
align-pre
align-pre
auto_ccc.
camera-te
chrome-wi
com.twin.
cpu_profi
crash_fau
easy_clea
ipks
locallate
locallate
logs
media_2ta
youngheon
```

Creating a hosted web-app

- The address of the remote web server is set in the index.html file created.

- Redirect to a new address by modifying the index.html file



```
<!DOCTYPE html>
<html>
<head>
        <script>location.href='https://webostv.developer.lge.com/develop/getting-started/build-your-first-web-app';</script>
</head>
<body>
</body>
</html>
```

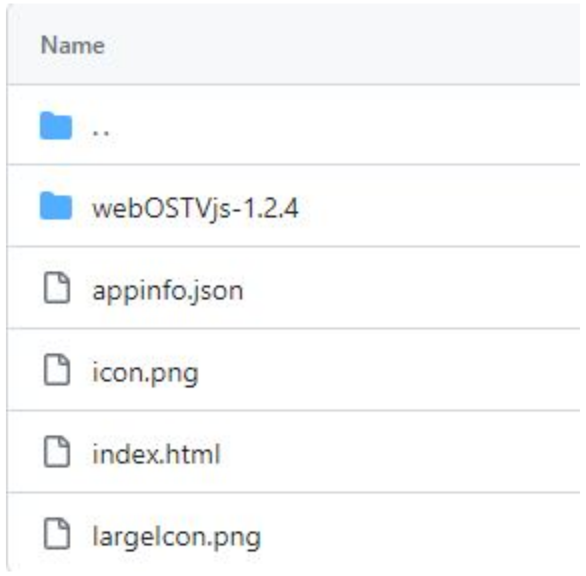
2. App-Basics

App Templates

- To help accelerate app development, the WebOS TV platform provides templates for basic and hosted web app.

- The WebOS-TV apps we created using CLI follows this template.

App Templates - Basic web app



Name

..

webOSTVjs-1.2.4

appinfo.json

icon.png

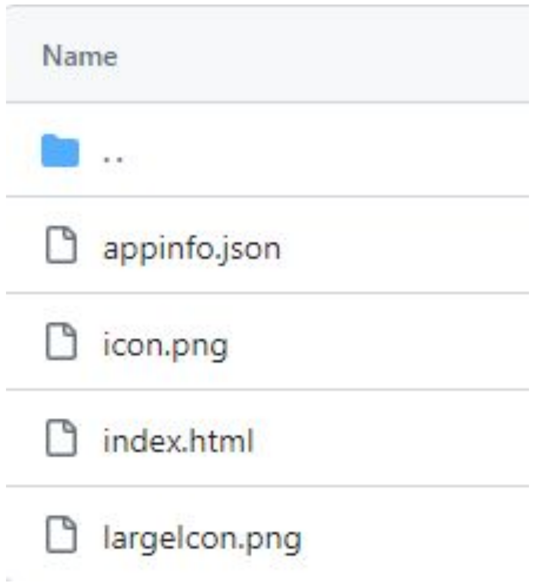
index.html

largeIcon.png

App Templates - Basic web app

Directory/File	Description
WebOSTVjs-x.x.x	The WebOS TV library directory. The WebOSTV.js is a portable library to access TV-specific features and functionality for WebOS TV. The WebOS TV-specific library uses Luna Service API which consists of essential services and features for WebOS TV.
appinfo.json	The web app configuration file. The appinfo.json file includes metadata of the web app. This file must exist for packaging the web app.
icon.png	The icon image file. This image is displayed on system notifications
largeicon.png	The large icon image file. This image is displayed on the top left corner of the screen, when the user hovers over an app tile on the Launcher
index.html	The web app's main page.

App Templates - Hosted web app



Name

..

appinfo.json

icon.png

index.html

largeIcon.png

App Templates - Hosted web app

Directory/File	Description
appinfo.json	The web app configuration file. The appinfo.json file includes metadata of the web app. This file must exist for packaging the web app.
icon.png	The icon image file. This image is displayed on system notifications
largeicon.png	The large icon image file. This image is displayed on the top left corner of the screen, when the user hovers over an app tile on the Launcher
index.html	The web app's main page that contains the redirect URL.

Web App Info

- appinfo.json for the WebOS-TV apps created with CLI follow the web app info template.
- The web app info template provides an appinfo.json file with sample values.
- The appinfo.json file holds the app's name, ID, app type, icon image, and the main page information to the file.

Web App Info

```
{
    "id": "com.domain.app",
    "version": "0.0.1",
    "vendor": "My Company",
    "type": "web",
    "main": "index.html",
    "title": "new app",
    "icon": "icon.png",
    "largeIcon": "largeIcon.png"
}
```

Web App Info - mandatory property

property	Description
id	String type. Specify your app ID.
title	Specify the app title to be shown on the Launcher and the app window.
type	Specify your app type. Only web is allowed currently.
main	Specify the launch point for your app. The file path must be relative to the appinfo.json file and needs to point to an HTML file.
icon	Specify the small icon file of your app, 80x80 pixels in PNG format.The file path must be relative to the appinfo.json file.
version	Specify your app version number, comprised of three non-negative integers separated by period.

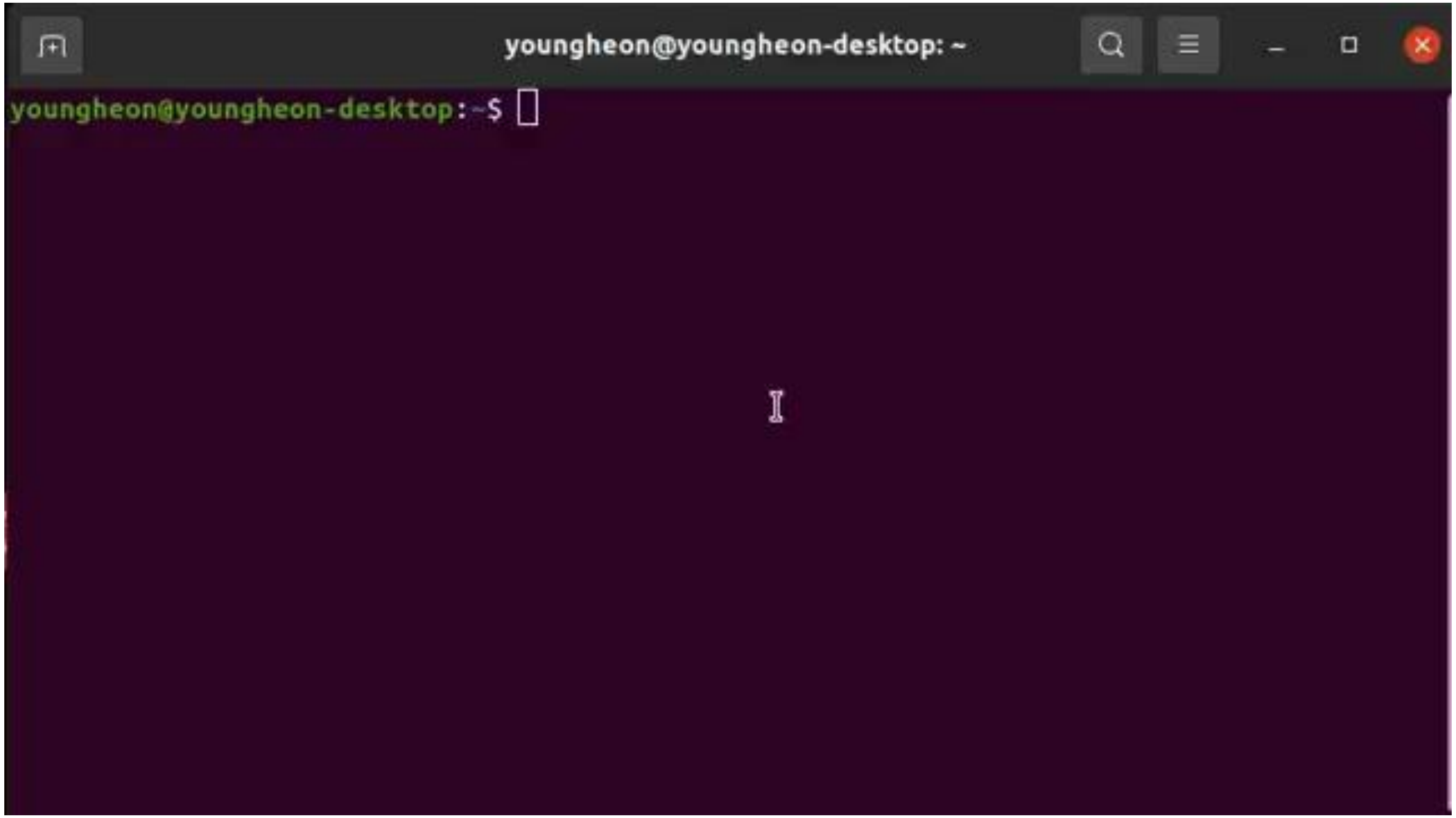
Web App Info - optional property

property	Description
vendor	Specify your app owner to be used in the launcher and device info dialogs.
largeIcon	Specify your large icon file of your app, 130x130 pixels in PNG format. The file path must be relative to the appinfo.json file.

Debugging app

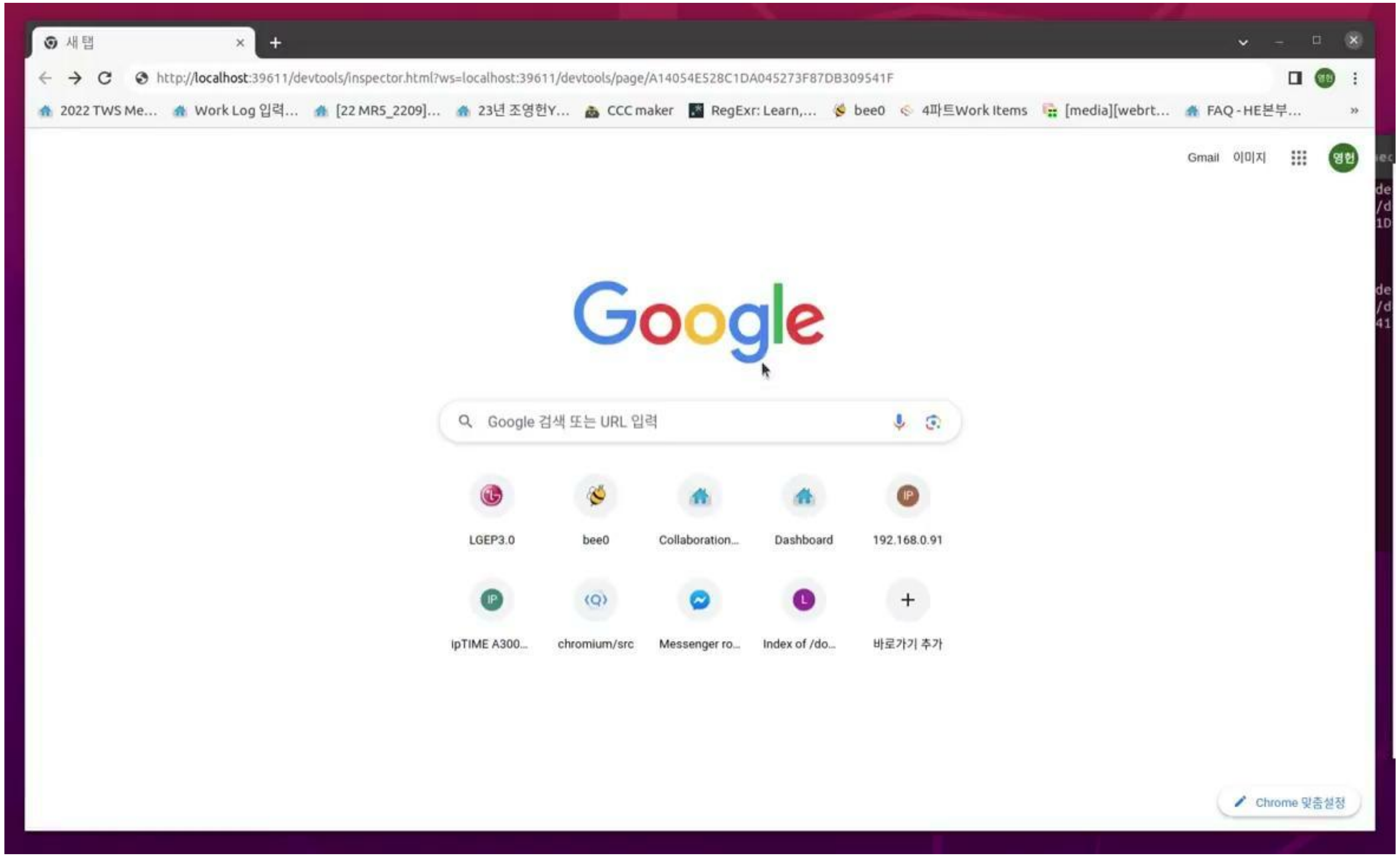
- The webOS TV platform supports Web Inspector for debugging web apps.
- Web Inspector monitors your app running on a target device with the execution information.
- Web Inspector uses the Chromium browser as a default browser.
- use the ares-inspect command
  ```
  ares-inspect --device tv1 com.domain.app
  ```

Debugging app running on TV

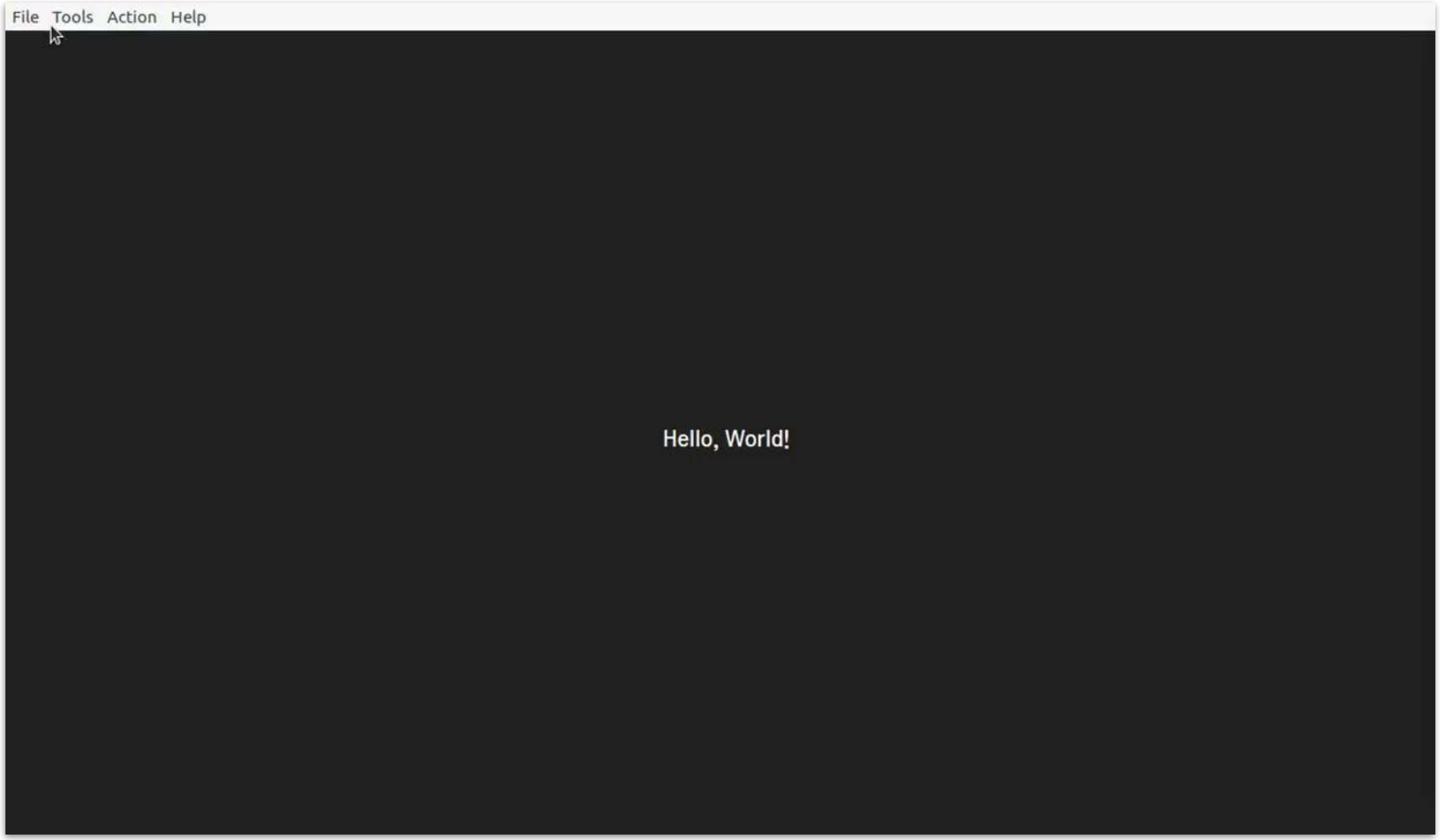


```
youngheon@youngheon-desktop:~$ 
```

Debugging app running on TV



Debugging with simulator



Hello, World!

3. JavaScript Service

Prerequisite

- JS-Service is created using Node.js
- Download and install NodeJS from the official link. http://www.nodejs.org/
- npm(node package manager) will also be installed
- To check if node and npm is successfully installed in your environment, use the following command
  node –v // check node version
  npm –v // check npm version

Prerequisite - supported Node.js version

WebOS TV platform version	Node.js version
WebOS TV 23	v12.22.2
WebOS TV 22	v12.21.0
WebOS TV 6.0	v12.14.1
WebOS TV 5.0	v8.12.0
WebOS TV 4.x	v0.12.2
WebOS TV 3.x	v0.12.2

Prerequisite

- The n package which can be installed from npm provides commands for maintaining the node/npm version installed on your PC.

- You could use the n package to easily update your node/npm if needed.
  https://www.npmjs.com/package/n

What is JS-Service

- JavaScript services on WebOS provide the way for apps to do work, even when the app isn't running.
- They also provide access to platform features that aren't usually available to web apps.

What is JS-Service

JS service on WebOS TV has the following characteristics:

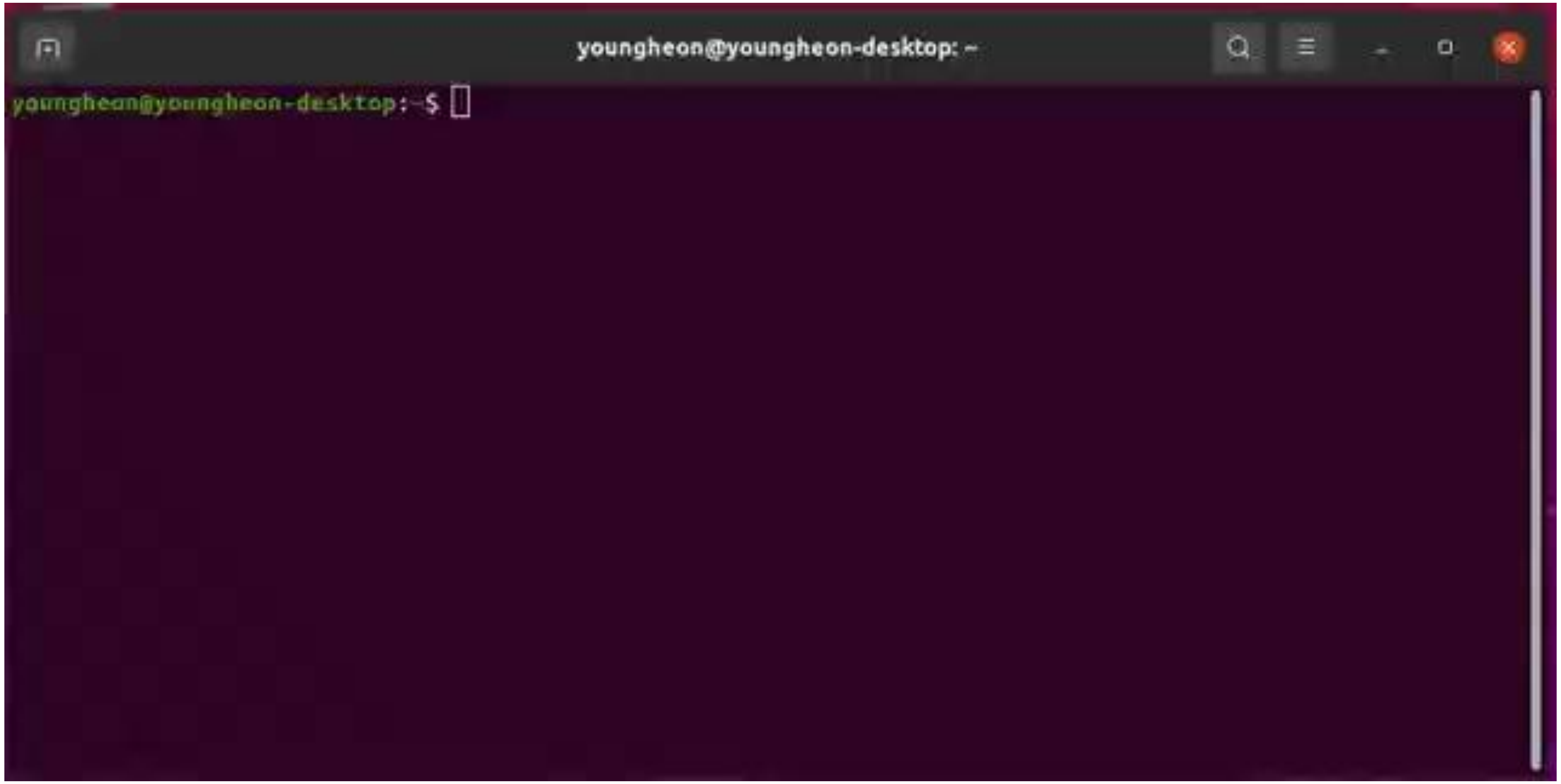
- It is written in JavaScript and created using Node.js.
- It runs in the background on WebOS TV.
- It provides additional access to platform features such as low-level networking, file system access, and binary data processing.
- It performs tasks for one or more apps.

What is JS-Service

- use JS-service to perform tasks that a WebOS app can't do or to do work by service in the background.

- example of JS service implementations:
    - Downloading attachments in the background for an email reader
    - Uploading images to a picture-sharing website from an app
    - Performing a long-running computation or file operation

Creating a JS-Service



