Gstreamer In RDK

Deepthi Suseelan, Sudeep Rayaroth
RDK Support

Agenda

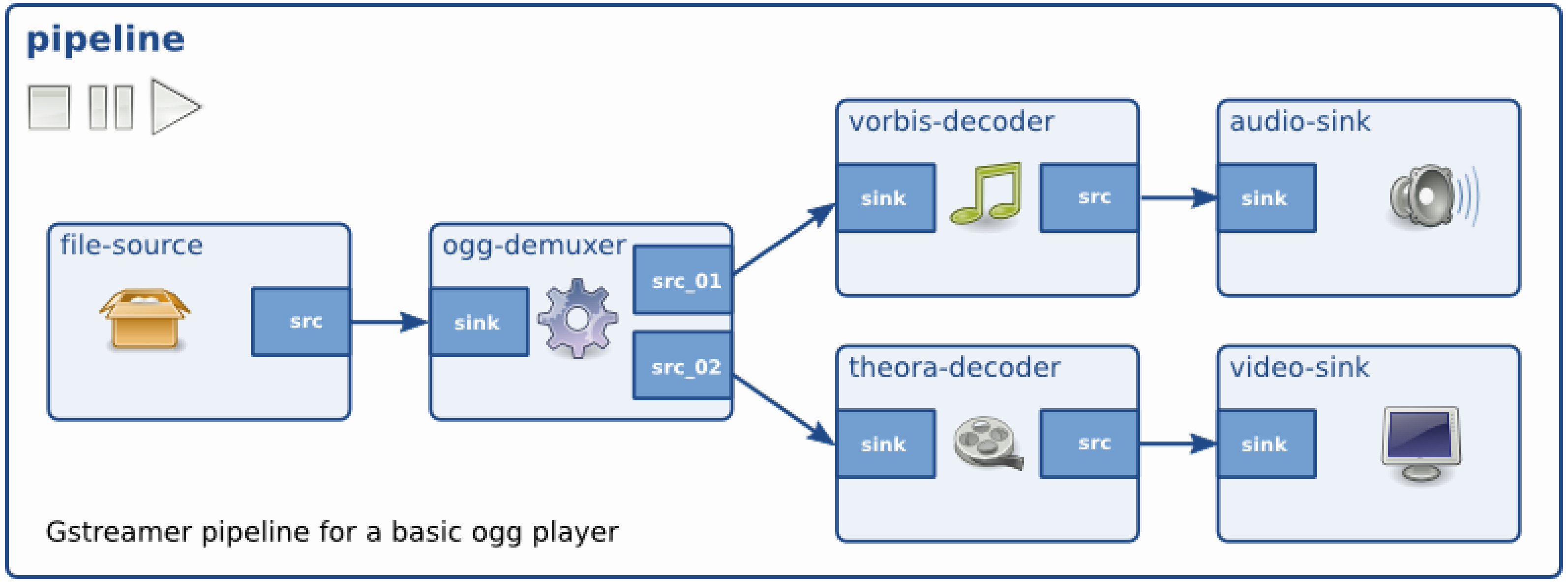
- Gstreamer Basics
- Gstreamer in RDK
  - In RMF (MediaFramework)
  - In WebKit
  - In AAMP (IP Player)

What is Gstreamer

- Multimedia framework used to write media applications
- Open Source - Based on GObject
- Provides access to Hardware
- Plugin architecture, pipeline architecture
- Scriptable Command Line Tools

Gstreamer Pipeline

Gstreamer organizes collection of elements which work together into directed graphs called **pipelines**



Gstreamer Tools: gst-inspect-1.0

- Prints information about the plugin/element
- Uses information from Gstreamer registry

```
root@raspberrypi-rdk-hybrid-thunder:/usr/lib/gstreamer-1.0# gst-inspect-1.0 filesrc
Factory Details:
  Rank                     primary (256)
  Long-name                File Source
  Klass                    Source/File
  Description              Read from arbitrary point in a file
  Author                   Erik Walthinsen <omega@cse.ogi.edu>

Plugin Details:
  Name                     coreelements
  Description              GStreamer core elements
  Filename                 /usr/lib/gstreamer-1.0/libgstcoreelements.so
  Version                  1.10.4
  License                  LGPL
  Source module            gstreamer
  Source release date      2017-02-23
  Binary package           GStreamer source release
  Origin URL               Unknown package origin

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBaseSrc
                         +----GstFileSrc

Implemented Interfaces:
  GstURIHandler

Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      ANY
```

```
Element Implementation:
  Has change_state() function: gst_base_src_change_state

Element has no clocking capabilities.

URI handling capabilities:
  Element can act as source.
  Supported URI protocols:
    file

Pads:
  SRC: 'src'
    Pad Template: 'src'

Element Properties:
  name                : The name of the object
                        flags: readable, writable
                        String. Default: "filesrc0"
  parent              : The parent of the object
                        flags: readable, writable
                        Object of type "GstObject"
  blocksize           : Size in bytes to read per buffer (-1 = default)
                        flags: readable, writable
                        Unsigned Integer. Range: 0 - 4294967295 Default: 4096
  num-buffers         : Number of buffers to output before sending EOS (-1 = unlimited)
                        flags: readable, writable
                        Integer. Range: -1 - 2147483647 Default: -1
  typefind            : Run typefind before negotiating
                        flags: readable, writable
                        Boolean. Default: false
  do-timestamp        : Apply current stream time to buffers
                        flags: readable, writable
                        Boolean. Default: false
  location            : Location of the file to read
                        flags: readable, writable, changeable only in NULL or READY state
                        String. Default: null
root@raspberrypi-rdk-hybrid-thunder:/usr/lib/gstreamer-1.0#
```

RDK PREFERRED PROGRAM

Gstreamer Tools: gst-discoverer-1.0

- Tool that can be used to print basic metadata and stream information about a media file.

gst-discoverer-1.0 /opt/www/sample1.ts

```
Done discovering file:///opt/www/sample1.ts

Topology:
  container: MPEG-2 Transport Stream
    subtitles: DVB subtitles
    audio: MPEG-1 Layer 2 (MP2)
    video: H.264

Properties:
  Duration: 0:02:00.011873148
  Seekable: yes
  Tags:
      language code: en
      audio codec: MPEG-1 Audio
      has crc: false
      channel mode: stereo
      nominal bitrate: 192000
      video codec: H.264
```

Gstreamer Tools: gst-launch-1.0

- gst-launch-1.0 filesrc location=/opt/www/sample1.ts ! tsdemux ! h264parse ! omxh264dec ! glimagesink
- gst-launch-1.0 filesrc location=/opt/www/sample1.ts ! tsdemux ! mpegaudioparse ! mpg123audiodec ! omxhdmiaudiosink
- gst-launch-1.0 **-v** filesrc location=/opt/www/sample1.ts ! tsdemux name=dmx ! queue ! h264parse ! omxh264dec ! glimagesink dmx. ! queue ! mpegaudioparse ! mpg123audiodec ! omxhdmiaudiosink

Playbin

- For easy playback
- Playbin will use plugin ranks and choose elements automatically

- gst-launch-1.0 playbin uri=file:///opt/www/sample1.ts
- gst-launch-1.0 playbin uri=http://localhost:50050/received_spts1.ts video-sink=westerossink
- gst-launch-1.0 -vm playbin uri=file:///opt/www/sample1.ts