```
youngheon@youngheon-desktop:~$ 
```

JS-Service Template

The WebOS TV platform provides templates for JS service.

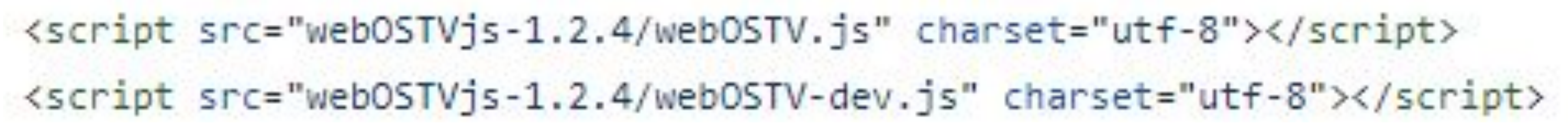
Directory/File	Description
helloworld_service.js	The sample JS service code that provides several simple commands. These commands are specified in the services.json file to use them.
package.json	The configuration file of NPM.
services.json	The configuration file defines what commands the service provides on the WebOS bus.

Calling JS-Service

- You can call webOS services using the webOSTV.js library on the webOS TV platform.

- The webOSTV.js library is basically included in the basic template of the CLI.

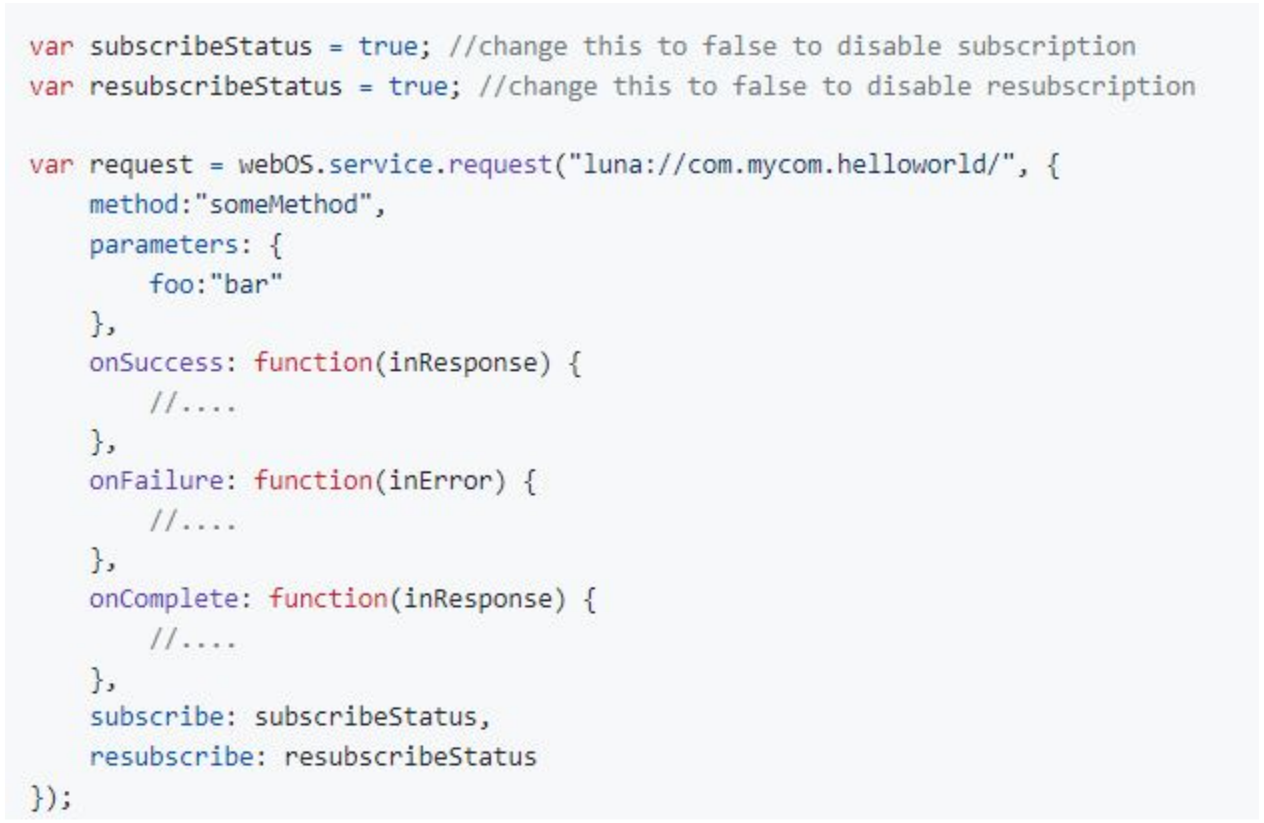
- In our basic web app created with CLI, you can see that the webOSTV.js library is imported in the index.html file.



```
<script src="webOSTVjs-1.2.4/webOSTV.js" charset="utf-8"></script>
<script src="webOSTVjs-1.2.4/webOSTV-dev.js" charset="utf-8"></script>
```

Calling JS-Service - from web app

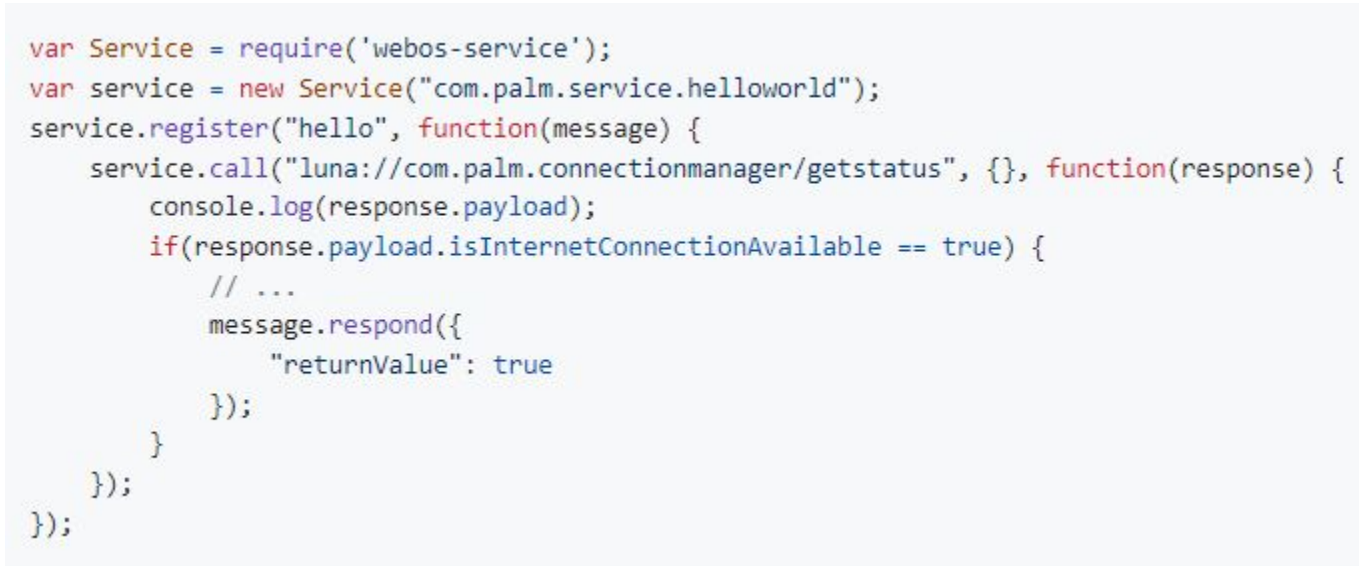
- Any application can include webOSTV.js and make webOS service calls using the webOS.service.request method.



```
var subscribeStatus = true; //change this to false to disable subscription
var resubscribeStatus = true; //change this to false to disable resubscription

var request = webOS.service.request("luna://com.mycom.helloworld/", {
    method:"someMethod",
    parameters: {
        foo:"bar"
    },
    onSuccess: function(inResponse) {
        //....
    },
    onFailure: function(inError) {
        //....
    },
    onComplete: function(inResponse) {
        //....
    },
    subscribe: subscribeStatus,
    resubscribe: resubscribeStatus
});
```

Calling JS-Service - from another JS service

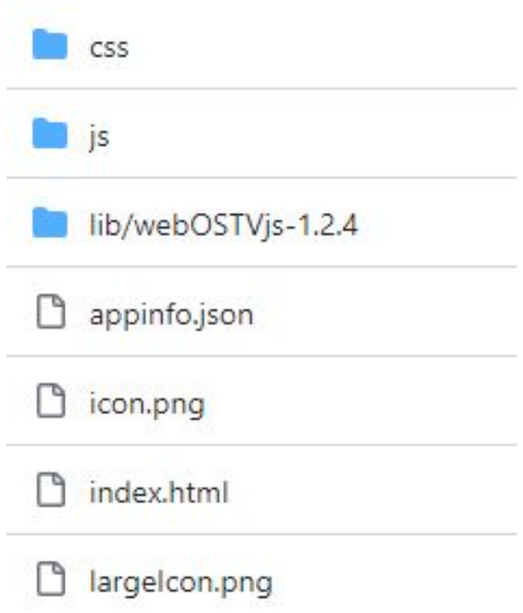
- Any service can include webOSTV.js and make webOS service calls using the service.call method.



```
var Service = require('webos-service');
var service = new Service("com.palm.service.helloworld");
service.register("hello", function(message) {
    service.call("luna://com.palm.connectionmanager/getstatus", {}, function(response) {
        console.log(response.payload);
        if(response.payload.isInternetConnectionAvailable == true) {
            // ...
            message.respond({
                "returnValue": true
            });
        }
    });
});
```

HelloWorldService

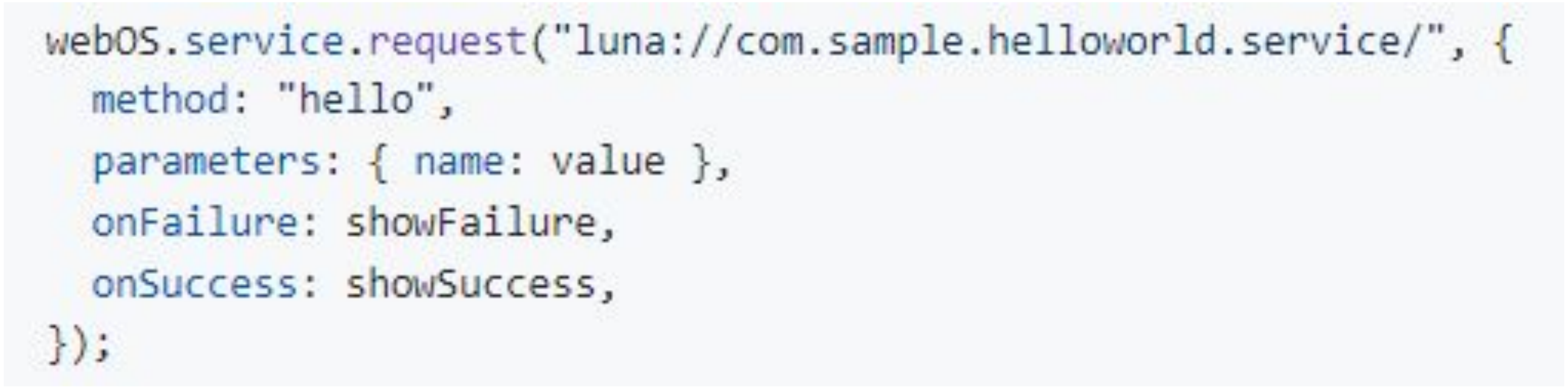
- The sample code consists of an app & service
- added css & js file(script.js) to the app



css
js
lib/webOSTVjs-1.2.4
appinfo.json
icon.png
index.html
largeIcon.png

HelloWorldService

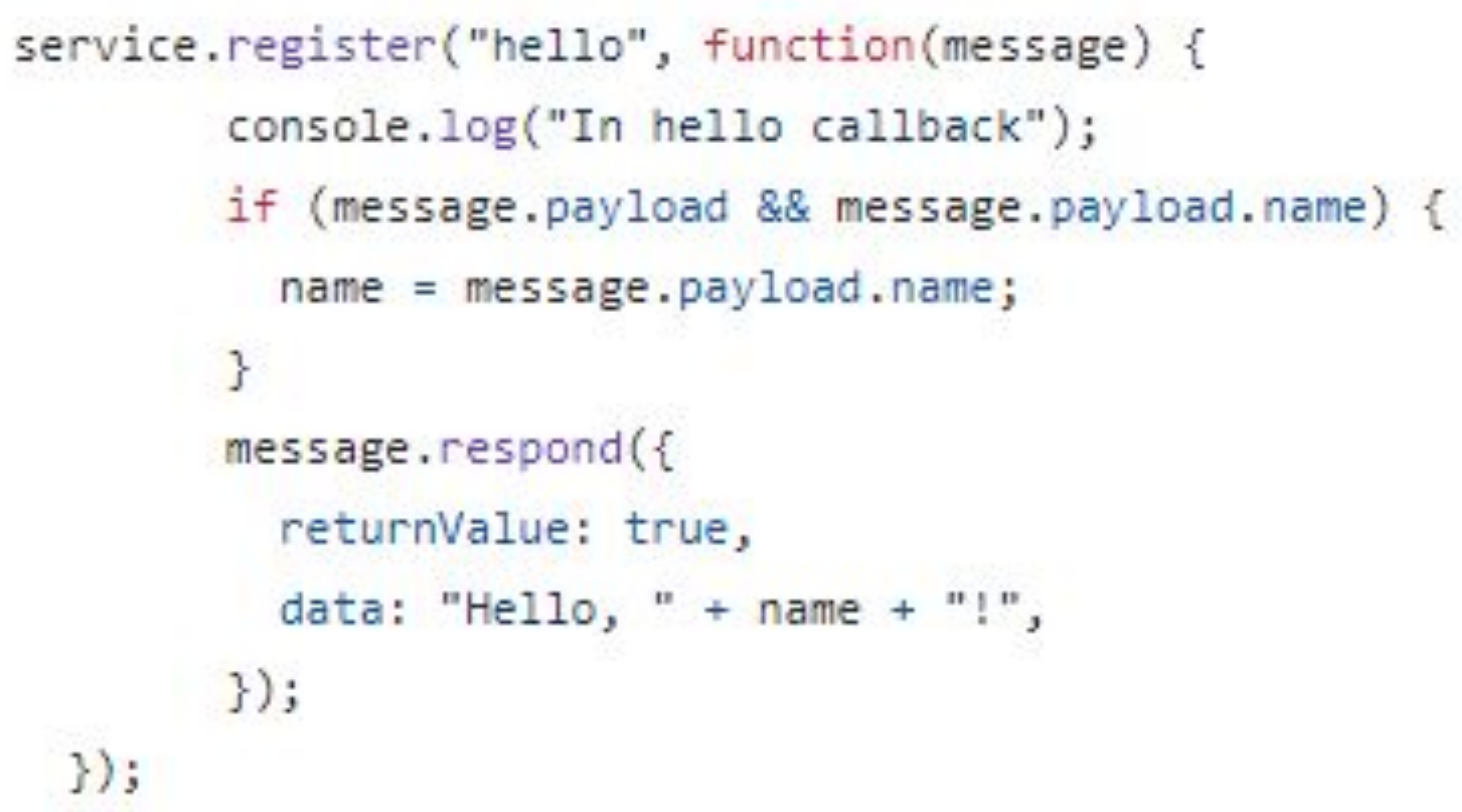
app/js/script.js



```
webOS.service.request("luna://com.sample.helloworld.service/", {
  method: "hello",
  parameters: { name: value },
  onFailure: showFailure,
  onSuccess: showSuccess,
});
```

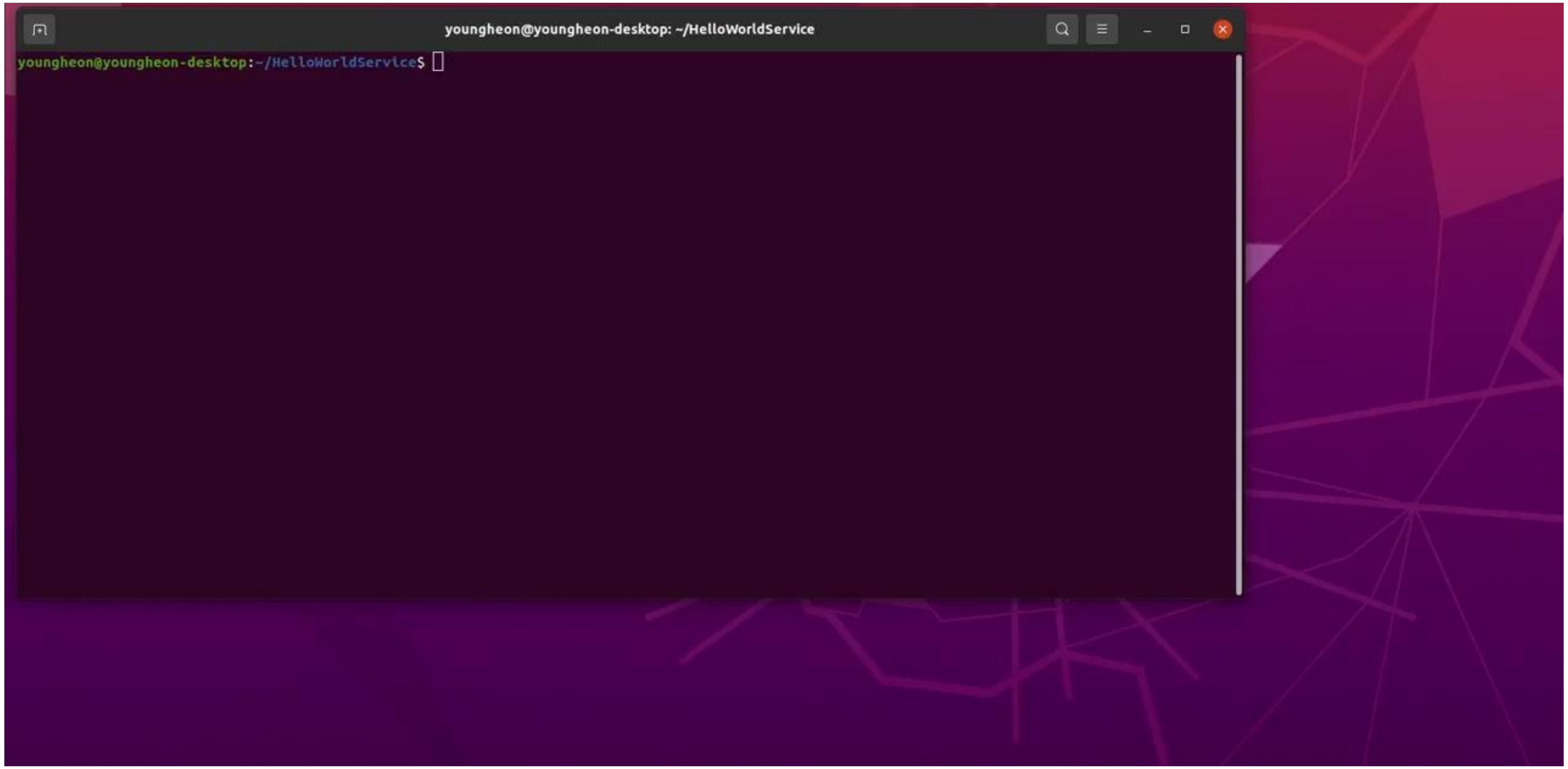
HelloWorldService

service/helloworld_service.js



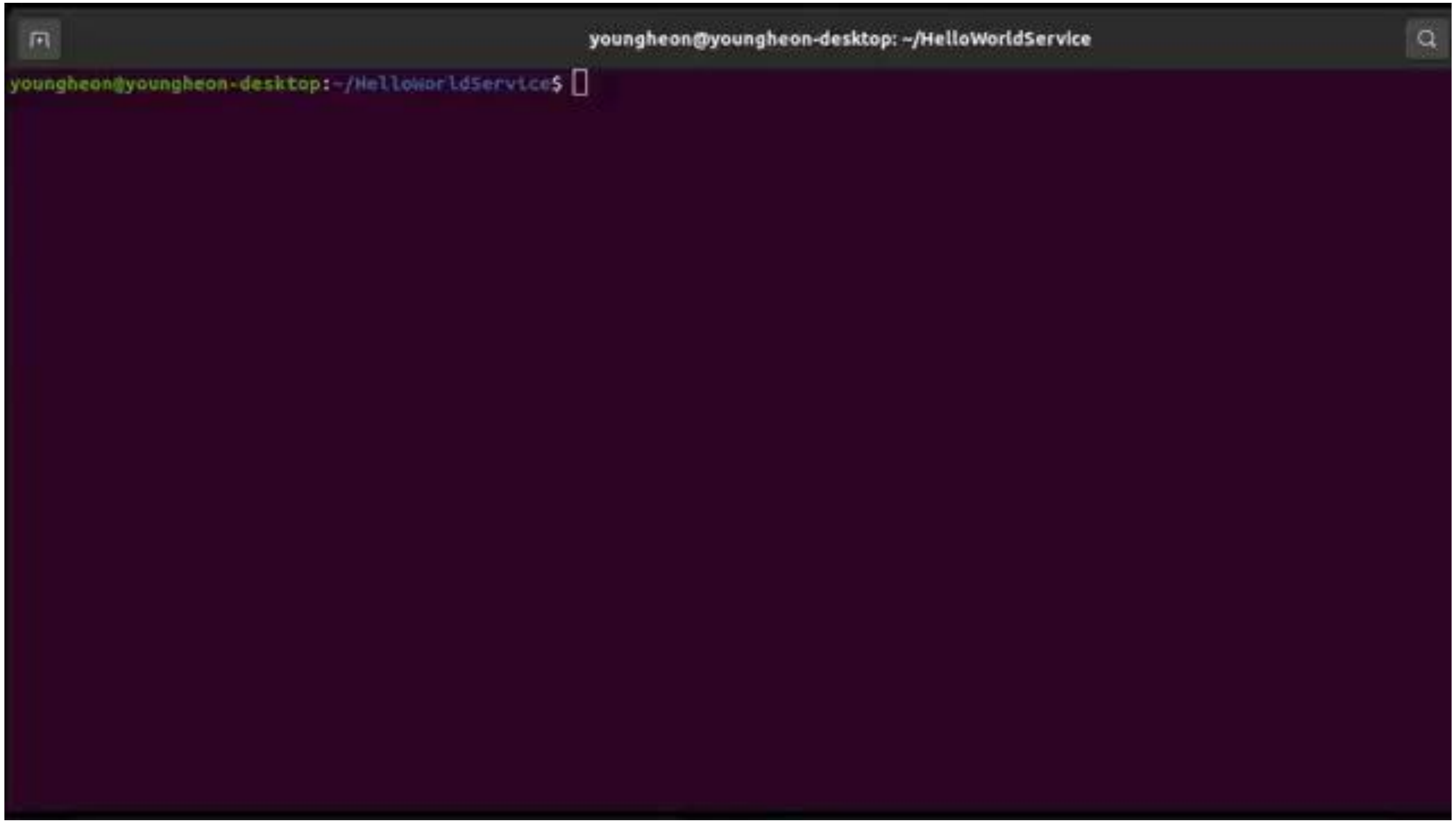
```
service.register("hello", function(message) {
        console.log("In hello callback");
        if (message.payload && message.payload.name) {
          name = message.payload.name;
        }
        message.respond({
          returnValue: true,
          data: "Hello, " + name + "!",
        });
  });
```

HelloWorldService - running on simulator



```
youngheon@youngheon-desktop:~/HelloWorldService$ 
```

HelloWorldService - running on TV



```
youngheon@youngheon-desktop:~/HelloWorldService$ 
```

Luna Service

- There are also predefined services called Luna Service.
- WebOS TV provides the Luna Service that consists of essential services and features for WebOS TV.
- Each service and feature provide application programming interface (API) and its methods.
- Note that Some APIs are partially or not supported on the WebOS TV simulator.

Luna Service - supported on simulator

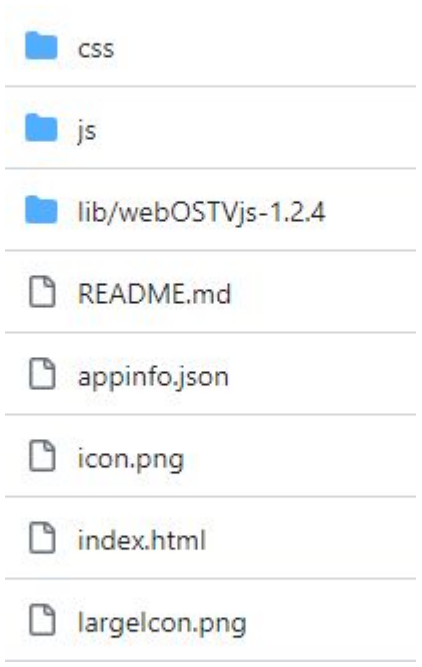
API	Description
Activity Manager	Monitors various parts of the system, and launches services when the corresponding events happen. Activity Manager can also be used to schedule work periodically, or at particular times.
Application Manager	The Application Manager provides the launch method to launch a specified application. You can launch an app directly by using the launch method with the specified app ID and appropriate arguments.
Connection Manager	Provides the status of available Internet connections. Connection manager provides methods for managing Internet connections. It enables apps to get the status of connections.
Database	Enables apps to store persistent data.
Device Unique ID	Provides app security and authentications services such as app signature verification.

Luna Service - supported on simulator

API	Description
Media Database	Enables apps to store large media-related datasets persistently.
Magic Remote	Provides methods related to the magic remote sensor. Only getSensorData method is supported on the simulator.
Settings Service	Provides a method for retrieving system setting value.
System Service	Provides information about the system time. Apps can subscribe to this method to get system time updates.
TV Device Information	Provides a method for retrieving TV system information. This API is used to check the version of webOS TV and its features.

LunaService app

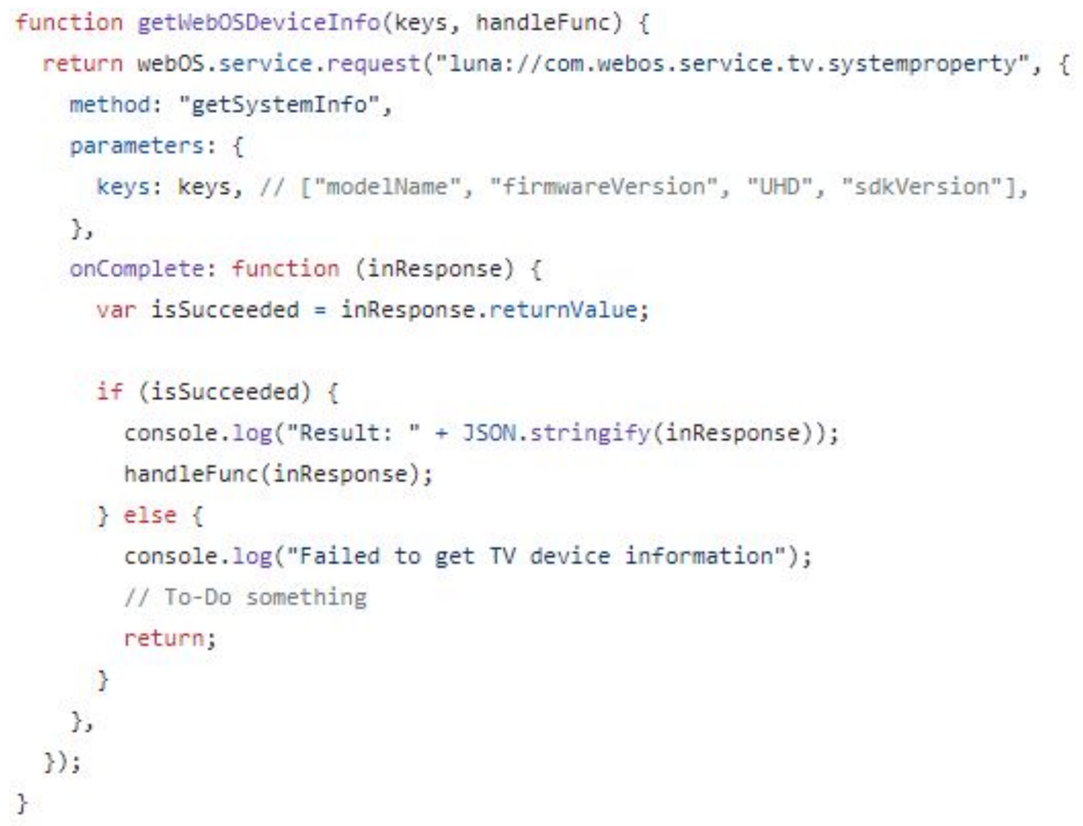
- This app uses the following luna-service API
    - TV Device Information
    - Settings Service
    - System Service



css
js
lib/webOSTVjs-1.2.4
README.md
appinfo.json
icon.png
index.html
largeIcon.png

LunaService app - TV Device Information

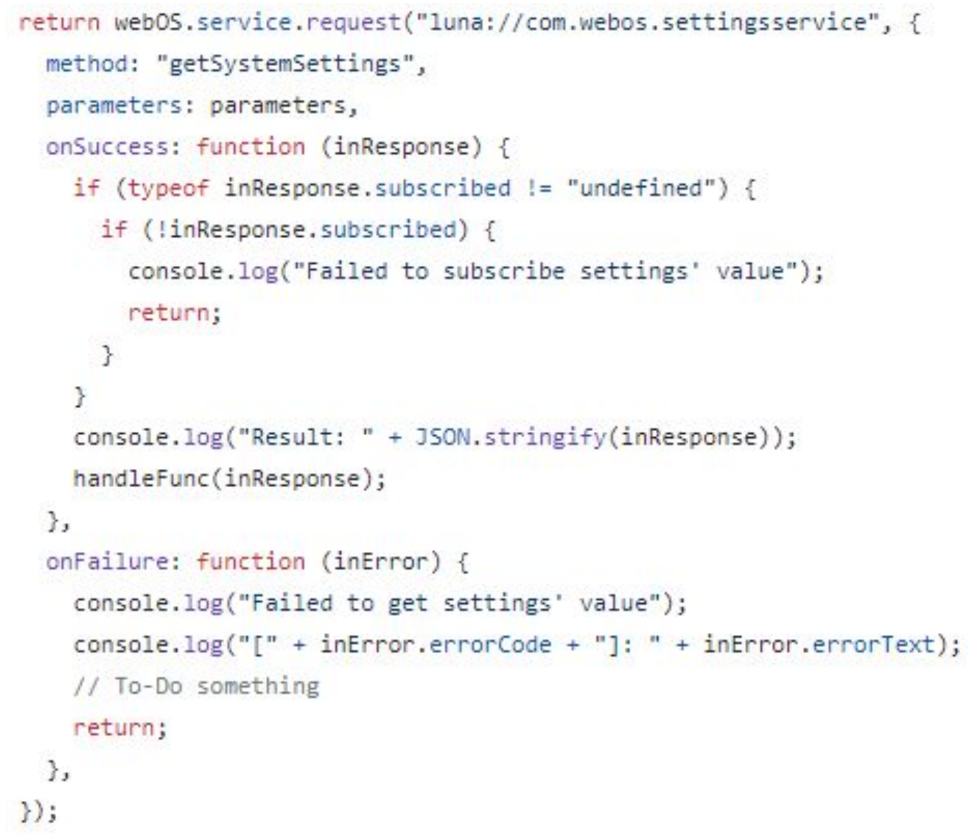
LunaService/js/device_info.js



```
function getWebOSDeviceInfo(keys, handleFunc) {
  return webOS.service.request("luna://com.webos.service.tv.systemproperty", {
    method: "getSystemInfo",
    parameters: {
      keys: keys, // ["modelName", "firmwareVersion", "UHD", "sdkVersion"],
    },
    onComplete: function (inResponse) {
      var isSucceeded = inResponse.returnValue;

      if (isSucceeded) {
        console.log("Result: " + JSON.stringify(inResponse));
        handleFunc(inResponse);
      } else {
        console.log("Failed to get TV device information");
        // To-Do something
        return;
      }
    },
  });
}
```

LunaService app - Settings Service

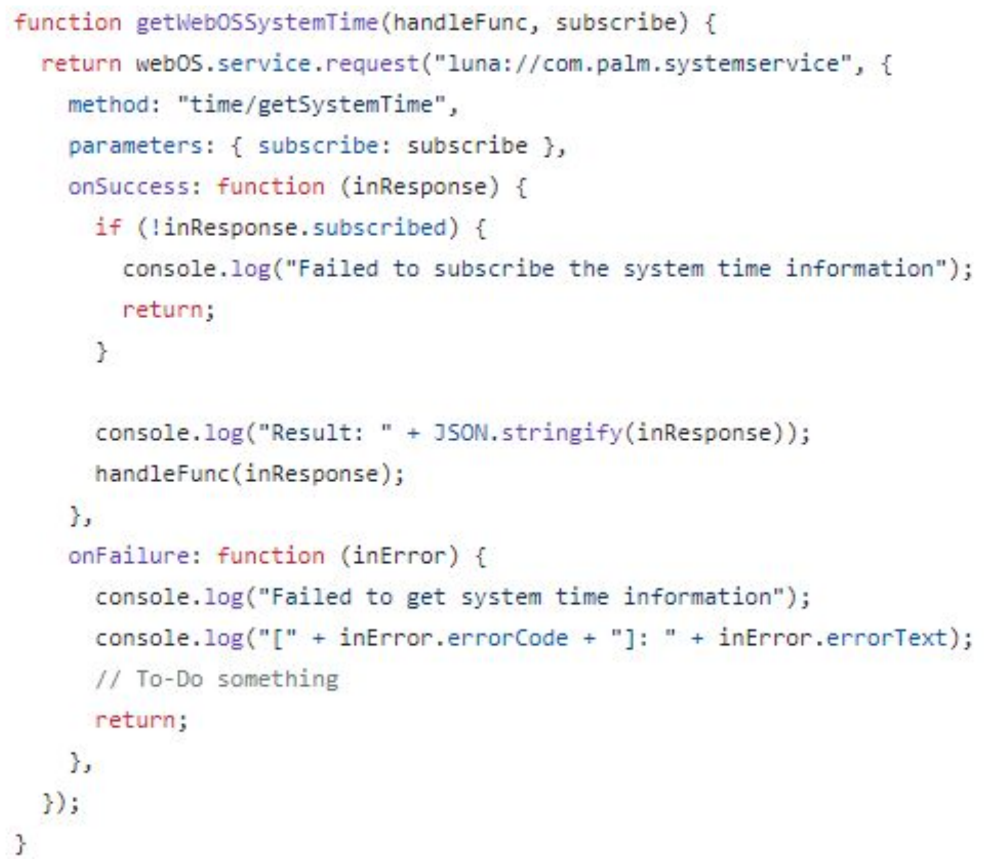
LunaService/js/settings.js



```
return webOS.service.request("luna://com.webos.settingsservice", {
  method: "getSystemSettings",
  parameters: parameters,
  onSuccess: function (inResponse) {
    if (typeof inResponse.subscribed != "undefined") {
      if (!inResponse.subscribed) {
        console.log("Failed to subscribe settings' value");
        return;
      }
    }
    console.log("Result: " + JSON.stringify(inResponse));
    handleFunc(inResponse);
  },
  onFailure: function (inError) {
    console.log("Failed to get settings' value");
    console.log("[" + inError.errorCode + "]: " + inError.errorText);
    // To-Do something
    return;
  },
});
```

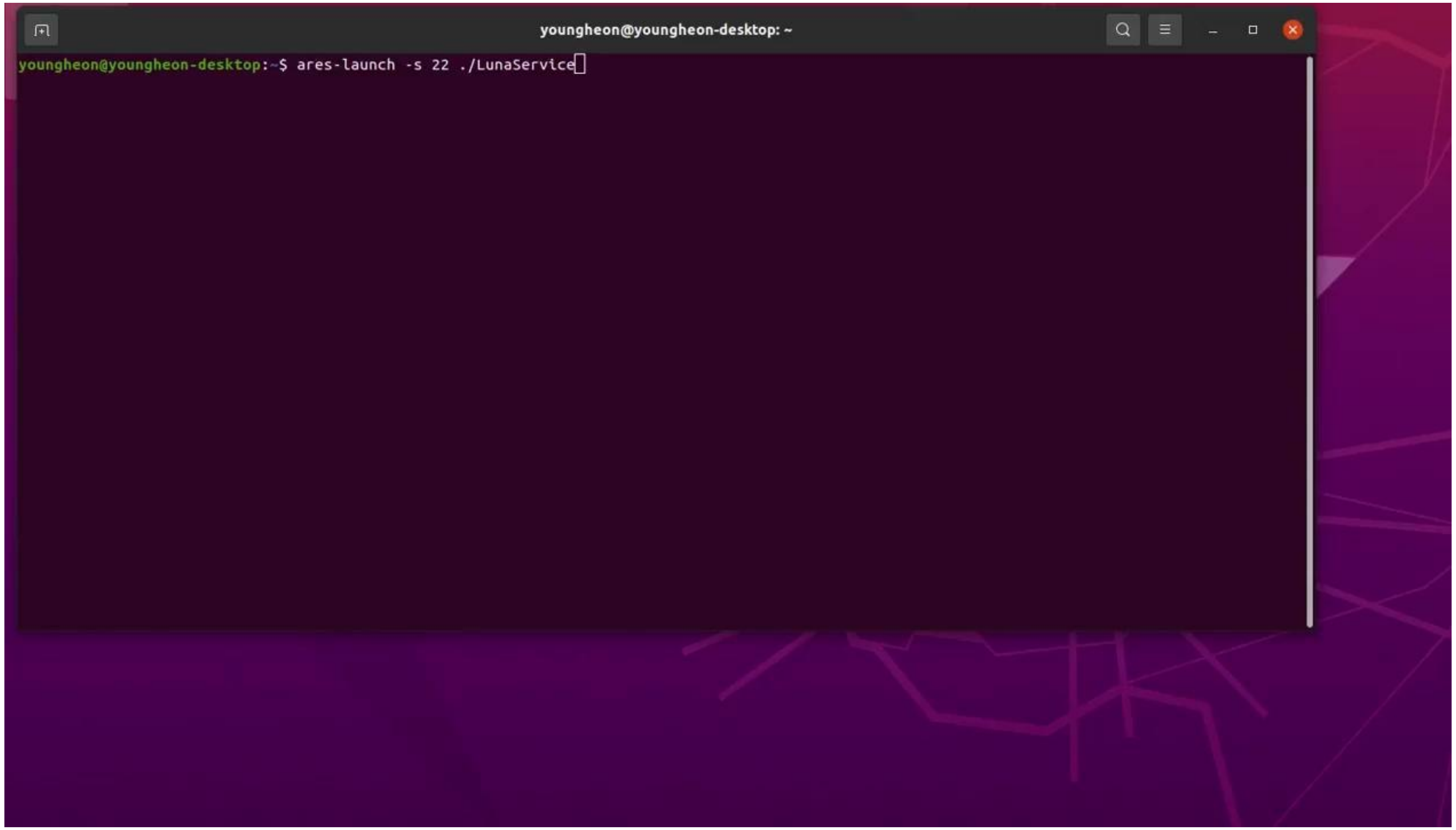
LunaService app - System Service

LunaService/js/system_time.js



```
function getWebOSSystemTime(handleFunc, subscribe) {
  return webOS.service.request("luna://com.palm.systemservice", {
    method: "time/getSystemTime",
    parameters: { subscribe: subscribe },
    onSuccess: function (inResponse) {
      if (!inResponse.subscribed) {
        console.log("Failed to subscribe the system time information");
        return;
      }

      console.log("Result: " + JSON.stringify(inResponse));
      handleFunc(inResponse);
    },
    onFailure: function (inError) {
      console.log("Failed to get system time information");
      console.log("[" + inError.errorCode + "]: " + inError.errorText);
      // To-Do something
      return;
    },
  });
}
```

LunaService app