Additional Debug

export GST_DEBUG=0,filesrc:6

export GST_DEBUG=*:3,*dec:6

gst-launch-1.0 --gst-debug-help

```
dvrsink                 0   dvrsink element
dvrsrc                  0   dvrsrc element
dynudpsink              0   UDP sink
ebmlread                0   EBML stream helper class
ebmlwrite               0   Write EBML structured data
encodebin               0   encoder bin
equalizer               0   equalizer
errorignore             0   Convert some GstFlowReturn types into others
exclusion               0   Template exclusion
faad                    0   AAC decoding
fakesink                0   fakesink element
fakesrc                 0   fakesrc element
fdsink                  0   fdsink element
fdsrc                   0   fdsrc element
festival                0   Festival text-to-speech synthesizer
fieldanalysis           0   Video field analysis
filesink                0   filesink element
filesrc                 6 LOG       filesrc element
fisheye                 0   fisheye
flacdec                 0   flac decoder
flacenc                 0   Flac encoding element
flacparse               0   Flac parser element
flactag                 0   flac tag rewriter
```

```
| # | Name     | Description                                                     |
|---|----------|-----------------------------------------------------------------|
| 0 | none     | No debug information is output.                                 |
| 1 | ERROR    | Logs all fatal errors. These are errors that do not allow the   |
|   |          | core or elements to perform the requested action. The           |
|   |          | application can still recover if programmed to handle the       |
|   |          | conditions that triggered the error.                            |
| 2 | WARNING  | Logs all warnings. Typically these are non-fatal, but           |
|   |          | user-visible problems are expected to happen.                   |
| 3 | FIXME    | Logs all "fixme" messages. Those typically that a codepath that|
|   |          | is known to be incomplete has been triggered. It may work in    |
|   |          | most cases, but may cause problems in specific instances.       |
| 4 | INFO     | Logs all informational messages. These are typically used for   |
|   |          | events in the system that only happen once, or are important    |
|   |          | and rare enough to be logged at this level.                     |
| 5 | DEBUG    | Logs all debug messages. These are general debug messages for   |
|   |          | events that happen only a limited number of times during an     |
|   |          | object's lifetime; these include setup, teardown, change of     |
|   |          | parameters, etc.                                                |
| 6 | LOG      | Logs all log messages. These are messages for events that       |
|   |          | happen repeatedly during an object's lifetime; these include    |
|   |          | streaming and steady-state conditions. This is used for log     |
|   |          | messages that happen on every buffer in an element for example.|
| 7 | TRACE    | Logs all trace messages. Those are message that happen very     |
|   |          | very often. This is for example is each time the reference      |
|   |          | count of a GstMiniObject, such as a GstBuffer or GstEvent, is   |
|   |          | modified.                                                       |
| 9 | MEMDUMP  | Logs all memory dump messages. This is the heaviest logging and|
|   |          | may include dumping the content of blocks of memory.            |
+------------------------------------------------------------------------------+
```

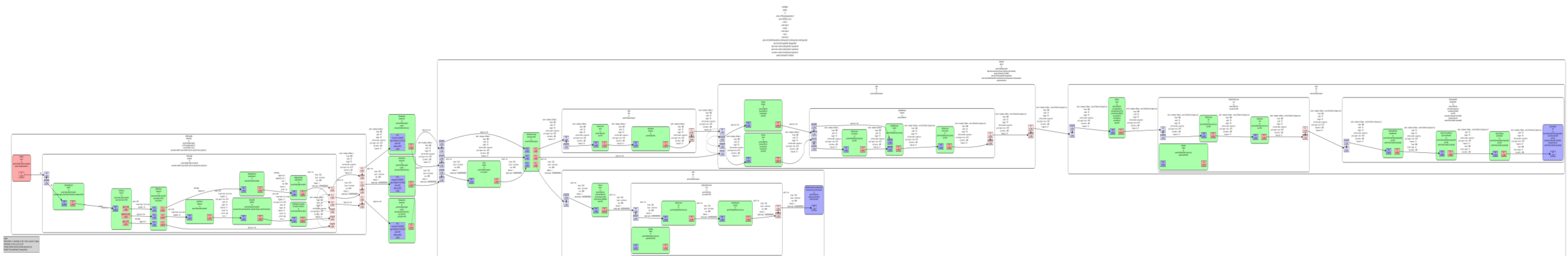
-development/index.html

RDK PREFERRED PROGRAM

Pipeline Graphs

For visual representation of pipeline
Handy when using all-in-one elements like playbin or decodebin

- export GST_DEBUG_DUMP_DOT_DIR=<directory_name>
- Run the gst-launch or play command. It will create dot files for each stage of the pipeline.
- Convert the dot file to an image (using dot utility or tools like graphviz):
  - dot -Tpng -oimage.png 0.00.01.055760521-gst-launch.PAUSED_PLAYING.dot