```
youngheon@youngheon-desktop:~$ ares-launch -s 22 ./LunaService
```

4.WebOS-TV-Sample-Apps

WebOS-TV-Sample-Apps

- We will introduce example sample Webos-TV-Apps from the developer site.
  https://webostv.developer.lge.com/develop/samples
  https://github.com/youngheoncho/webos-hackathon

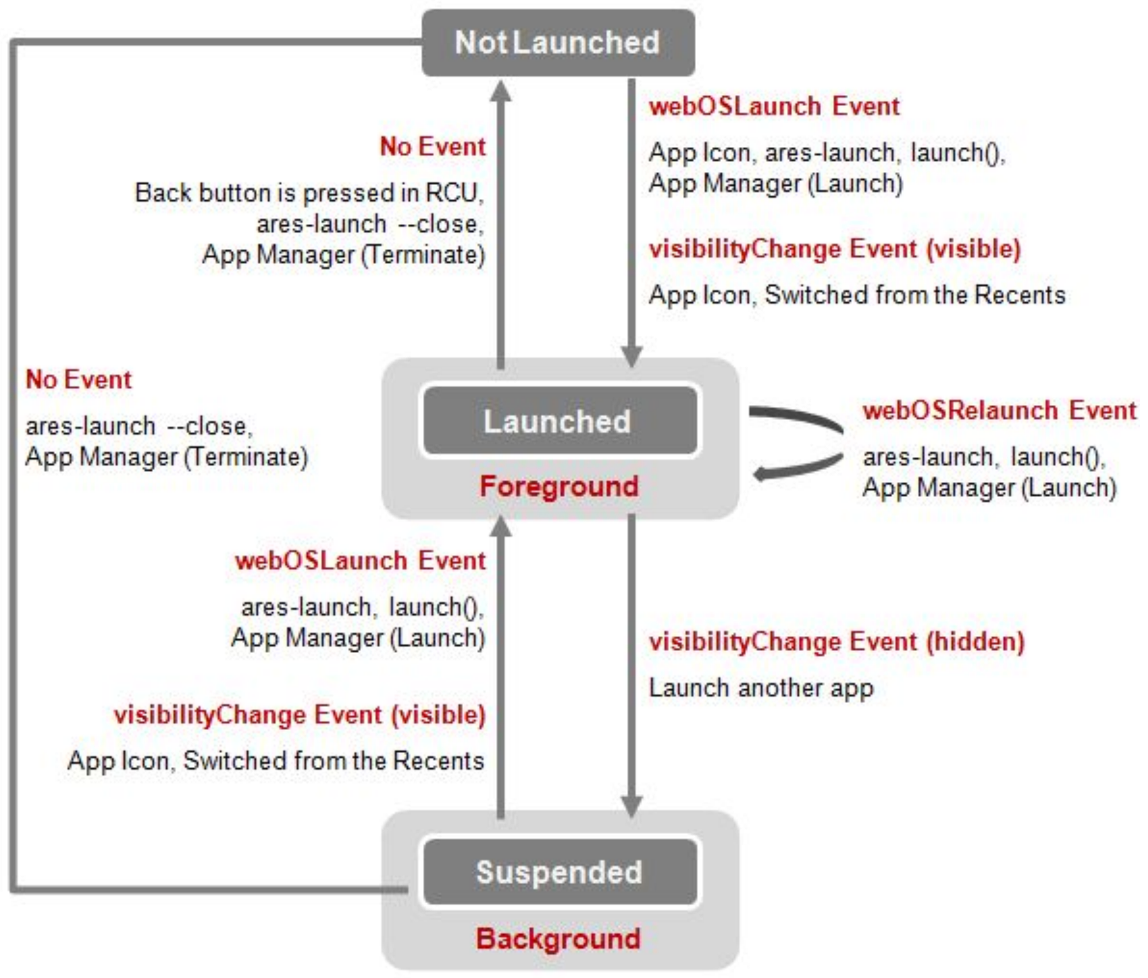
- We will focus on how the WebOS related features are implemented

- More sample apps can be found on the WebOS-TV developer site.

WebOS-TV-Sample-Apps

- AppLifeCycle
- RemoteControl
- BackButtonControl
- WebStorage
- MediaPlayback

AppLifeCycle



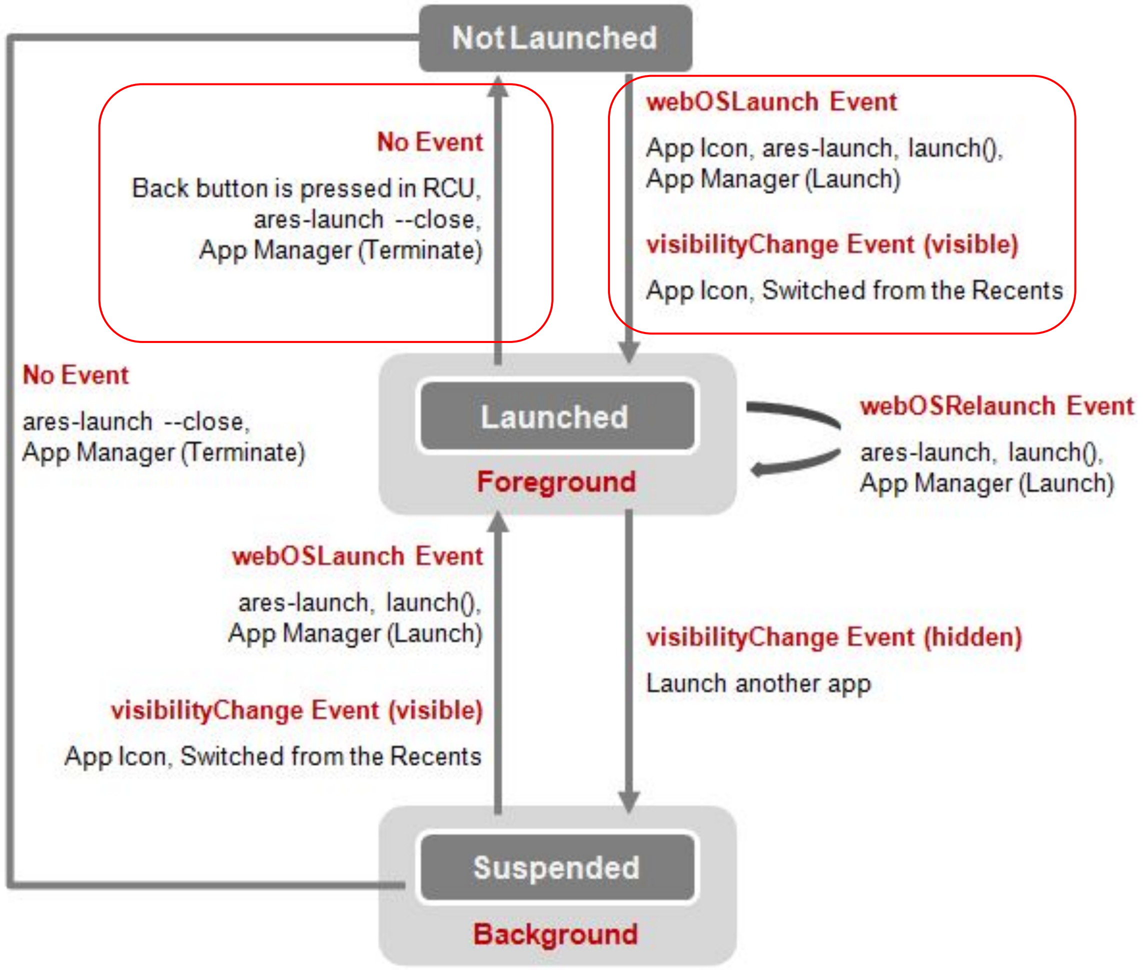
Not Launched
webOSLaunch Event
App Icon, ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
No Event
Back button is pressed in RCU, ares-launch --close, App Manager (Terminate)
No Event
ares-launch --close, App Manager (Terminate)
Launched
Foreground
webOSRelaunch Event
ares-launch, launch(), App Manager (Launch)
webOSLaunch Event
ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
visibilityChange Event (hidden)
Launch another app
Suspended
Background

AppLifeCycle

- The possible states of a webOS TV app are:
    - Not Launched : The app has not been launched or it has been terminated.
    - Launched : The app is running in the foreground.
    - Suspended : The app is suspended in the background.

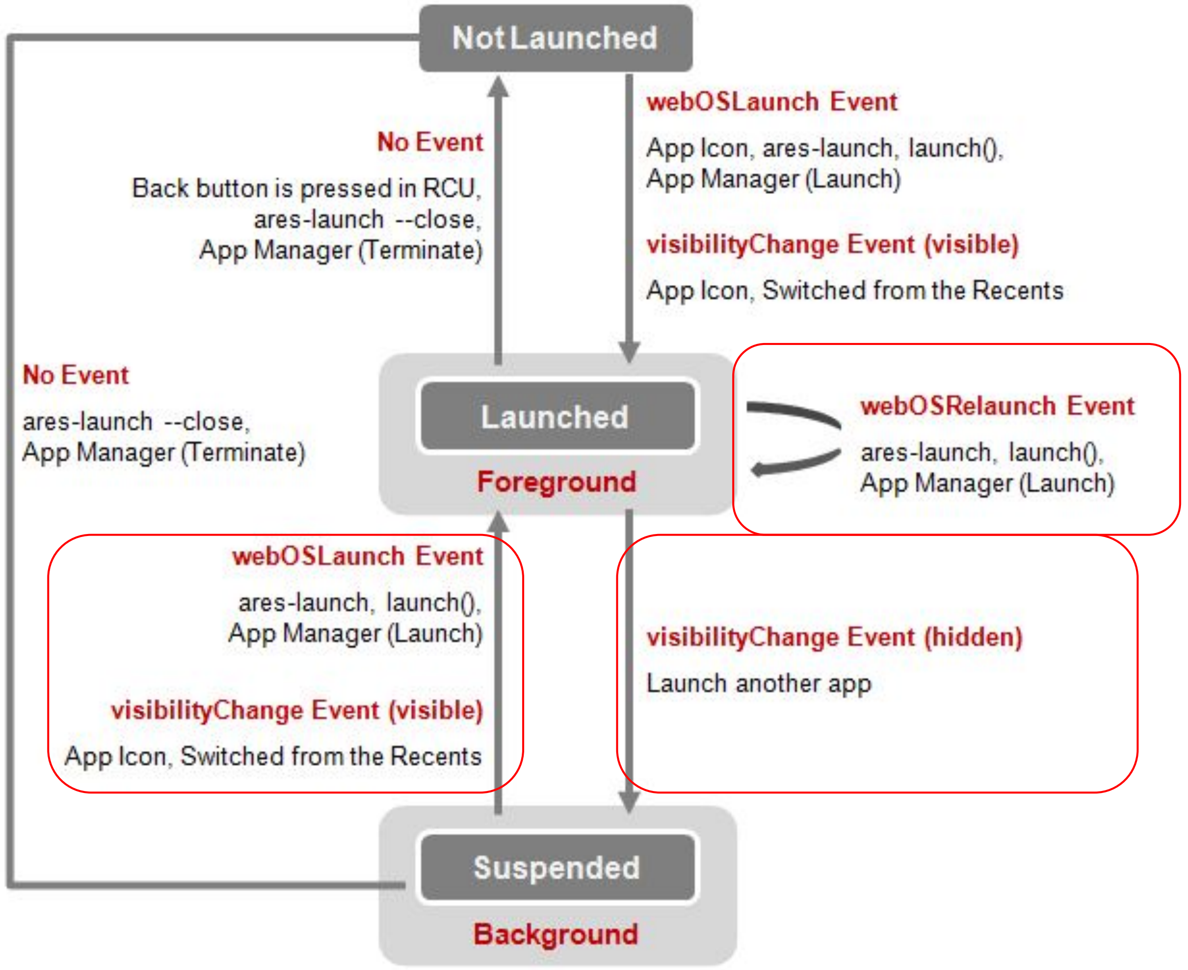
AppLifeCycle

Not Launched – Launched



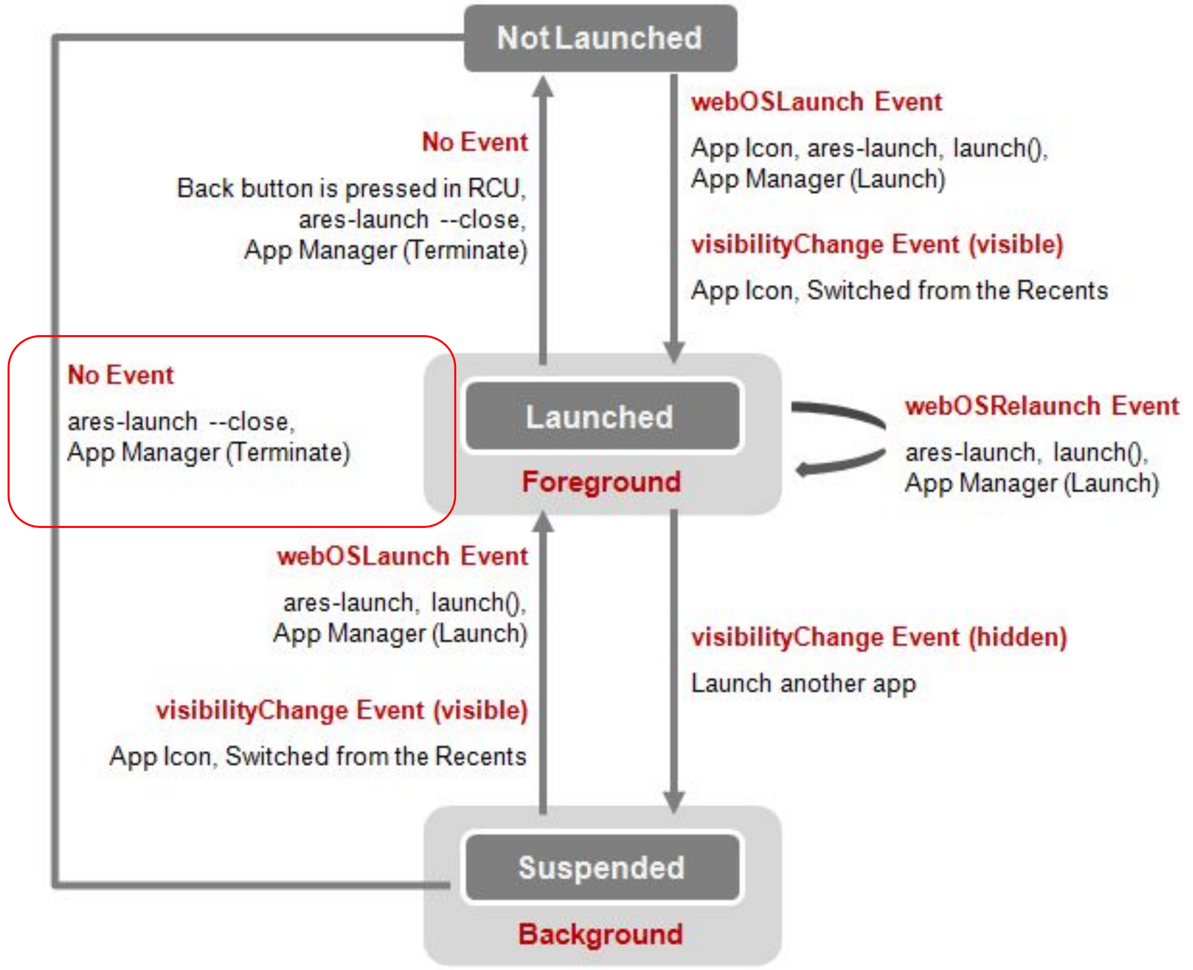
AppLifeCycle

Launched – Suspended


Not Launched
No Event
Back button is pressed in RCU, ares-launch --close, App Manager (Terminate)
webOSLaunch Event
App Icon, ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
No Event
ares-launch --close, App Manager (Terminate)
Launched
Foreground
webOSRelaunch Event
ares-launch, launch(), App Manager (Launch)
webOSLaunch Event
ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
visibilityChange Event (hidden)
Launch another app
Suspended
Background

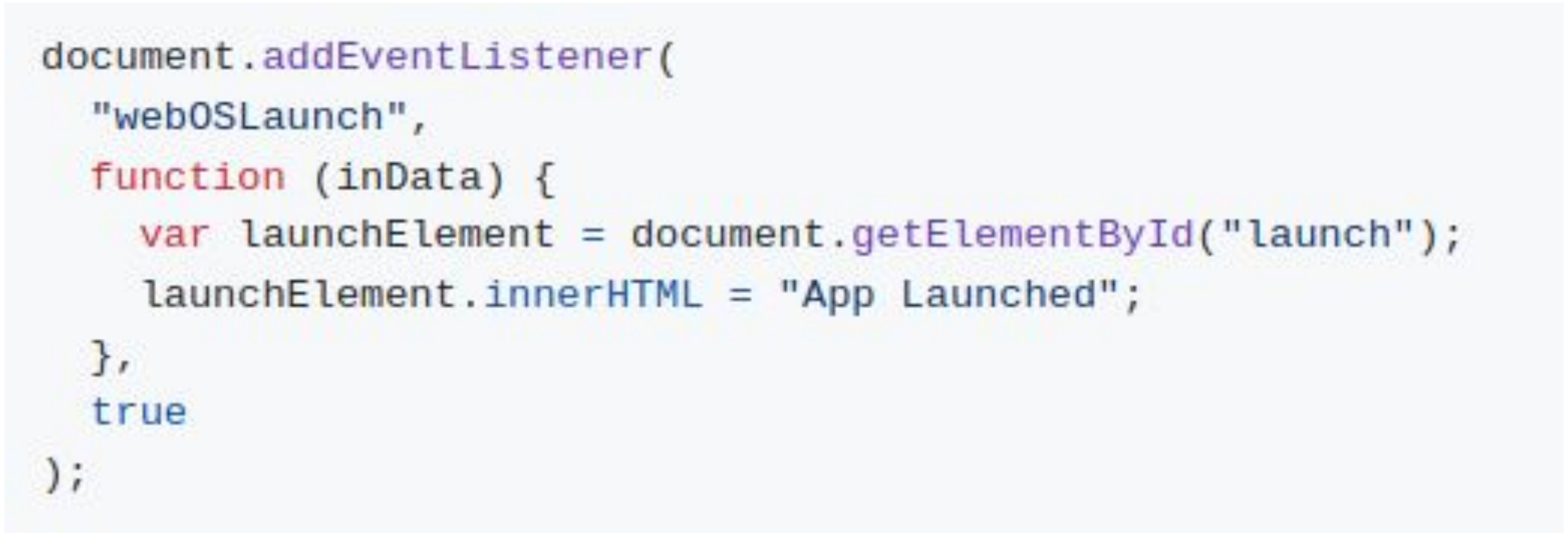
AppLifeCycle

Suspended – Not Launched


Not Launched
No Event
Back button is pressed in RCU, ares-launch --close, App Manager (Terminate)
webOSLaunch Event
App Icon, ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
No Event
ares-launch --close, App Manager (Terminate)
Launched
Foreground
webOSRelaunch Event
ares-launch, launch(), App Manager (Launch)
webOSLaunch Event
ares-launch, launch(), App Manager (Launch)
visibilityChange Event (visible)
App Icon, Switched from the Recents
visibilityChange Event (hidden)
Launch another app
Suspended
Background

AppLifeCycle

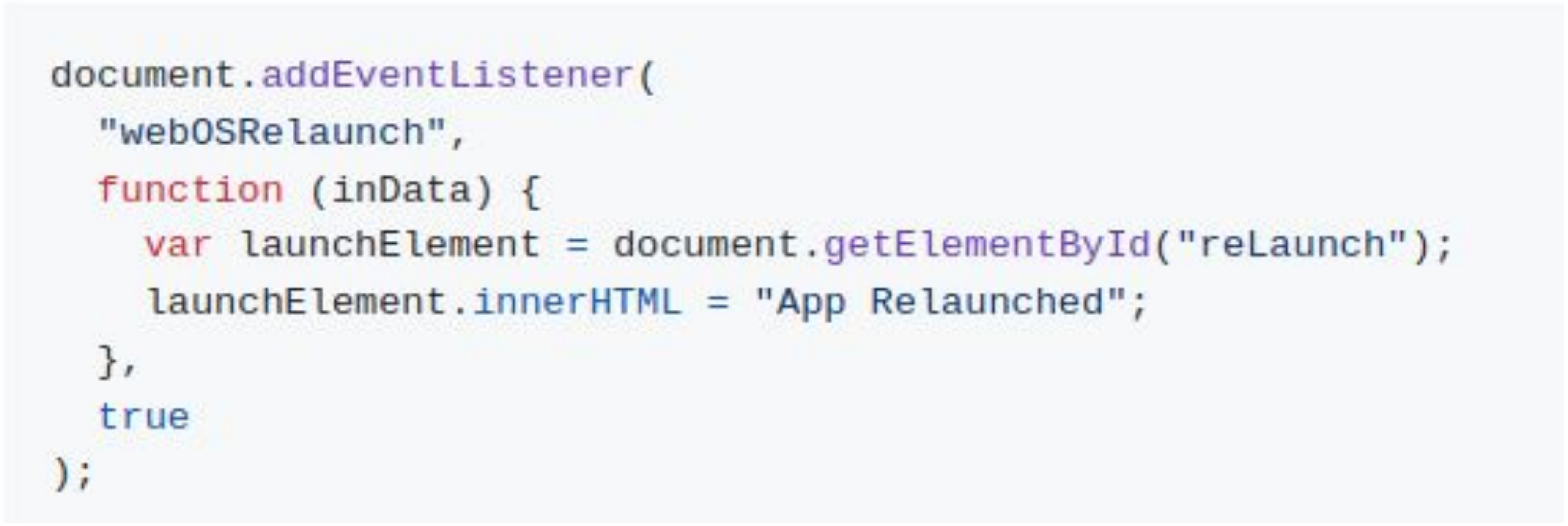
webOSLaunch



```
document.addEventListener(
  "webOSLaunch",
  function (inData) {
    var launchElement = document.getElementById("launch");
    launchElement.innerHTML = "App Launched";
  },
  true
);
```

AppLifeCycle

webOSRelaunch



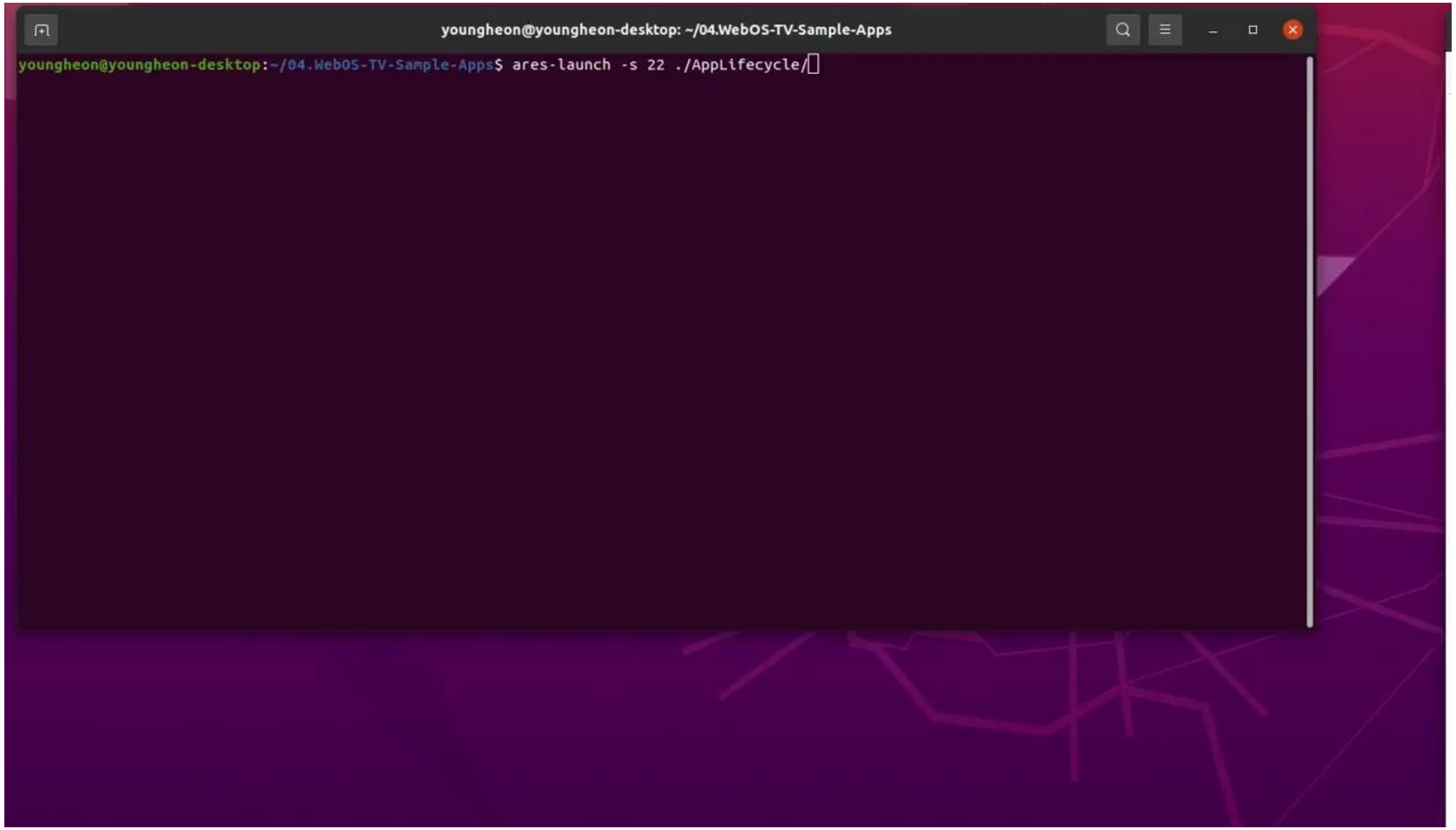
```
document.addEventListener(
  "webOSRelaunch",
  function (inData) {
    var launchElement = document.getElementById("reLaunch");
    launchElement.innerHTML = "App Relaunched";
  },
  true
);
```

AppLifeCycle

visibilityChange

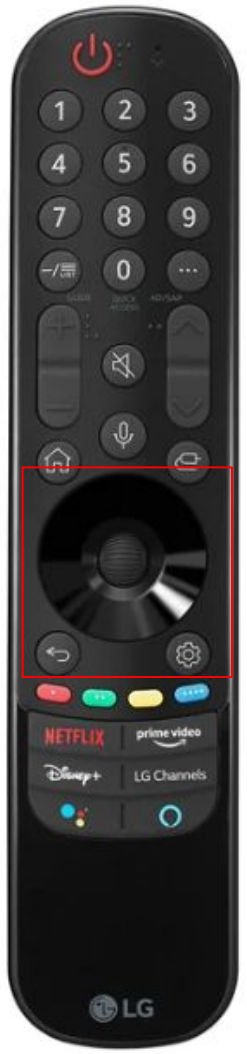
```
document.addEventListener(
  visibilityChange,
  function () {
    if (document[hidden]) {
      var visibilityElement = document.getElementById("visibility");
      visibilityElement.innerHTML = "App Hidden";
    } else {
      var visibilityElement = document.getElementById("visibility");
      visibilityElement.innerHTML = "App Shown";
    }
  },
  true
);
```

AppLifeCycle



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 ./AppLifecycle/
```

Remote Control

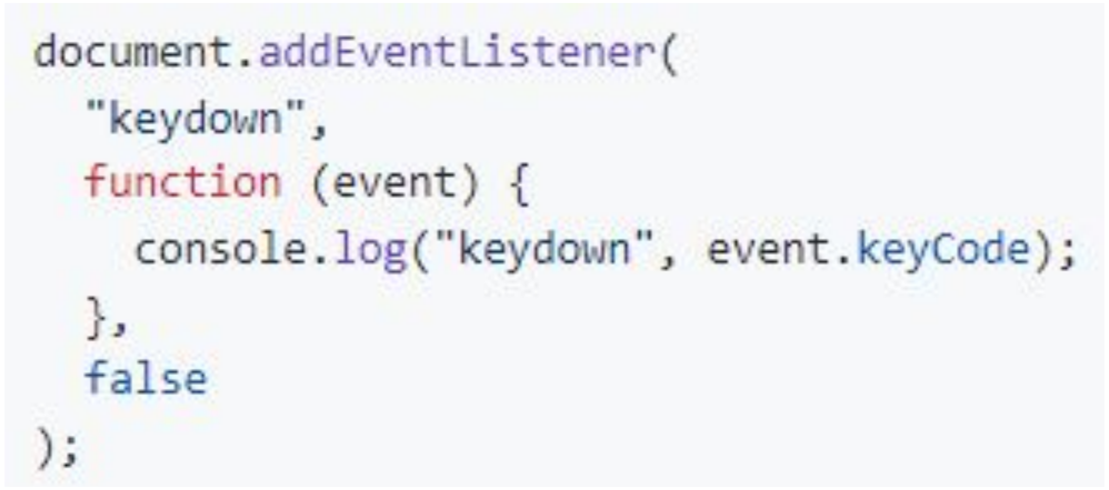


- OK – Select/Enter
- Wheel – Scrolls content on hover
- Navigation – Up/Right/Down/Left
- Back
  - Short Press : Return to Previous page / exit app
  - Longpress: Return to the last input mode

Remote Control - Keycode

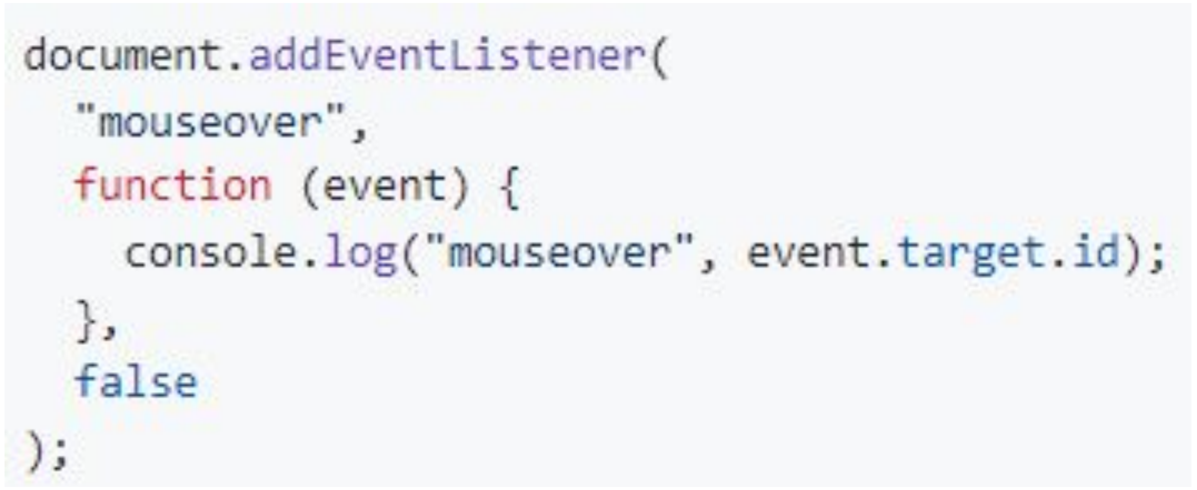
Button	Keycode
Left	37
Up	38
Right	39
Down	40
OK	13
Back	461

Remote Control - Keycode



```
document.addEventListener(
  "keydown",
  function (event) {
    console.log("keydown", event.keyCode);
  },
  false
);
```

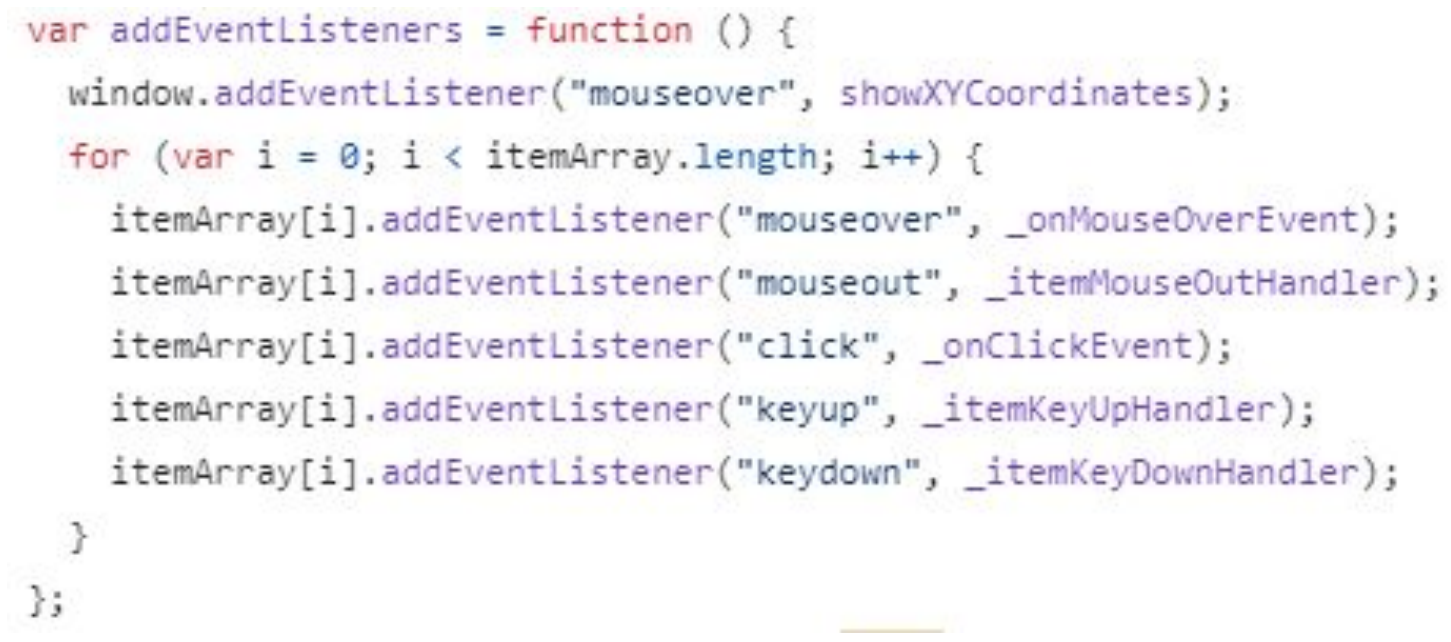
Remote Control



```
document.addEventListener(
  "mouseover",
  function (event) {
    console.log("mouseover", event.target.id);
  },
  false
);
```

Remote Control

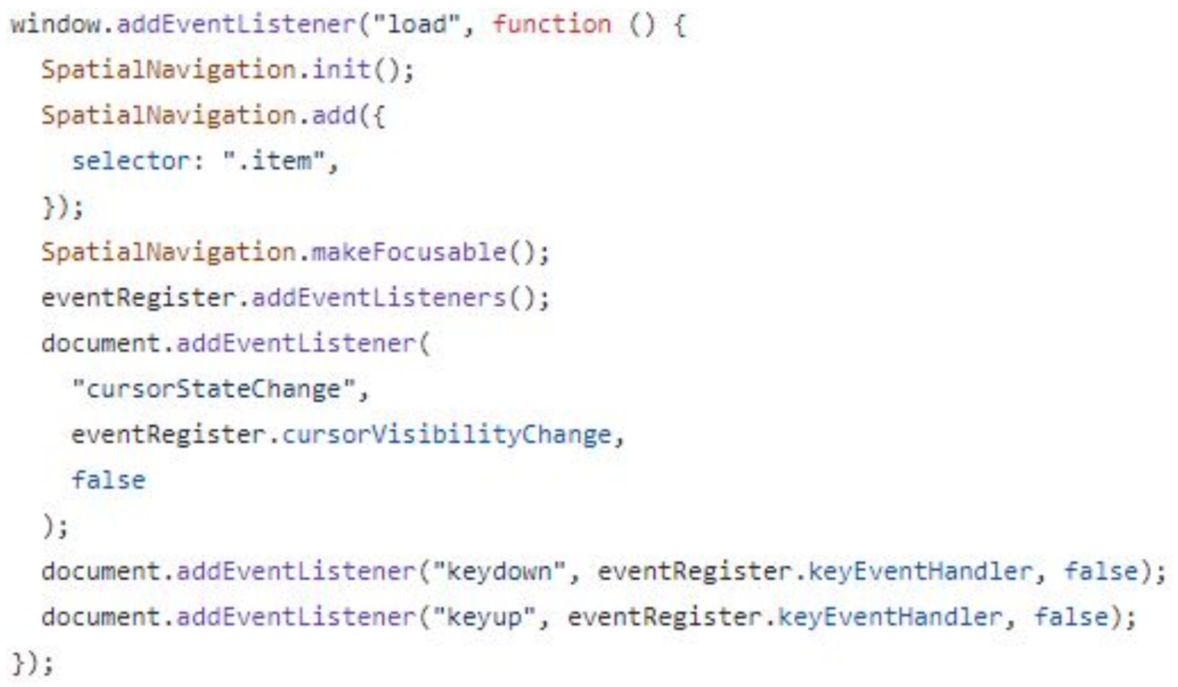
RemoteControl/js/index.js



```
var addEventListeners = function () {
  window.addEventListener("mouseover", showXYCoordinates);
  for (var i = 0; i < itemArray.length; i++) {
    itemArray[i].addEventListener("mouseover", _onMouseOverEvent);
    itemArray[i].addEventListener("mouseout", _itemMouseOutHandler);
    itemArray[i].addEventListener("click", _onClickEvent);
    itemArray[i].addEventListener("keyup", _itemKeyUpHandler);
    itemArray[i].addEventListener("keydown", _itemKeyDownHandler);
  }
};
```

Remote Control

RemoteControl/js/index.js



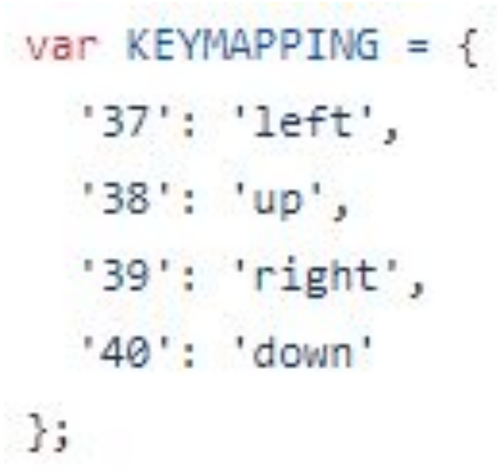
```
window.addEventListener("load", function () {
  SpatialNavigation.init();
  SpatialNavigation.add({
    selector: ".item",
  });
  SpatialNavigation.makeFocusable();
  eventRegister.addEventListeners();
  document.addEventListener(
    "cursorStateChange",
    eventRegister.cursorVisibilityChange,
    false
  );
  document.addEventListener("keydown", eventRegister.keyEventHandler, false);
  document.addEventListener("keyup", eventRegister.keyEventHandler, false);
});
```

Remote Control - Spatial Navigation

- Javascript-based implementation of Spatial Navigation.
- Used for moving focus between UI elements with 4-way navigation keys.
- Refer to the github repository https://github.com/luke-chang/js-spatial-navigation

Remote Control - Spatial Navigation

RemoteControl/lib/spatial_navigation.js

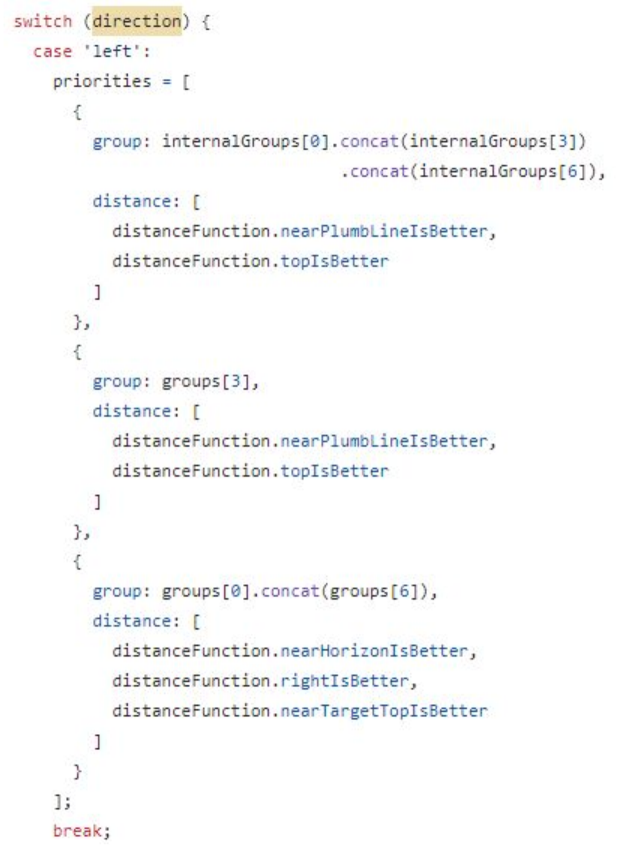


```
var KEYMAPPING = {
  '37': 'left',
  '38': 'up',
  '39': 'right',
  '40': 'down'
};
```

navigate(target, direction, candidates, config) function

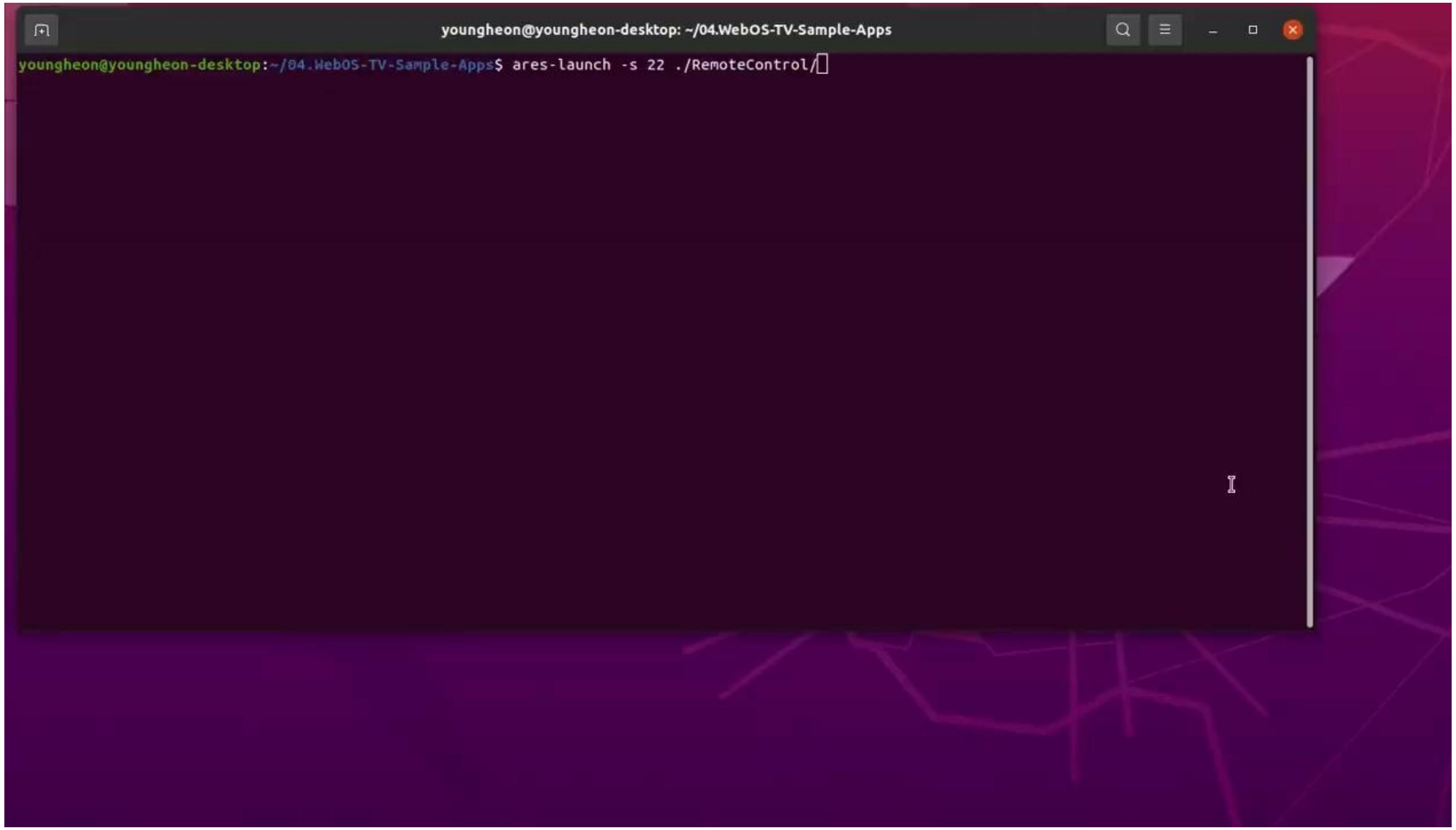
Remote Control - Spatial Navigation

RemoteControl/lib/spatial_navigation.js
navigate(target, direction, candidates, config) function



```
switch (direction) {
  case 'left':
    priorities = [
      {
        group: internalGroups[0].concat(internalGroups[3])
                                .concat(internalGroups[6]),
        distance: [
          distanceFunction.nearPlumbLineIsBetter,
          distanceFunction.topIsBetter
        ]
      },
      {
        group: groups[3],
        distance: [
          distanceFunction.nearPlumbLineIsBetter,
          distanceFunction.topIsBetter
        ]
      },
      {
        group: groups[0].concat(groups[6]),
        distance: [
          distanceFunction.nearHorizonIsBetter,
          distanceFunction.rightIsBetter,
          distanceFunction.nearTargetTopIsBetter
        ]
      }
    ];
    break;
```

Remote Control



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 ./RemoteControl/
```

Remote Control

basic processing principles for user inputs of the Remote Control.

- Selectable UI elements must be fully navigable by the screen cursor and 4-way navigation keys (Up, Down, Left, and Right)
- Selectable UI elements must act in the same way when controlled by the screen cursor and when controlled by the OK button.
- One of the UI elements must be focused.
    - When the cursor disappears from the screen by entering the navigation keys, the focus by the cursor must be switched to the focus by the navigation keys.
    - Contrary to the above, when the cursor is activated and is moving onto a UI element, the focus by the navigation keys must be switched to the focus by the cursor on the current position.
- A selection effect is mandatory to show which element is activated. The possible selection effects are animation, highlight, color, or size change, etc.
- The page and list scroll can be appropriately controlled using the wheel of the Magic Remote.

Back Button Control

- This app demonstrates that the back button control function based on the browser history is supported through the use of the DOM's history object.
- Back button is used to move to the home UI or previous page
- history object: pushState & popState
- This app uses Spatial Navigation

Back Button Control

- This app is divided into 2 sub apps for comparison. “com.sample.backbuttonkey”, “com.sample.backbuttonhistory”

- These apps are identical in function

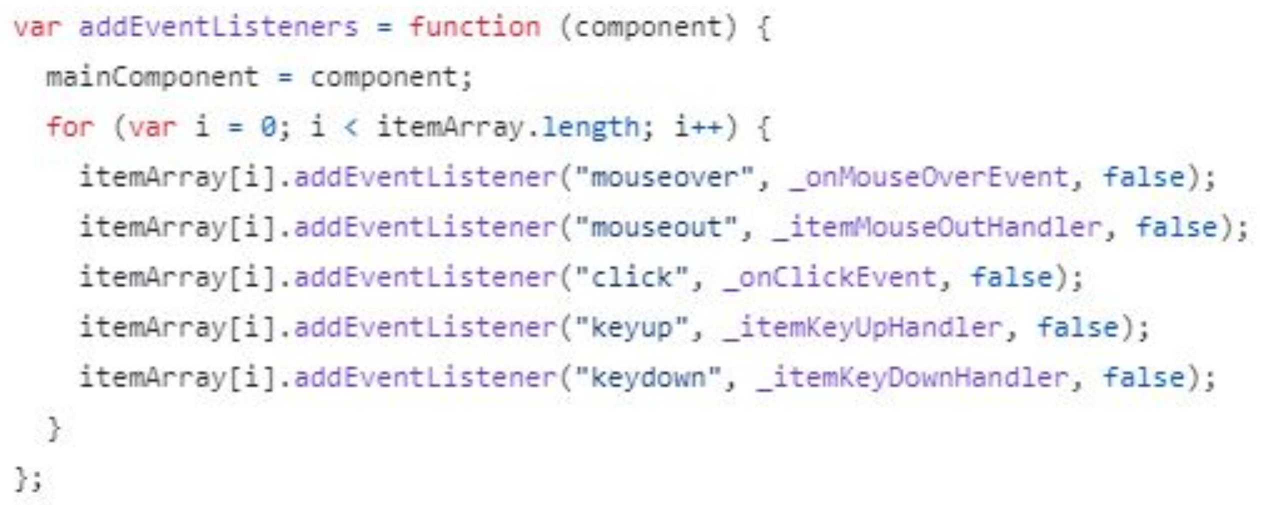
- However they handle back button using different methods
    - com.sample.backbuttonkey : keydown event
    - com.sample.backbuttonhistory : history object

Back Button Control

- Spatial Navigation is implemented the same way as in Remote Control app
- Function to add eventlisteners to elements is defined in the navigationSupport.js
- History management(moving to previous page) is handled in index.js

Back Button Control

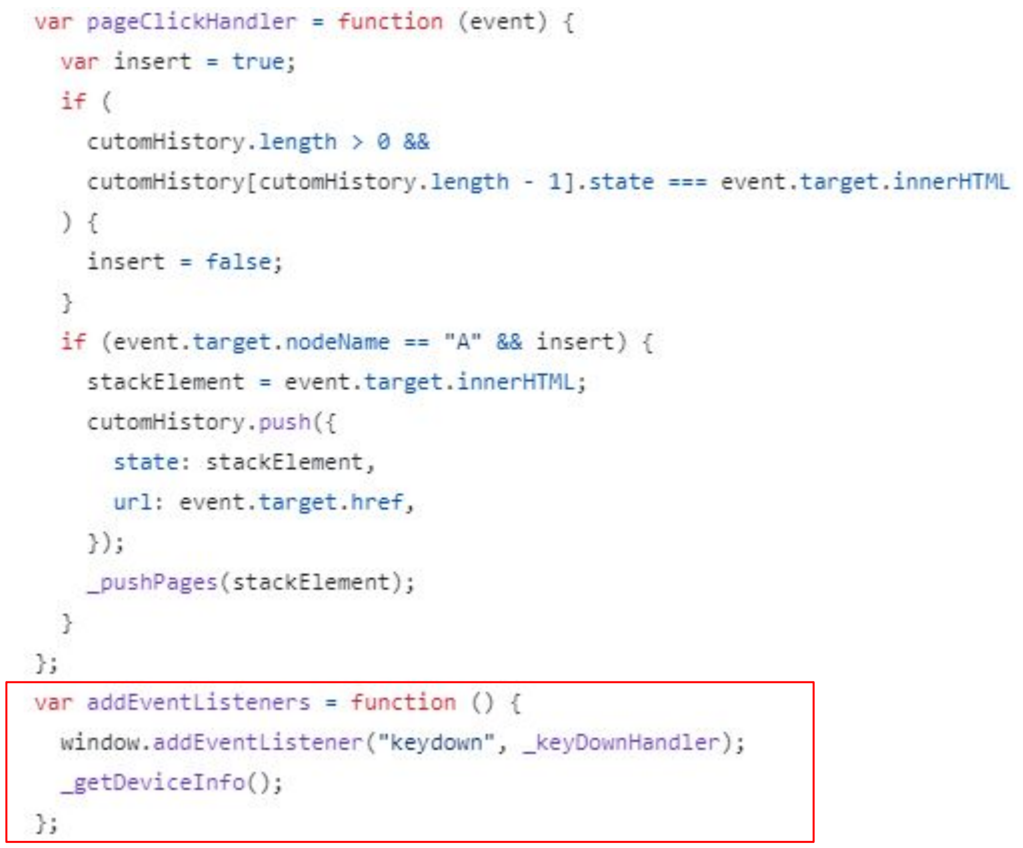
com.sample.backbuttonkey/js/navigationSupport.js
com.sample.backbuttonhistory/js/navigationSupport.js