Writing applications

- gst_parse_launch:

  Can use the same gst-launch command argument with an application wrapped around it
  Can handle events and add other functionalities

  ```
  pipeline = gst_parse_launch ("playbin uri=file:///opt/www/sample1.ts", NULL);
  ```

- gst_element_factory_make:

  To create elements from an application.

  ```
  filesrc = gst_element_factory_make ("filesrc", "my_filesource");
  ```

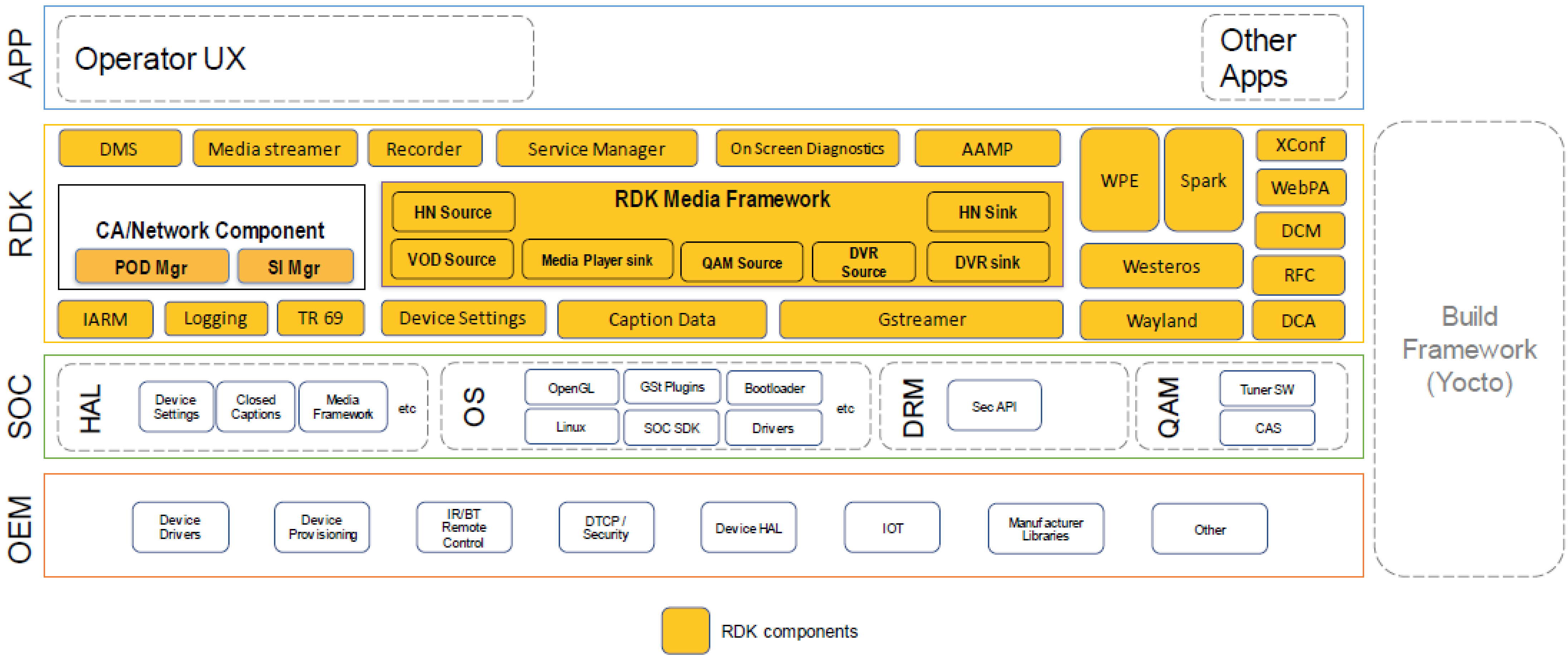
Gstreamer In RDK

- RDK uses gstreamer framework for various types of media playback
- The main modules which uses Gstreamer are:
  - RMF (RDK Media Framework)
  - AAMP (Advanced Adaptive Media Player)
  - Webkit

RDK Media Framework - RMF

- RMF in RDK provides the core functionality of audio/video playback
- Loosely based on Gstreamer
- Defines generic source, sink and filter; known as RMF elements
- Hardware interface is through Gstreamer plugins
- Constructing an RMF media pipeline by connecting various RMF elements results in the formation of a gstreamer pipeline made up of the gstreamer elements used to implement each RMF element.

RDK Components

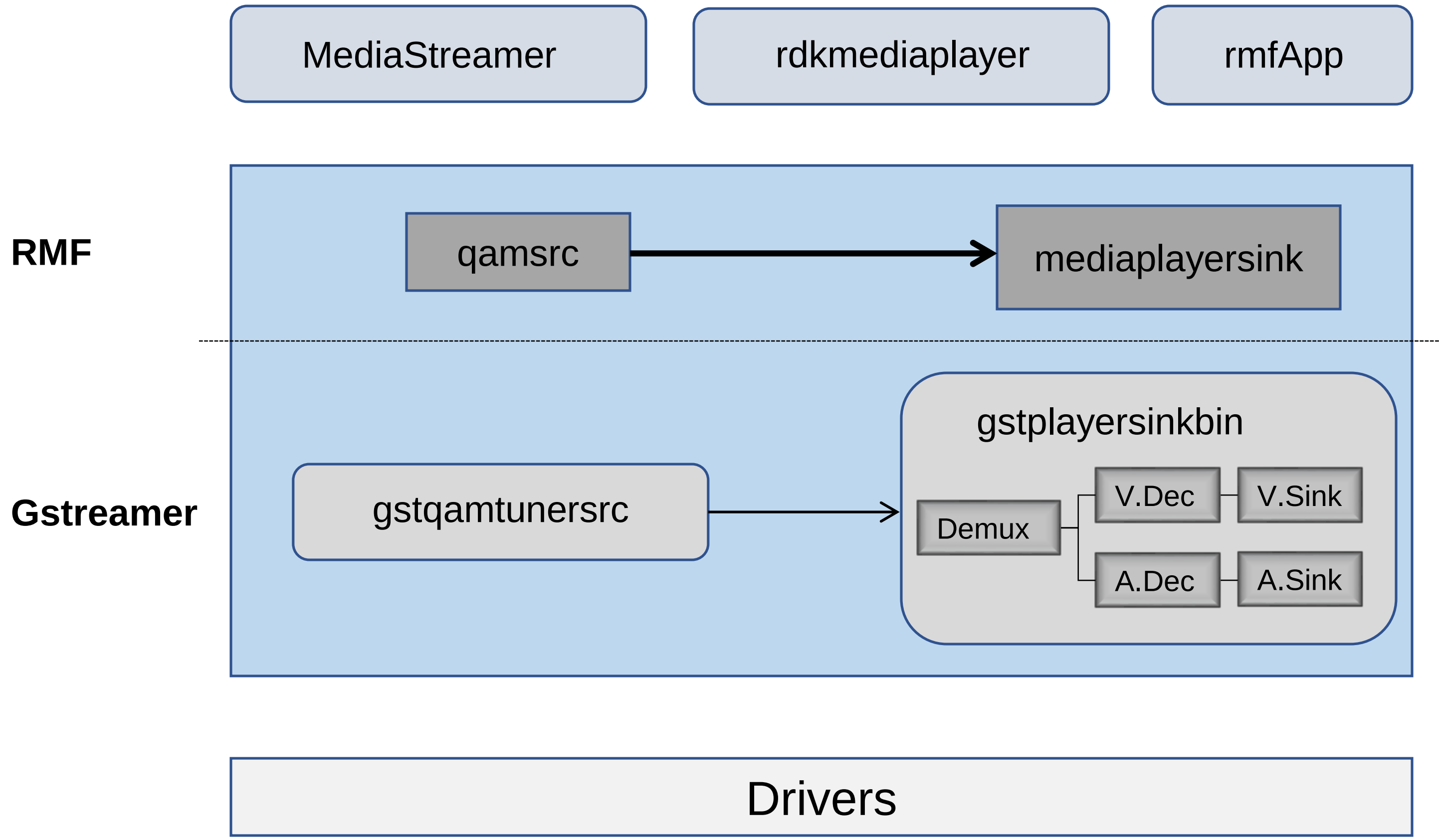


RDK PREFERRED PROGRAM

RMF and Gstreamer

Category	RMF	Gstreamer
Source	QAMSource	qamtunersrc
	DVRSource	dvrsrc
	HNSource	httpsrc
Sink	HNSink	httpsink
	DVRSink	dvrsink
	MediaPlayerSink	playersinkbin
Filter	RBIFilter	rbifilter

Components Interaction



rmfApp