```
var addEventListeners = function (component) {
  mainComponent = component;
  for (var i = 0; i < itemArray.length; i++) {
    itemArray[i].addEventListener("mouseover", _onMouseOverEvent, false);
    itemArray[i].addEventListener("mouseout", _itemMouseOutHandler, false);
    itemArray[i].addEventListener("click", _onClickEvent, false);
    itemArray[i].addEventListener("keyup", _itemKeyUpHandler, false);
    itemArray[i].addEventListener("keydown", _itemKeyDownHandler, false);
  }
};
```

Back Button Control

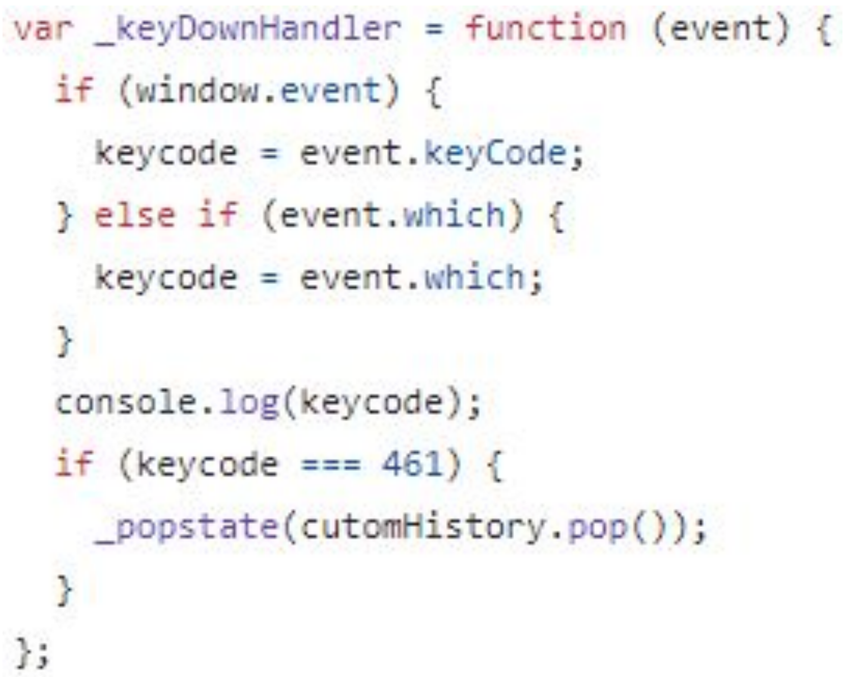
com.sample.backbuttonkey/js/index.js



```
var pageClickHandler = function (event) {
  var insert = true;
  if (
    cutomHistory.length > 0 &&
    cutomHistory[cutomHistory.length - 1].state === event.target.innerHTML
  ) {
    insert = false;
  }
  if (event.target.nodeName == "A" && insert) {
    stackElement = event.target.innerHTML;
    cutomHistory.push({
      state: stackElement,
      url: event.target.href,
    });
    _pushPages(stackElement);
  }
};
var addEventListeners = function () {
  window.addEventListener("keydown", _keyDownHandler);
  _getDeviceInfo();
};
```

Back Button Control

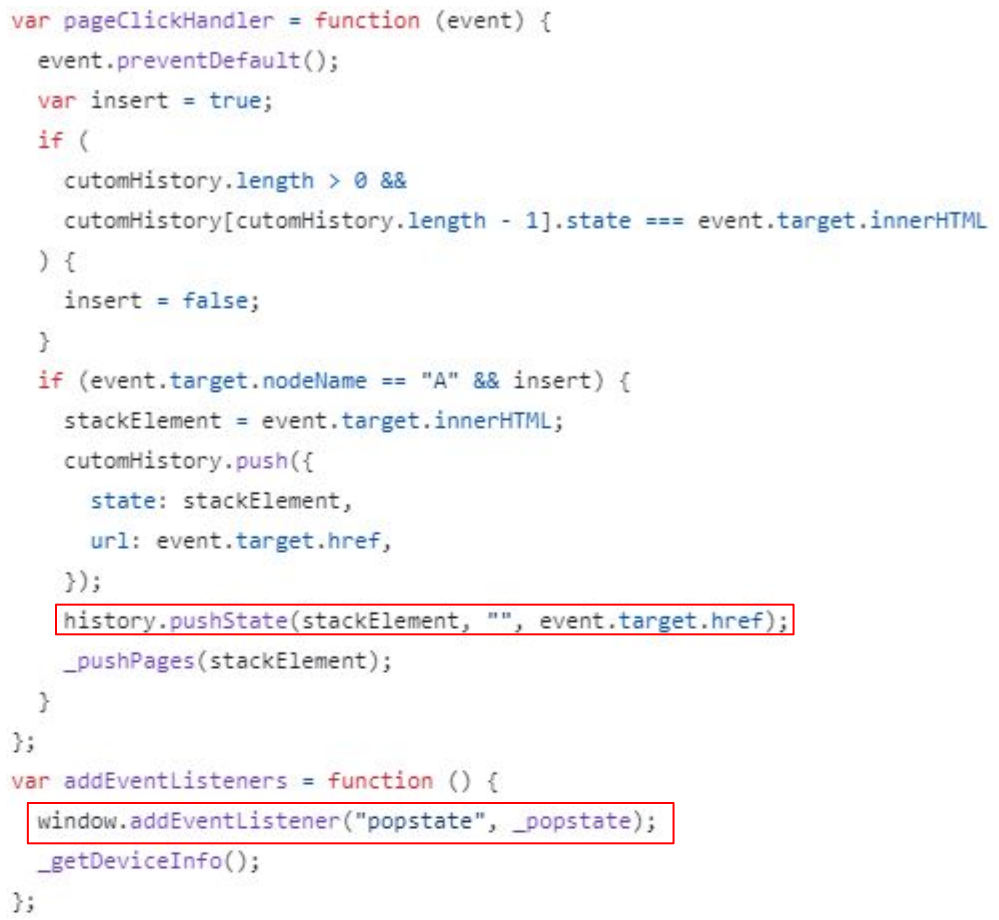
com.sample.backbuttonkey/js/index.js



```
var _keyDownHandler = function (event) {
  if (window.event) {
    keycode = event.keyCode;
  } else if (event.which) {
    keycode = event.which;
  }
  console.log(keycode);
  if (keycode === 461) {
    _popstate(cutomHistory.pop());
  }
};
```

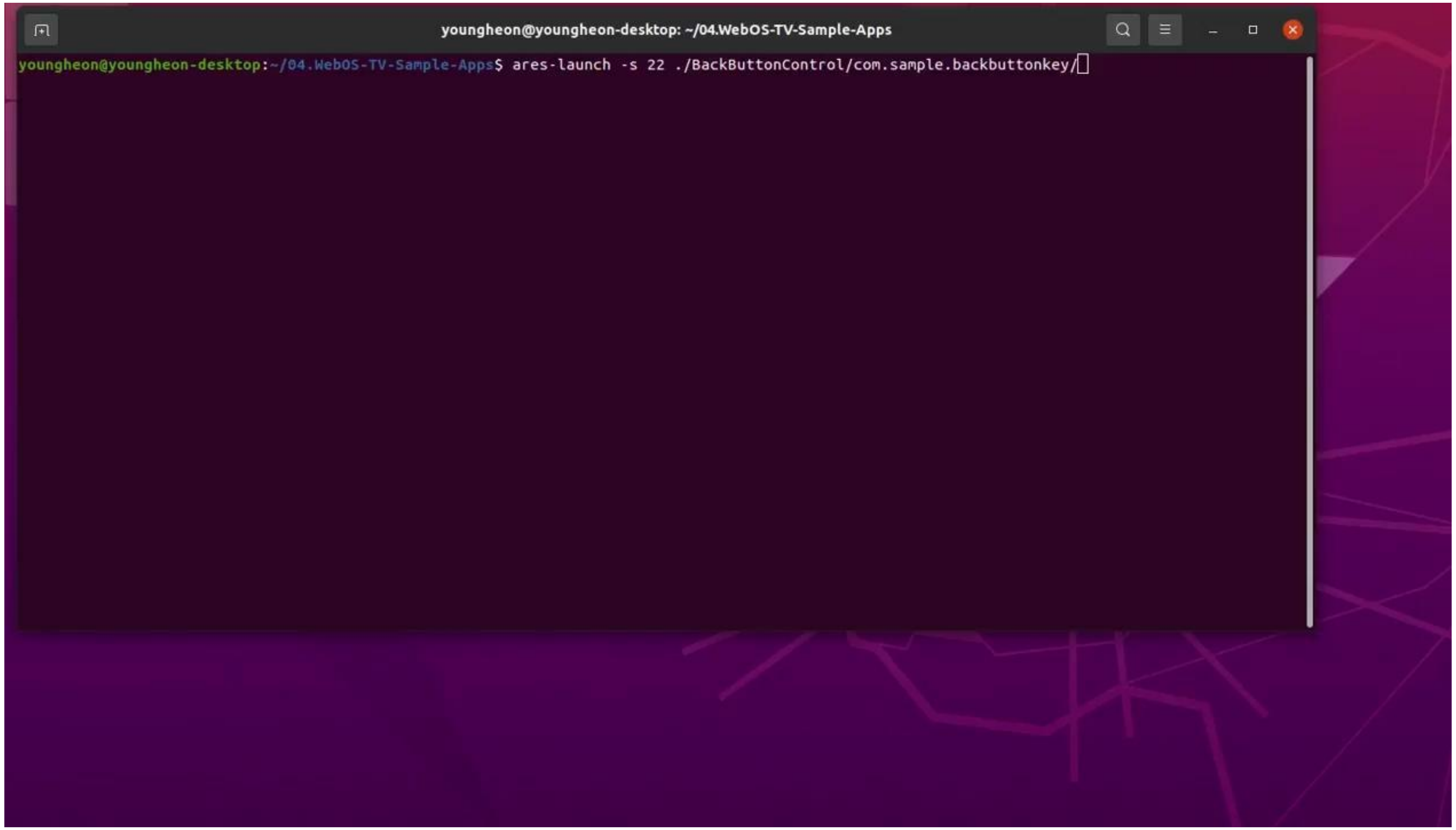
Back Button Control

com.sample.backbuttonhistory/js/index.js



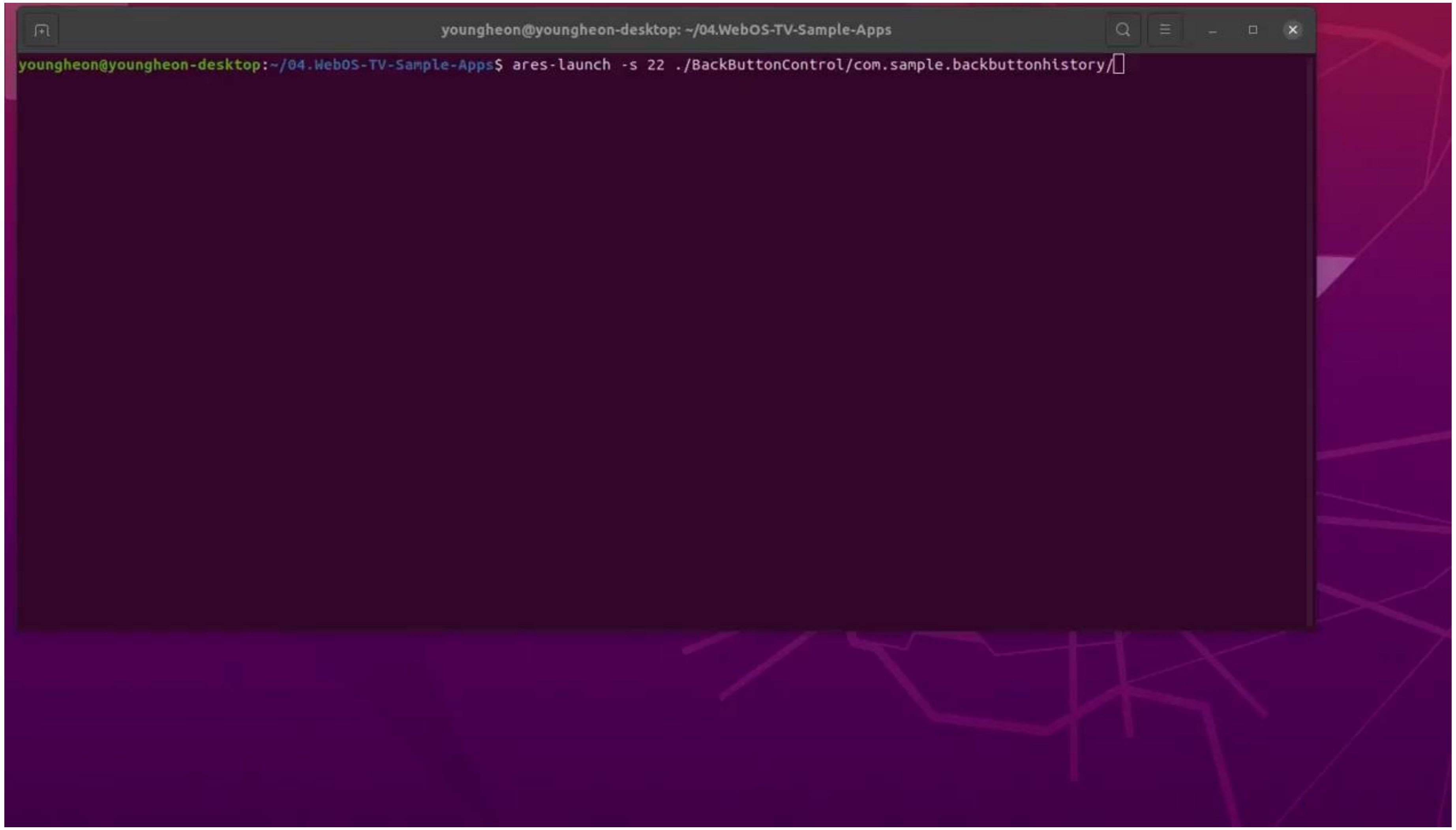
```
var pageClickHandler = function (event) {
  event.preventDefault();
  var insert = true;
  if (
    cutomHistory.length > 0 &&
    cutomHistory[cutomHistory.length - 1].state === event.target.innerHTML
  ) {
    insert = false;
  }
  if (event.target.nodeName == "A" && insert) {
    stackElement = event.target.innerHTML;
    cutomHistory.push({
      state: stackElement,
      url: event.target.href,
    });
    history.pushState(stackElement, "", event.target.href);
    _pushPages(stackElement);
  }
};
var addEventListeners = function () {
  window.addEventListener("popstate", _popstate);
  _getDeviceInfo();
};
```

Back Button Control - backbuttonkey



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 ./BackButtonControl/com.sample.backbuttonkey/
```

Back Button Control - backbuttonhistory



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 ./BackButtonControl/com.sample.backbuttonhistory/
```

WebStorage

- web storage is a web app development method and protocol used for storing data in a web browser.
- supports persistent data storage
- similar to cookies, but with a significantly enhanced capacity and no information stored in the HTTP request header
- WebOS TV offers two different storage areas: local storage and session storage

WebStorage - Local Storage

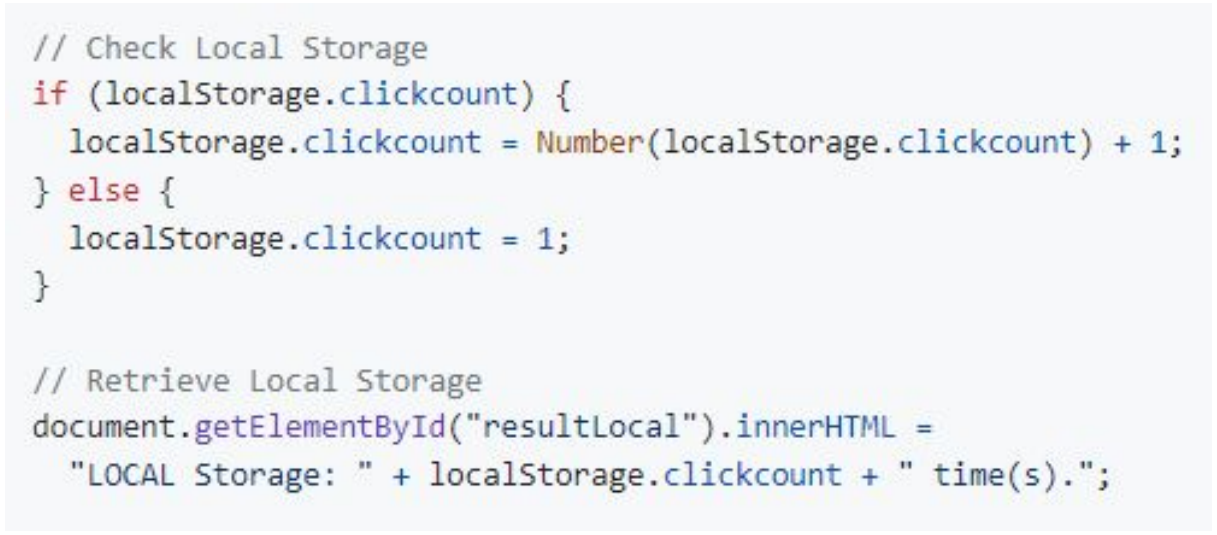
- Any web apps can store their data locally with local storage in webOS TV
- The storage limit is far larger(at least 5 MB) than cookies, and the information is never transferred to the server
- The data in local storage can be deleted by the "Initial setting" menu (factory reset)
- It can also be deleted on the following conditions:
    - Packaged app: The data is deleted when users update or remove your app on webOS TV.
    - Hosted app: The data is deleted when the total usage of data reaches the limit.

WebStorage - Session Storage

- The session storage keeps its value on the browser only for the duration of the session.
- It maintains a storage area that is available for the duration of the page session
- Stored values are cleared if you turn off your app after using the session storage in your app

WebStorage - Local Storage

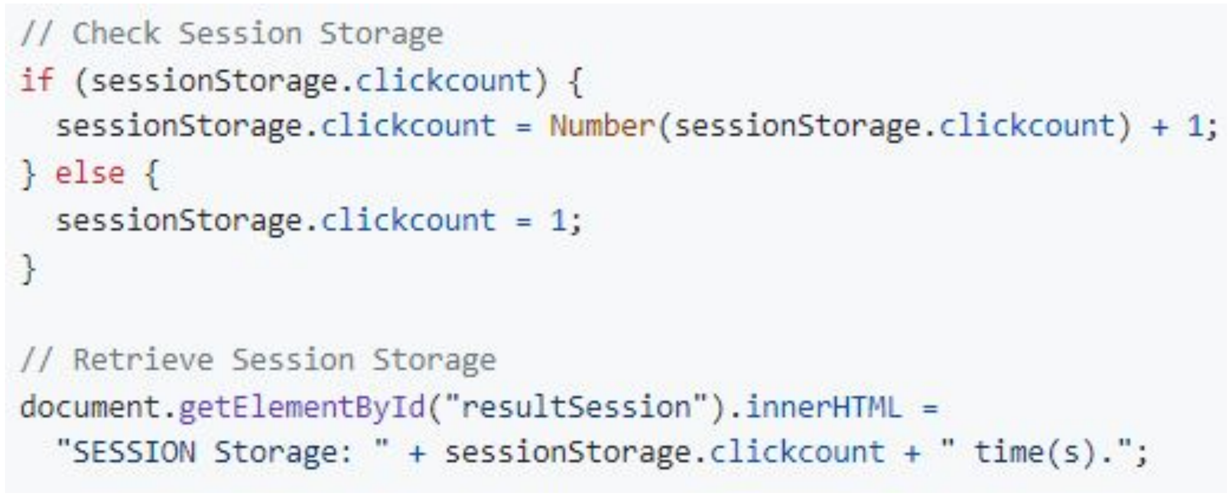
WebStorage/js/webstorage.js



```
// Check Local Storage
if (localStorage.clickcount) {
  localStorage.clickcount = Number(localStorage.clickcount) + 1;
} else {
  localStorage.clickcount = 1;
}

// Retrieve Local Storage
document.getElementById("resultLocal").innerHTML =
  "LOCAL Storage: " + localStorage.clickcount + " time(s).";
```

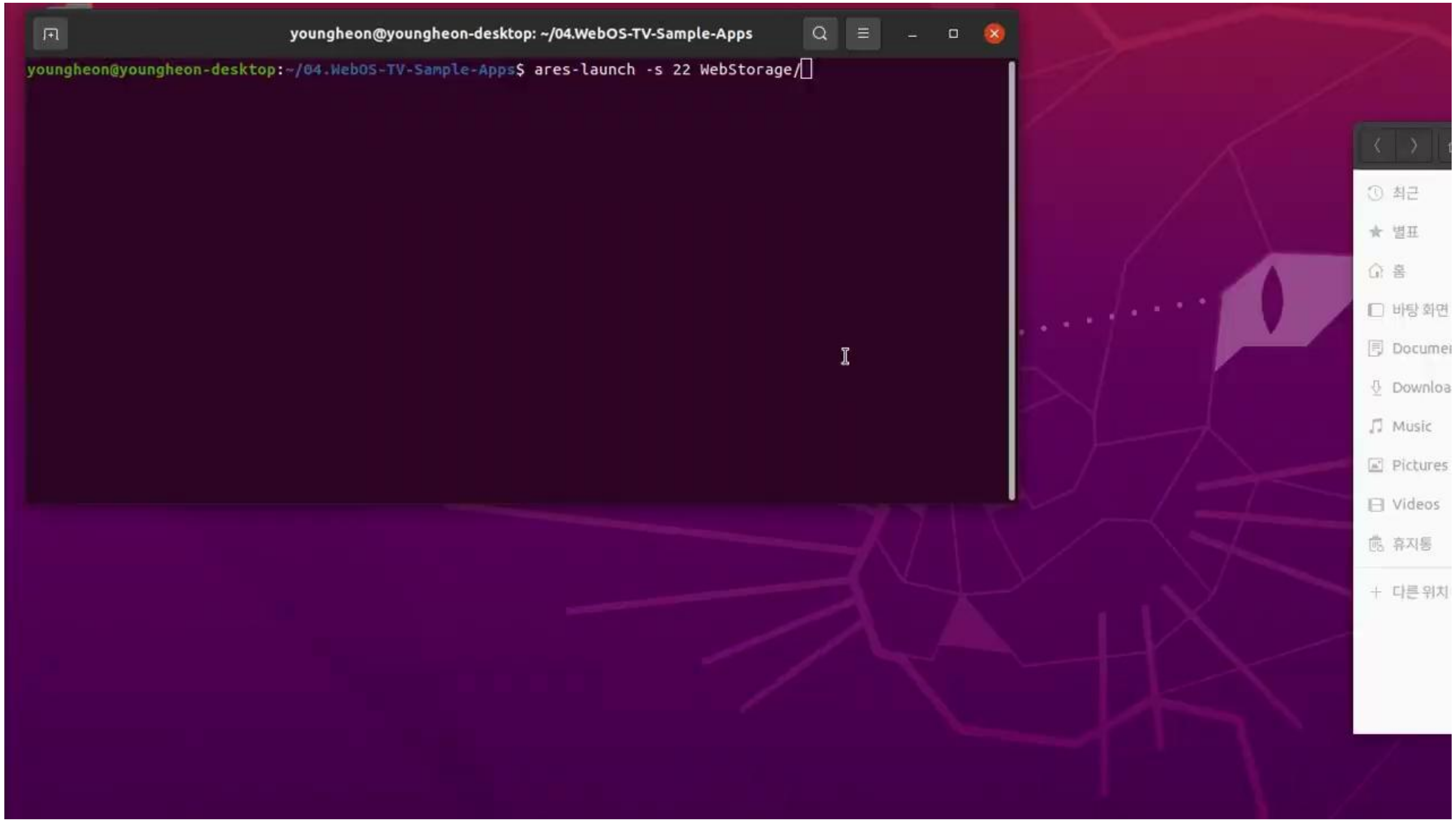
WebStorage - Local Storage

WebStorage/js/webstorage.js



```
// Check Session Storage
if (sessionStorage.clickcount) {
  sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
  sessionStorage.clickcount = 1;
}

// Retrieve Session Storage
document.getElementById("resultSession").innerHTML =
  "SESSION Storage: " + sessionStorage.clickcount + " time(s).";
```

WebStorage



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 WebStorage/
```

Media Playback

This app

- shows how to implement media playback in a web app with adaptive bitrate streaming

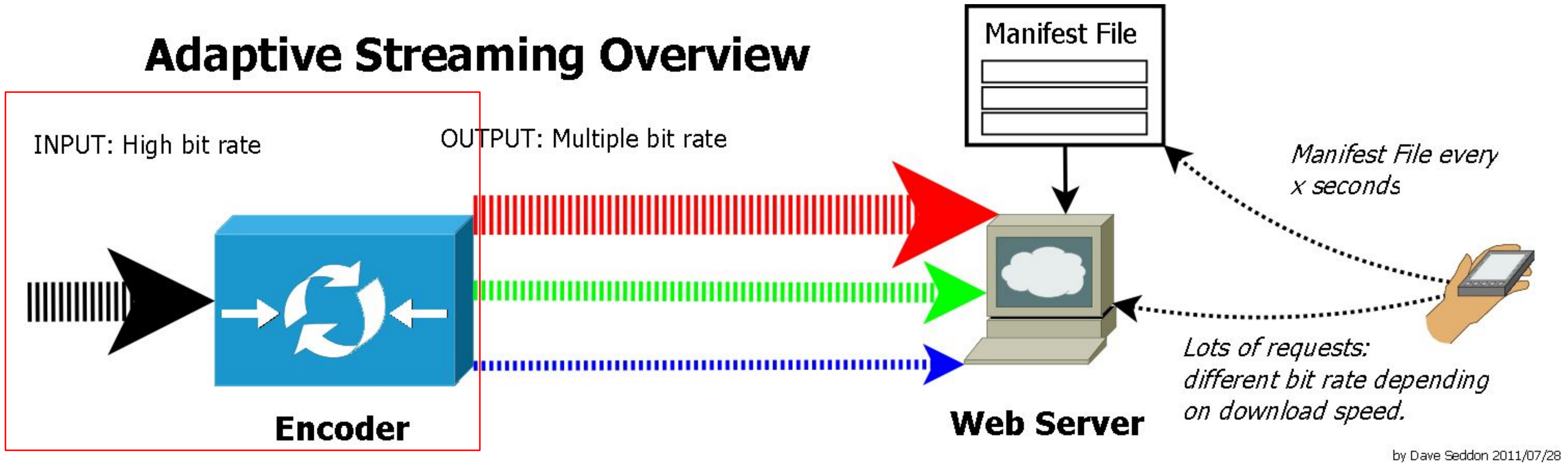
- uses Shaka Player
  - uses Media Source Extensions(MSE) instead of plugins of FLASH
  - uses shaka playerDash and HLS URLs

Media Playback - Adaptive bitrate streaming

- Adaptive bitrate streaming is a technique used in streaming multimedia over computer networks.
- based almost exclusively on HTTP, and are designed to work efficiently over large distributed HTTP networks.
- Adaptive bitrate streaming works by detecting a user's bandwidth and CPU capacity in real time, adjusting the quality of the media stream accordingly.
- enables very little buffering, fast start time and a good experience for both high-end and low-end connections.

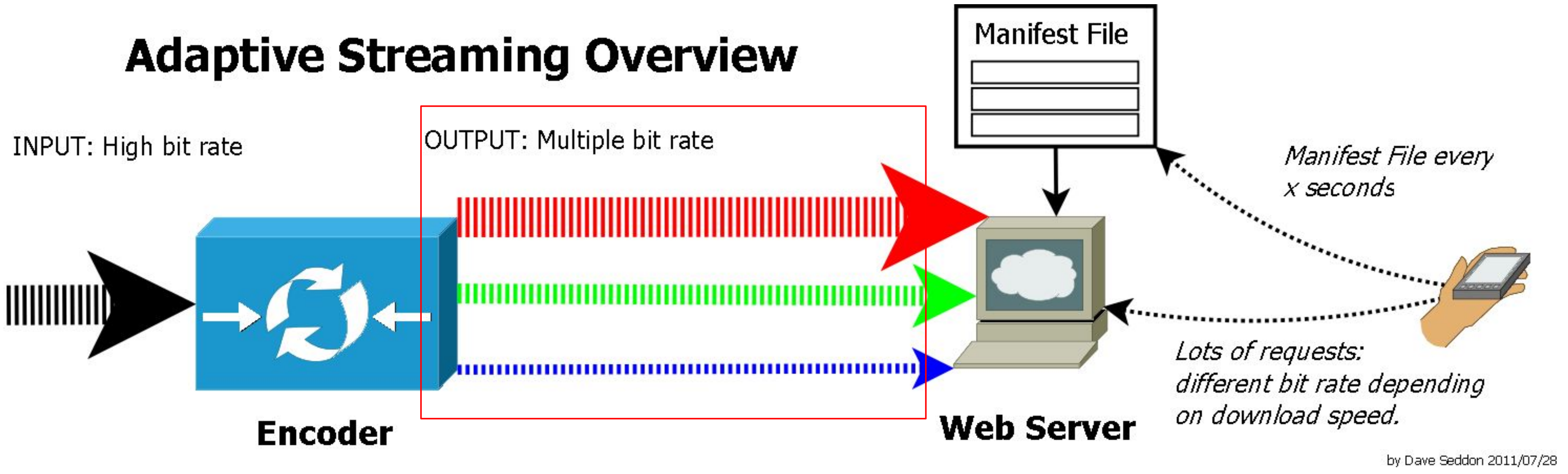
Media Playback - Adaptive bitrate streaming

A single source media of video or audio type & high bit rate is received in the encoder.


Adaptive Streaming Overview
INPUT: High bit rate
OUTPUT: Multiple bit rate
Encoder
Manifest File
Web Server
Manifest File every x seconds
Lots of requests: different bit rate depending on download speed.
by Dave Seddon 2011/07/28

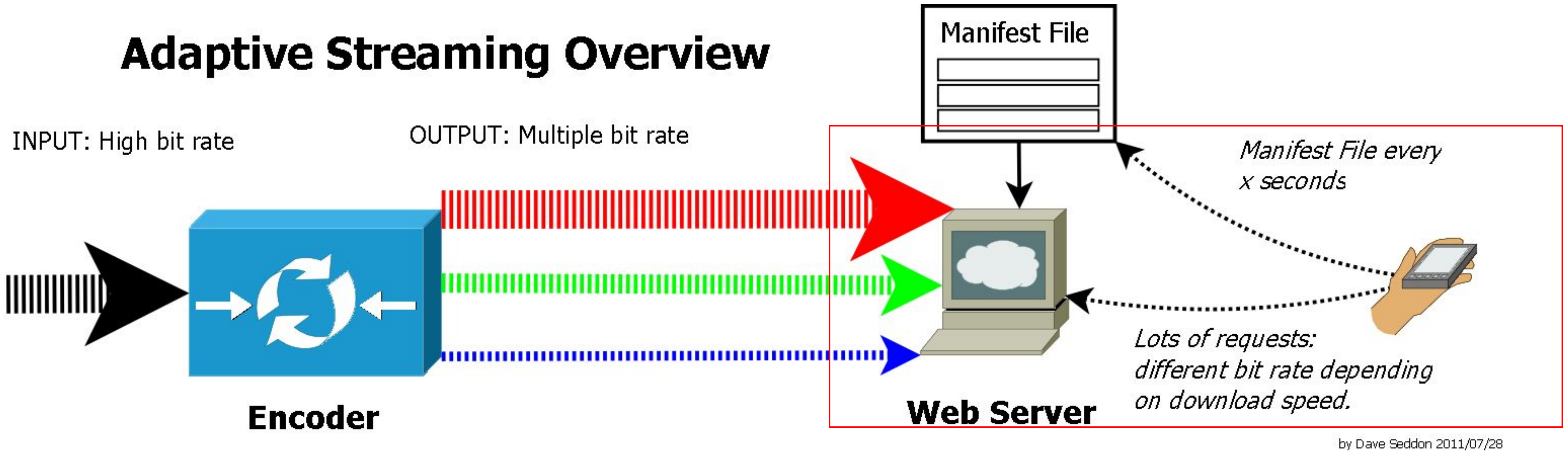
Media Playback - Adaptive bitrate streaming

The source is encoded at multiple bit rates then sent to the player client.


Adaptive Streaming Overview
INPUT: High bit rate
OUTPUT: Multiple bit rate
Manifest File
Encoder
Web Server
Manifest File every x seconds
Lots of requests: different bit rate depending on download speed.
by Dave Seddon 2011/07/28

Media Playback - Adaptive bitrate streaming

The player client switches between streaming the different encodings depending on available resources at the moment.


Adaptive Streaming Overview
INPUT: High bit rate
OUTPUT: Multiple bit rate
Manifest File
Manifest File every x seconds
Encoder
Web Server
Lots of requests: different bit rate depending on download speed.
by Dave Seddon 2011/07/28

Media Playback - Shaka Player

- Shaka Player is an open-source JavaScript library for adaptive media.
- plays adaptive media formats (such as DASH, HLS) in a browser
- uses MediaSource Extensions instead of using plugins or Flash.
- Refer to github repository https://github.com/shaka-project/shaka-player

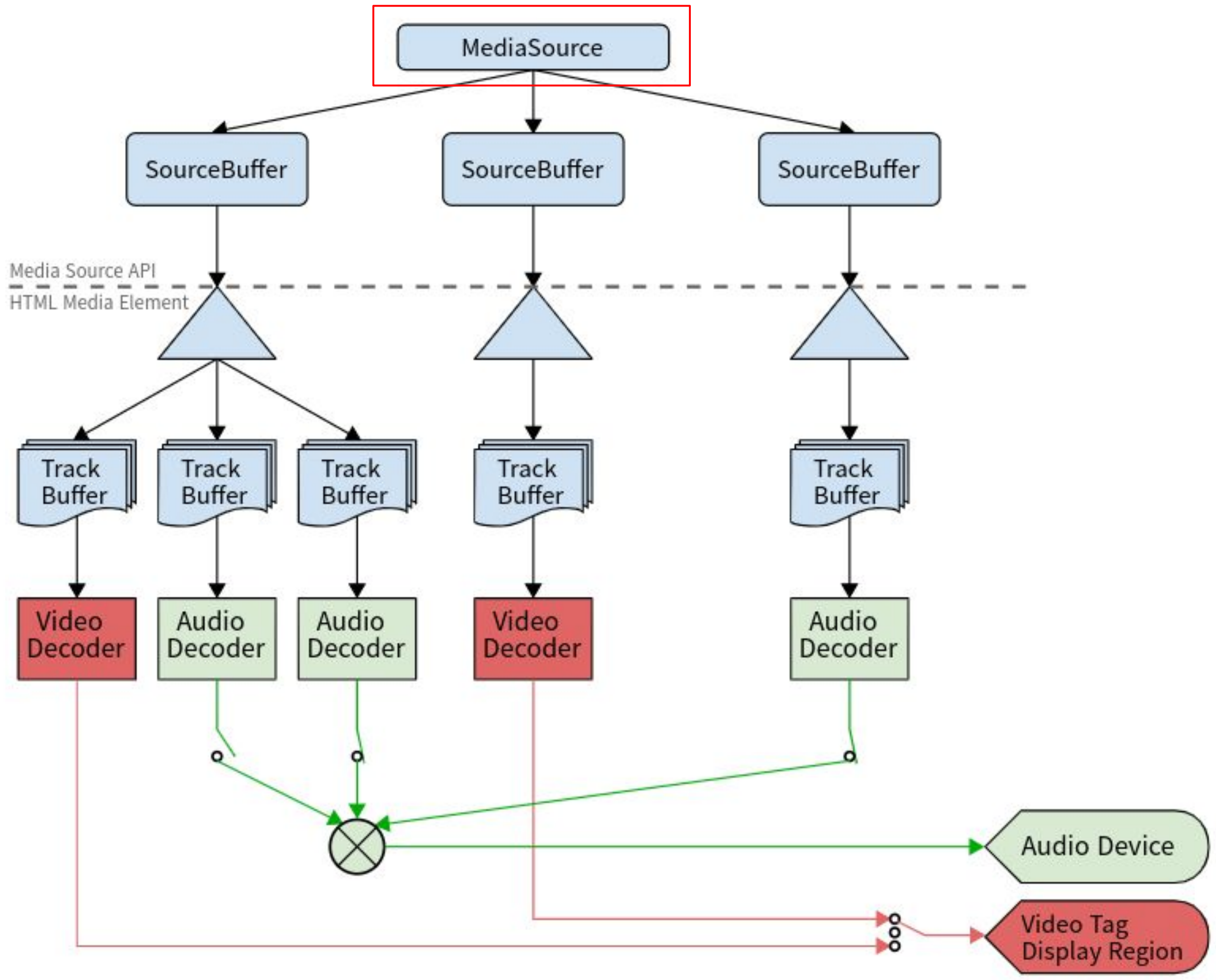
Media Playback - MSE

- Open web standard which provides functionality enabling plugin-free web-based streaming media.

- allows JavaScript to dynamically construct media streams for <audio> and <video>

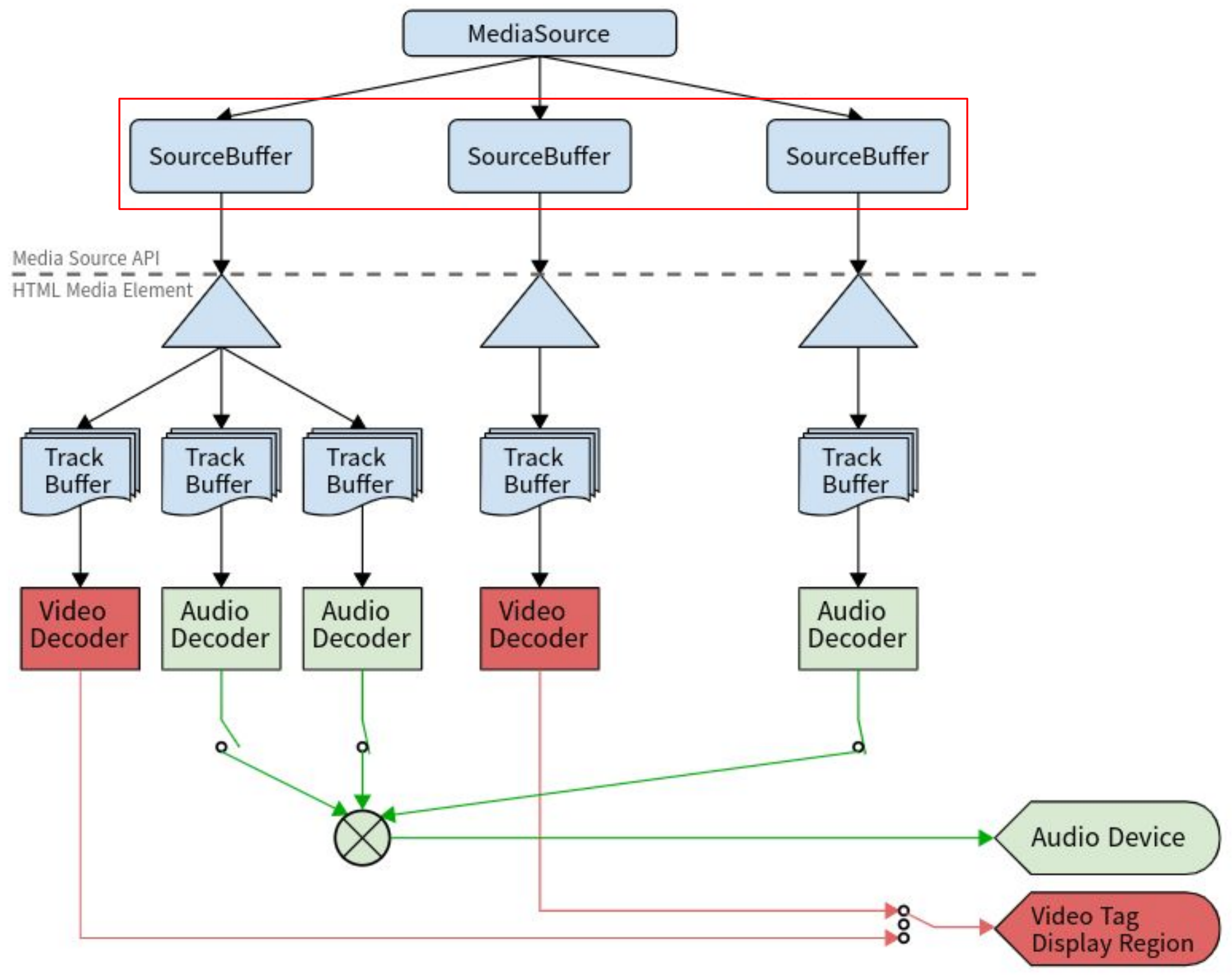
Media Playback - MSE

A MediaSource object that can serve as a source of media data for an HTMLMediaElement is defined.



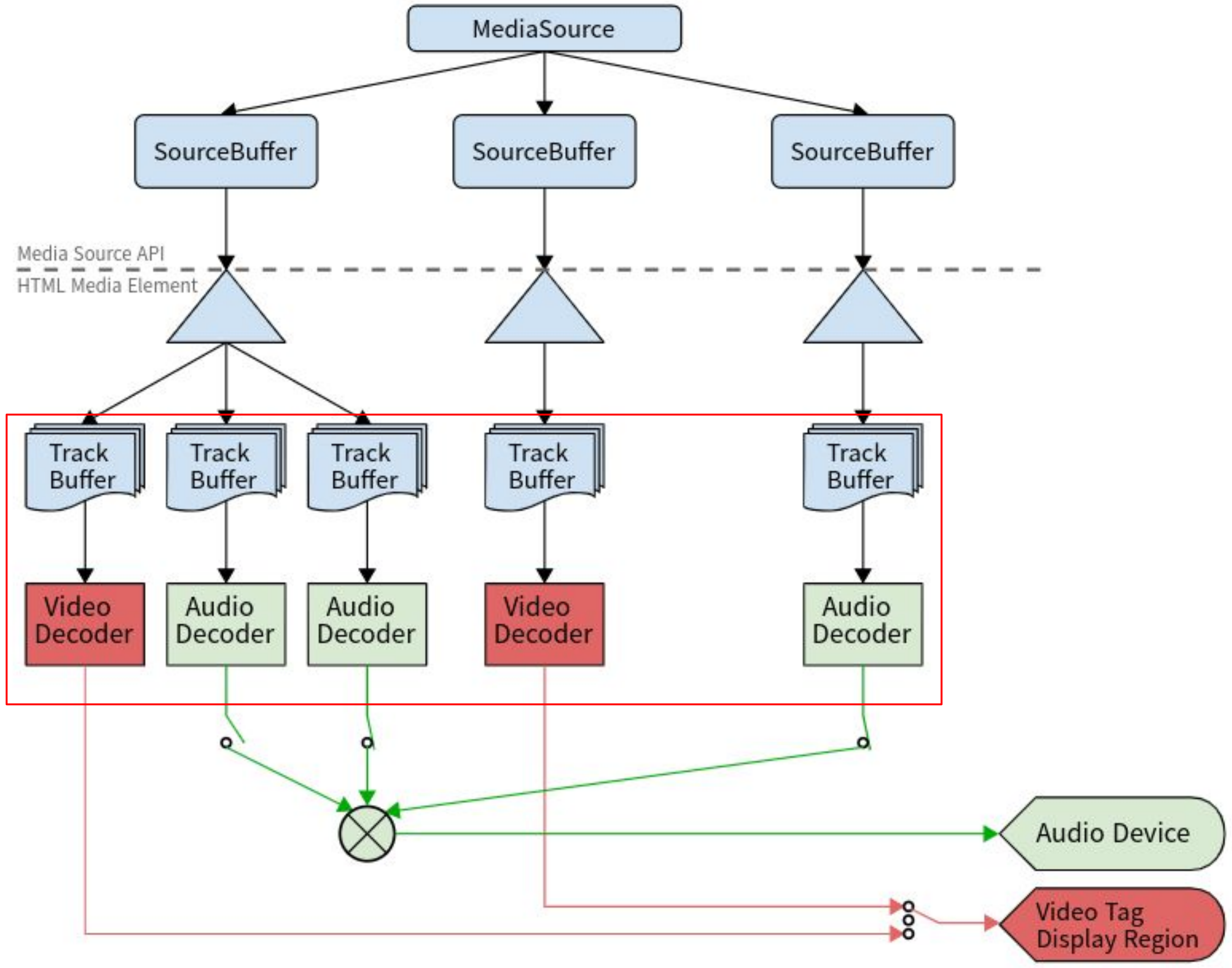
Media Playback - MSE

Applications append data segments to the SourceBuffer objects



Media Playback - MSE

Data from the SourceBuffer objects is managed as track buffers for audio, video and text data that is decoded and played.