In a simulated live playback, the following rmfApp command maybe used to playback the local SPTS video file:

launch -source qamsource -sink mediaplayersink ocap://0x125d

where
source : qamsource
sink: mediaplayersink
url: ocap://0x125d

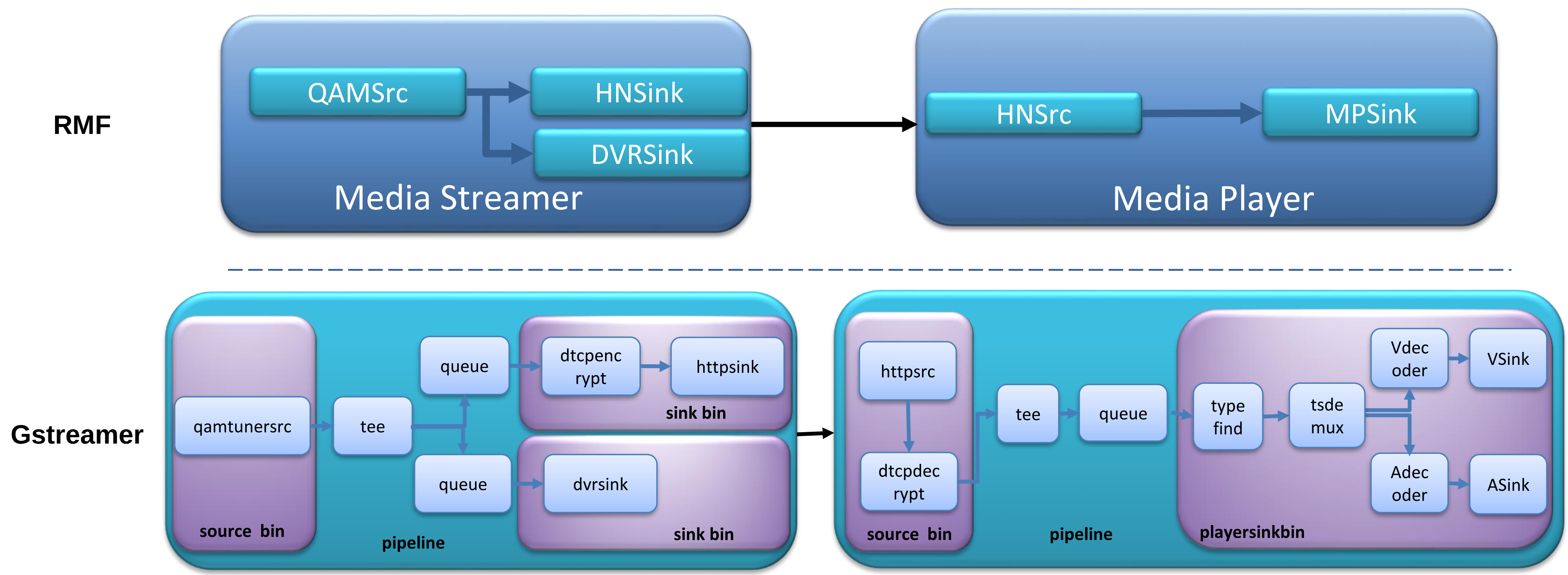
$ rmfApp->launch –source qamsource –sink mediaplayersink ocap://0x125d