Media Playback - DASH & HLS

- Dynamic Adaptive Streaming over HTTP (DASH) & HTTP Live Streaming (HLS) are protocols used for adaptive bitrate streaming
- specifies how adaptive content should be fetched
- Shows how to break video and audio data into a sequence of small segments, each containing a short interval of playback time

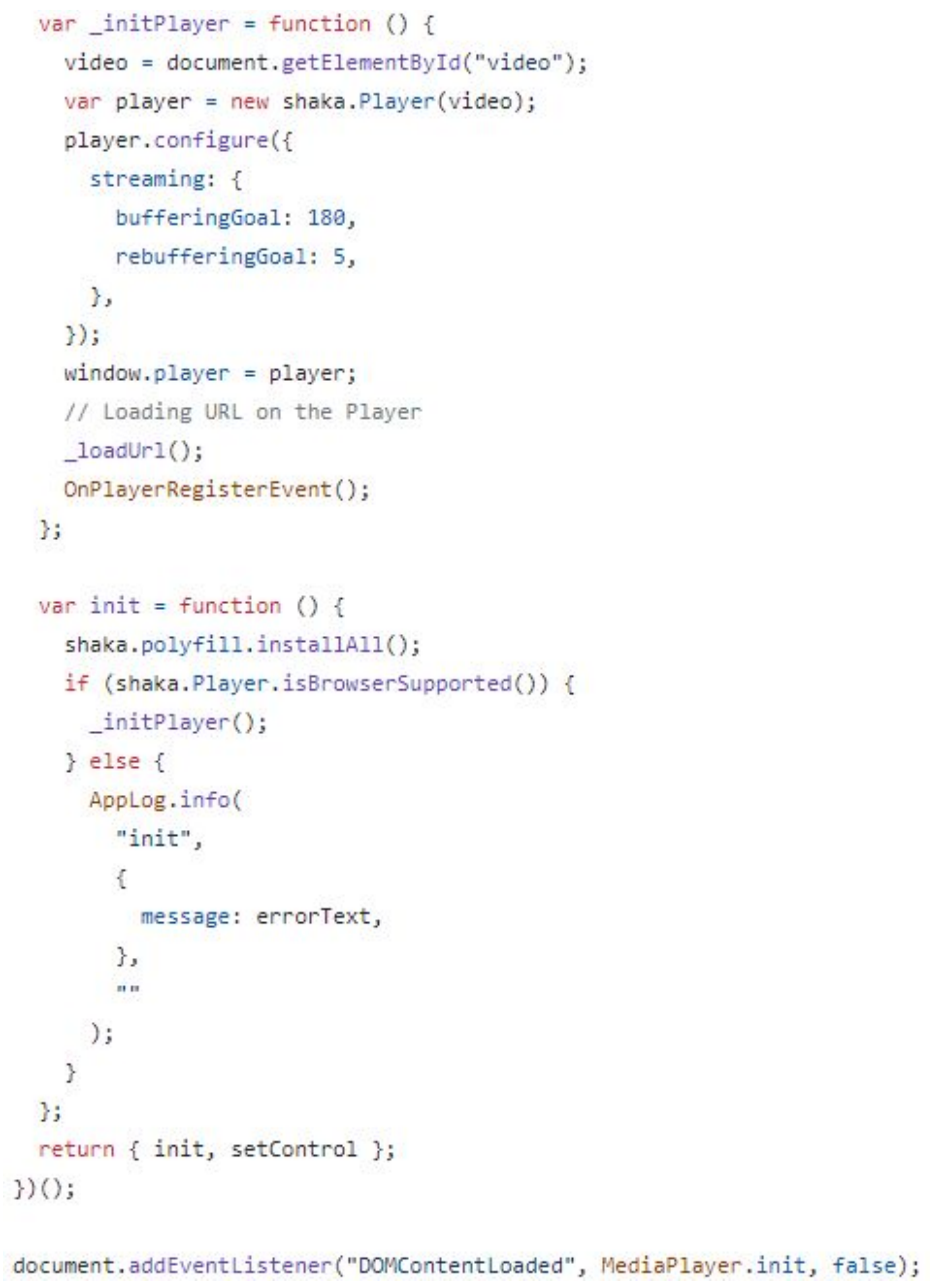
Media Playback

The js directory for Media Playback app consists of

- AppLog.js : Handles console log for debugging
- Errorcode.js : Handle error message for HTTP error
- **MetaData.js** : Handles mediaURL, map player controls ID
- **MediaPlayer.js** : Handles player control
- MediaPlayerUI.js : Handles UI control. Add event listeners
- Status.js : Update status depending on event
- Toast.js : Handle toast messages
- Util.js : utility functions for handling timers & player control ID

Media Playback

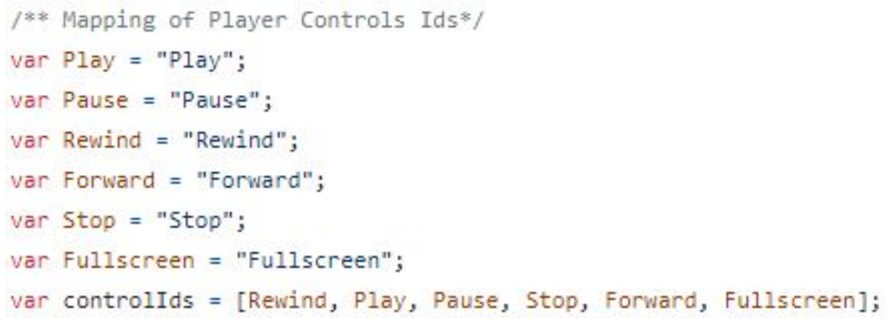
MediaPlayback/js/MediaPlayer.js



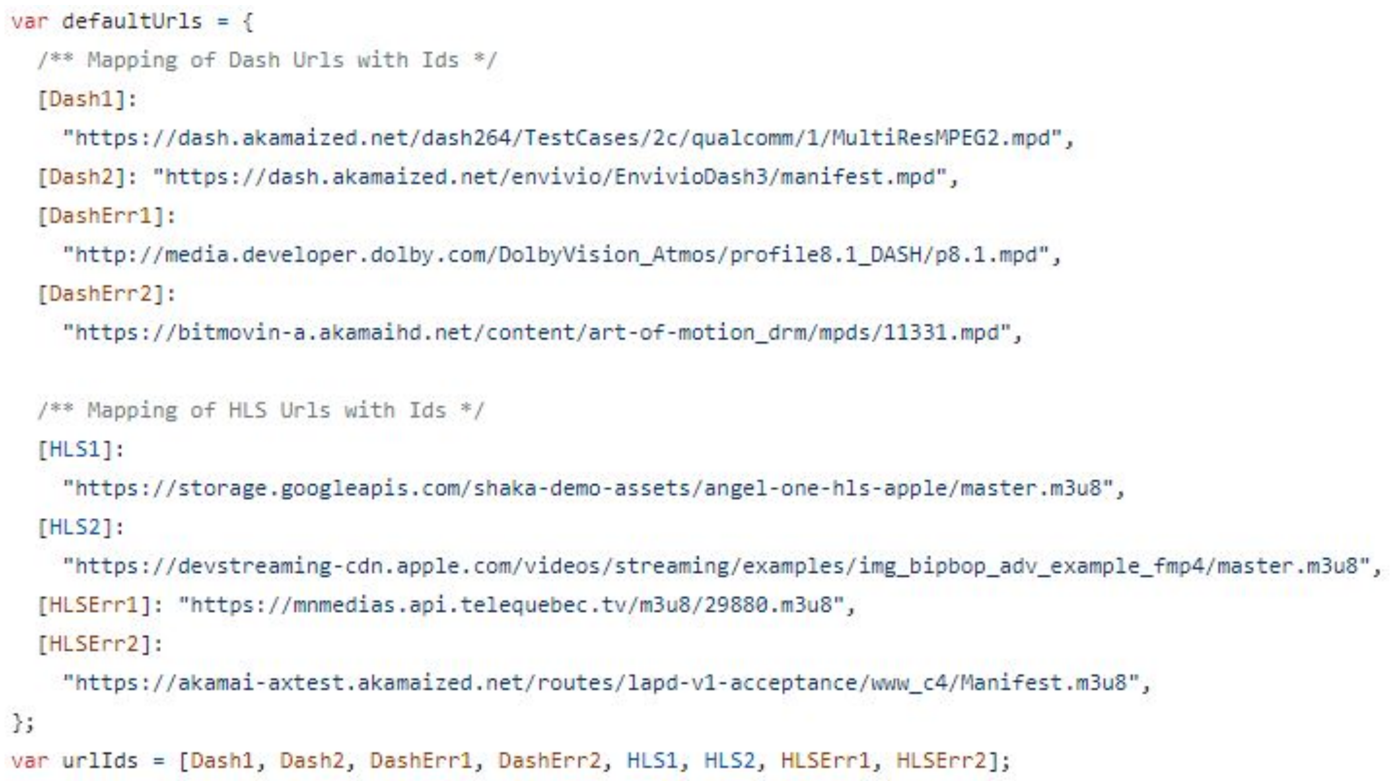
```
  var _initPlayer = function () {
    video = document.getElementById("video");
    var player = new shaka.Player(video);
    player.configure({
      streaming: {
        bufferingGoal: 180,
        rebufferingGoal: 5,
      },
    });
    window.player = player;
    // Loading URL on the Player
    _loadUrl();
    OnPlayerRegisterEvent();
  };

  var init = function () {
    shaka.polyfill.installAll();
    if (shaka.Player.isBrowserSupported()) {
      _initPlayer();
    } else {
      AppLog.info(
        "init",
        {
          message: errorText,
        },
        ""
      );
    }
  };
  return { init, setControl };
})();

document.addEventListener("DOMContentLoaded", MediaPlayer.init, false);
```

Media Playback

MediaPlayback/js/MediaData.js



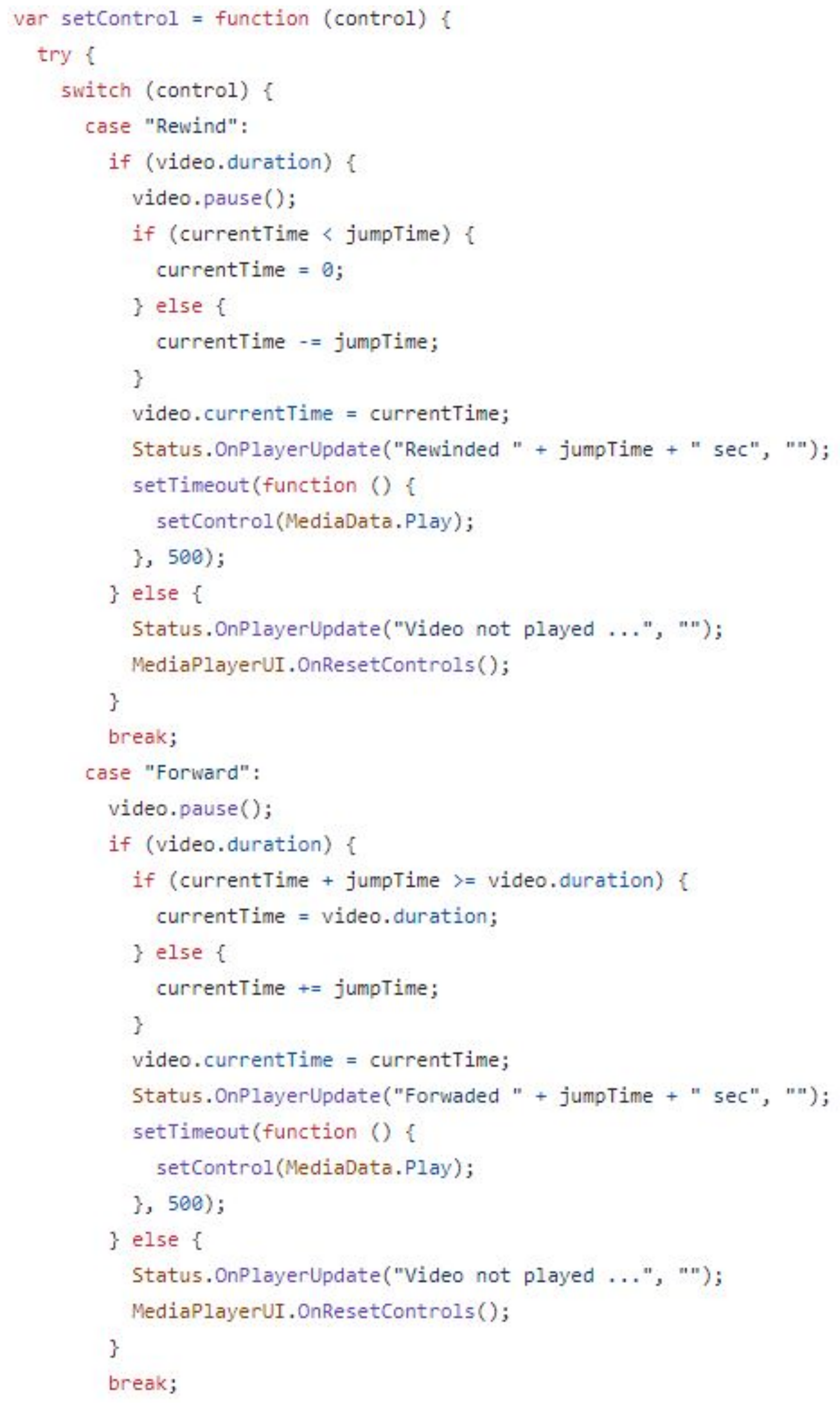
```
/** Mapping of Player Controls Ids*/
var Play = "Play";
var Pause = "Pause";
var Rewind = "Rewind";
var Forward = "Forward";
var Stop = "Stop";
var Fullscreen = "Fullscreen";
var controlIds = [Rewind, Play, Pause, Stop, Forward, Fullscreen];
```



```
var defaultUrls = {
  /** Mapping of Dash Urls with Ids */
  [Dash1]:
    "https://dash.akamaized.net/dash264/TestCases/2c/qualcomm/1/MultiResMPEG2.mpd",
  [Dash2]: "https://dash.akamaized.net/envivio/EnvivioDash3/manifest.mpd",
  [DashErr1]:
    "http://media.developer.dolby.com/DolbyVision_Atmos/profile8.1_DASH/p8.1.mpd",
  [DashErr2]:
    "https://bitmovin-a.akamaihd.net/content/art-of-motion_drm/mpds/11331.mpd",

  /** Mapping of HLS Urls with Ids */
  [HLS1]:
    "https://storage.googleapis.com/shaka-demo-assets/angel-one-hls-apple/master.m3u8",
  [HLS2]:
    "https://devstreaming-cdn.apple.com/videos/streaming/examples/img_bipbop_adv_example_fmp4/master.m3u8",
  [HLSErr1]: "https://mnmedias.api.telequebec.tv/m3u8/29880.m3u8",
  [HLSErr2]:
    "https://akamai-axtest.akamaized.net/routes/lapd-v1-acceptance/www_c4/Manifest.m3u8",
};
var urlIds = [Dash1, Dash2, DashErr1, DashErr2, HLS1, HLS2, HLSErr1, HLSErr2];
```

Media Playback

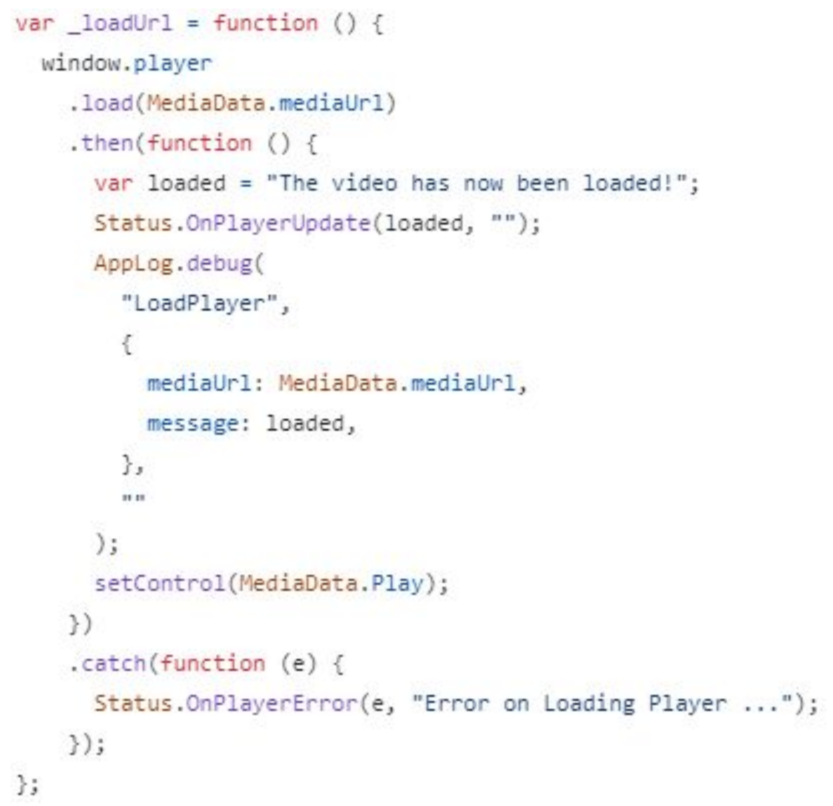
MediaPlayback/js/MediaPlayer.js



```
var setControl = function (control) {
  try {
    switch (control) {
      case "Rewind":
        if (video.duration) {
          video.pause();
          if (currentTime < jumpTime) {
            currentTime = 0;
          } else {
            currentTime -= jumpTime;
          }
          video.currentTime = currentTime;
          Status.OnPlayerUpdate("Rewinded " + jumpTime + " sec", "");
          setTimeout(function () {
            setControl(MediaData.Play);
          }, 500);
        } else {
          Status.OnPlayerUpdate("Video not played ...", "");
          MediaPlayerUI.OnResetControls();
        }
        break;
      case "Forward":
        video.pause();
        if (video.duration) {
          if (currentTime + jumpTime >= video.duration) {
            currentTime = video.duration;
          } else {
            currentTime += jumpTime;
          }
          video.currentTime = currentTime;
          Status.OnPlayerUpdate("Forwaded " + jumpTime + " sec", "");
          setTimeout(function () {
            setControl(MediaData.Play);
          }, 500);
        } else {
          Status.OnPlayerUpdate("Video not played ...", "");
          MediaPlayerUI.OnResetControls();
        }
        break;
```

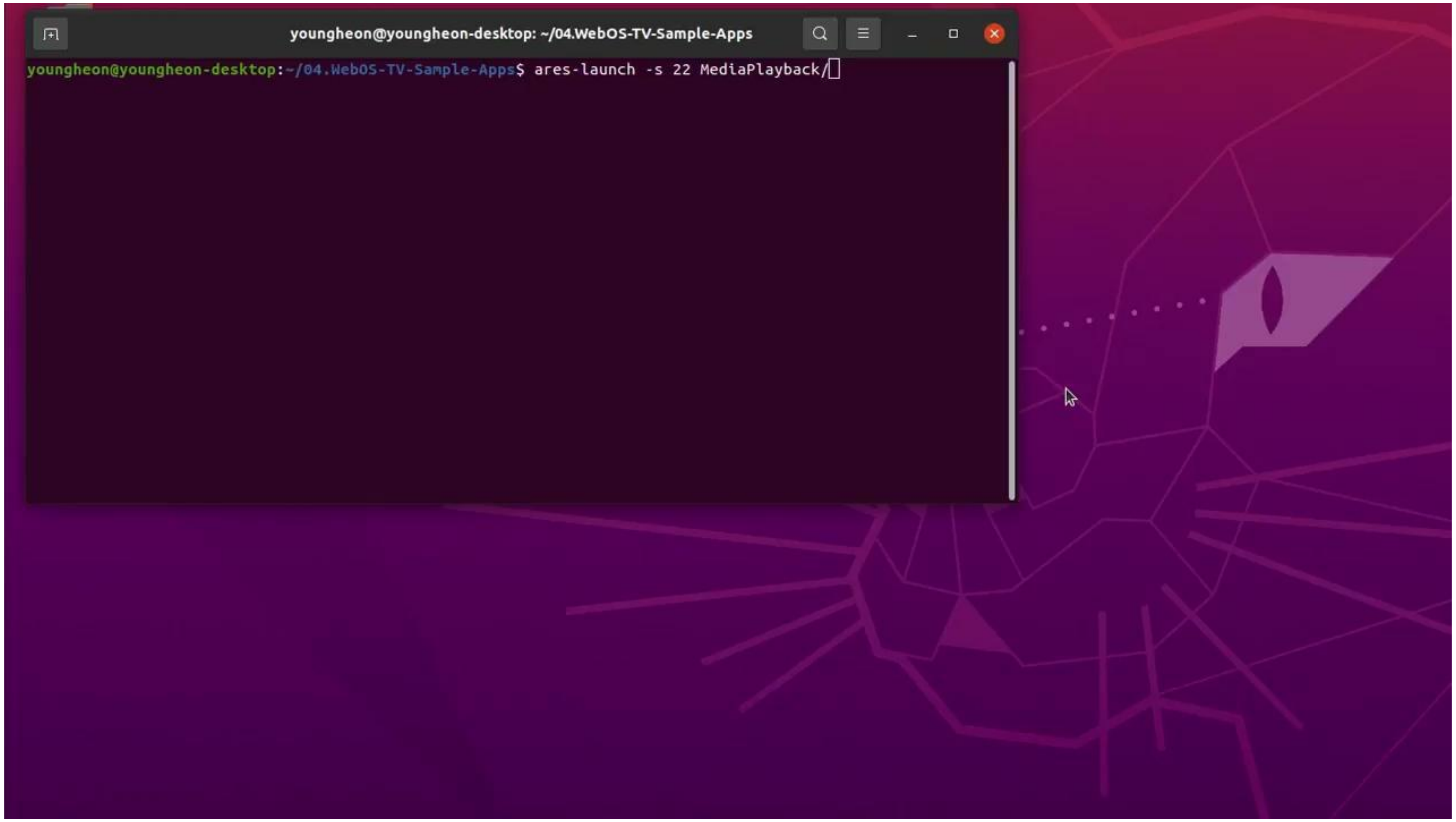
Media Playback

MediaPlayback/js/MediaPlayer.js



```
var _loadUrl = function () {
  window.player
    .load(MediaData.mediaUrl)
    .then(function () {
      var loaded = "The video has now been loaded!";
      Status.OnPlayerUpdate(loaded, "");
      AppLog.debug(
        "LoadPlayer",
        {
          mediaUrl: MediaData.mediaUrl,
          message: loaded,
        },
        ""
      );
      setControl(MediaData.Play);
    })
    .catch(function (e) {
      Status.OnPlayerError(e, "Error on Loading Player ...");
    });
};
```

Media Playback



```
youngheon@youngheon-desktop:~/04.WebOS-TV-Sample-Apps$ ares-launch -s 22 MediaPlayback/
```