rmfApp: Create pipeline

```
RMFMediaSourceBase* pipeline::createSource()
{
#ifdef USE_HNSRC
    if( c_source == HN_SOURCE)
    {
        return new HNSource();
    }
#endif
#ifdef USE_DVR
    if( c_source == DVR_SOURCE)
    {
        return new DVRSource();
    }
#endif
#ifdef USE_QAMSRC
    if( c_source == QAM_SOURCE)
    {
        pipeline::initQAM();
        return new RMFQAMSrc();
    }
#endif
#ifdef USE_VODSRC
    if( c_source == VOD_SOURCE)
    {
        pipeline::initQAM();
        return new RMFVODSrc();
    }
#endif
    {
        return 0;
    }
}
```

```
RMFMediaSinkBase* pipeline::createSink()
{
#ifdef USE_MEDIAPLAYERSINK
    if( c_sink == MEDIAPLAYER_SINK)
    {
        MediaPlayerSink* pSink = new MediaPlayerSink();
        pSink->init();
        pSink->setVideoRectangle(0, 0, 1280, 720);
        /* RDKSEC-811 Coverity fix checked return */
        if ( pSink->setSource(c_pSource) != RMF_RESULT_SUCCESS)
        {
            /* Unhandled error */
            g_print("Error: setSource failed\n");
        }
        c_pMediaPlayerSink = pSink;
        return pSink;
    }
#endif
#ifdef USE_DVR
    if ( c_sink == DVR_SINK)
    {
        g_print("createSink: recordingId=%s\n", c_recordingId.c_str() );
        DVRSink* pSink = new DVRSink(c_recordingId);
        pSink->init();
        pSink->setSource(c_pSource);
        return pSink;
    }
#endif
    return 0;
}
```

RMF Elements – Live/TSB Streaming



HNSource bin: Sample code

```
RMFResult HNSourcePrivate::populateBin(GstElement* bin)
{
    int blocksize = 128*1024;
    char *pHttpSrc = NULL;
    GstElement* source = NULL;

    source = gst_element_factory_make("httpsrc", "curl");

    gst_bin_add(GST_BIN(bin), source);
    g_object_set(G_OBJECT (source), "blocksize", blocksize,  NULL);

    GstElement* last_el = source;
    bool isDtcpIpEnabled = m_url.find("DTCP1HOST") != std::string::npos;
    if (isDtcpIpEnabled)
    {
        static int http_len = std::string("http://").length();
        int ip_start_pos = m_url.find("http://") + http_len;
        int ip_end_pos = m_url.find(":",ip_start_pos);
        std::string ip_addr = m_url.substr(ip_start_pos, ip_end_pos-ip_start_pos);

        GstElement* dtcp = gst_element_factory_make("dtcpdec", "dtcpdecrypt");

        m_dtcpServerIp = ip_addr;
        g_object_set(dtcp, "dtcp_src_ip", ip_addr.c_str(), NULL);
        g_object_set(dtcp, "dtcp_port", DTCP_SERVER_PORT, NULL);
        g_object_set(G_OBJECT (dtcp), "buffersize", blocksize,  NULL);

        gst_bin_add(GST_BIN(bin), dtcp);

        // Link the filter to the soup source.
        if (!gst_element_link(source, dtcp))
        {
            HNSRCLOG_ERROR("HNSource: Failed to link source and filter\n");
            return RMF_RESULT_INTERNAL_ERROR;
        }
```

Gst-plugins

gst-plugins-rdk:

- httpsrc
- httpsink
- dtcpenc
- dtcpdec
- rbifilter
- mediaplayersink_generic

gst-plugins-rdk-dvr:

- dvrsrc
- dvrsink
- aesencrypt
- aesdecrypt

gst-plugins-soc:

- demux
- videodecoder
- videosink
- audiodecoder
- audiosink
- **playersinkbin**

gst-plugins-rdk/soc

- qamtunersrc

Debugging Issue: Audio loss

Issue : No audio for a particular channel

Step 1 : gst-launch-1.0 filesrc location=/opt/www/sample1.ts ! playersinkbin   (as used by RMF)
Result : no audio

Step 2 : gst-launch-1.0 playbin uri=file:///opt/www/sample1.ts
Result : giving audio
Inference : Issue with playersinkbin elements

Step 3 : Add verbose to gst-launch with playersinkbin,

```
mp3 playback
0:00:00.308841537  5989   0x5f8cf0 ERROR        playersinkbin gstplayersinkbin.c:721:plug_pad:<playersinkbin0> Audio decoder is failed..........
(gst-launch-1.0:5989): GStreamer-CRITICAL **: gst_bin_add: assertion 'GST_IS_ELEMENT (element)' failed
(gst-launch-1.0:5989): GStreamer-CRITICAL **: gst_element_link_pads_full: assertion 'GST_IS_ELEMENT (dest)' failed
0:00:00.322474713  5989   0x5f8cf0 ERROR        playersinkbin gstplayersinkbin.c:753:plug_pad:<playersinkbin0> Failed to link m_audio_parser to m_adec
```

Found that for mp3 playback, "mpegaudioparse" and "avdec_mp3" are used from playersinkbin. Inspect them:

```
gst-inspect-1.0 avdec_mp3
No such element or plugin 'avdec_mp3'
```

Step 4 : Generate pipeline graph for gst-launch with playbin

Solution : Either include gstlibav plugins or to choose the plugins used by playbin

Webkit overview

What is Webkit ?

01 Web content engine

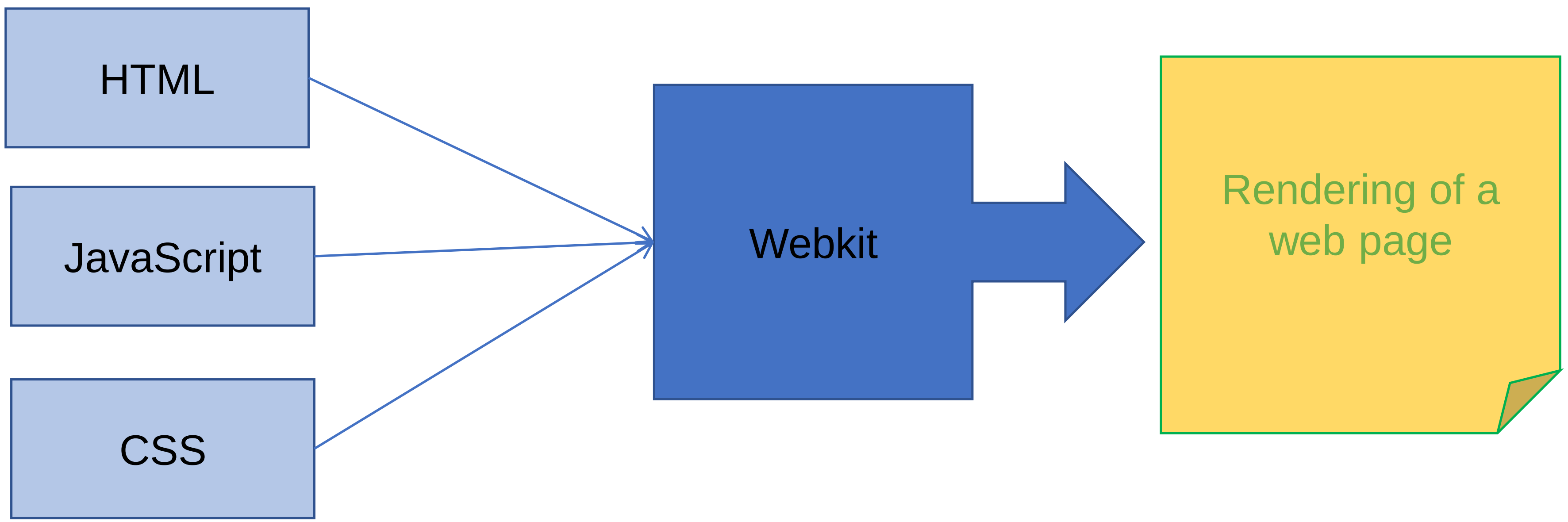
02 Processes and renders web content

03 Started as a fork of KHTML and KJS

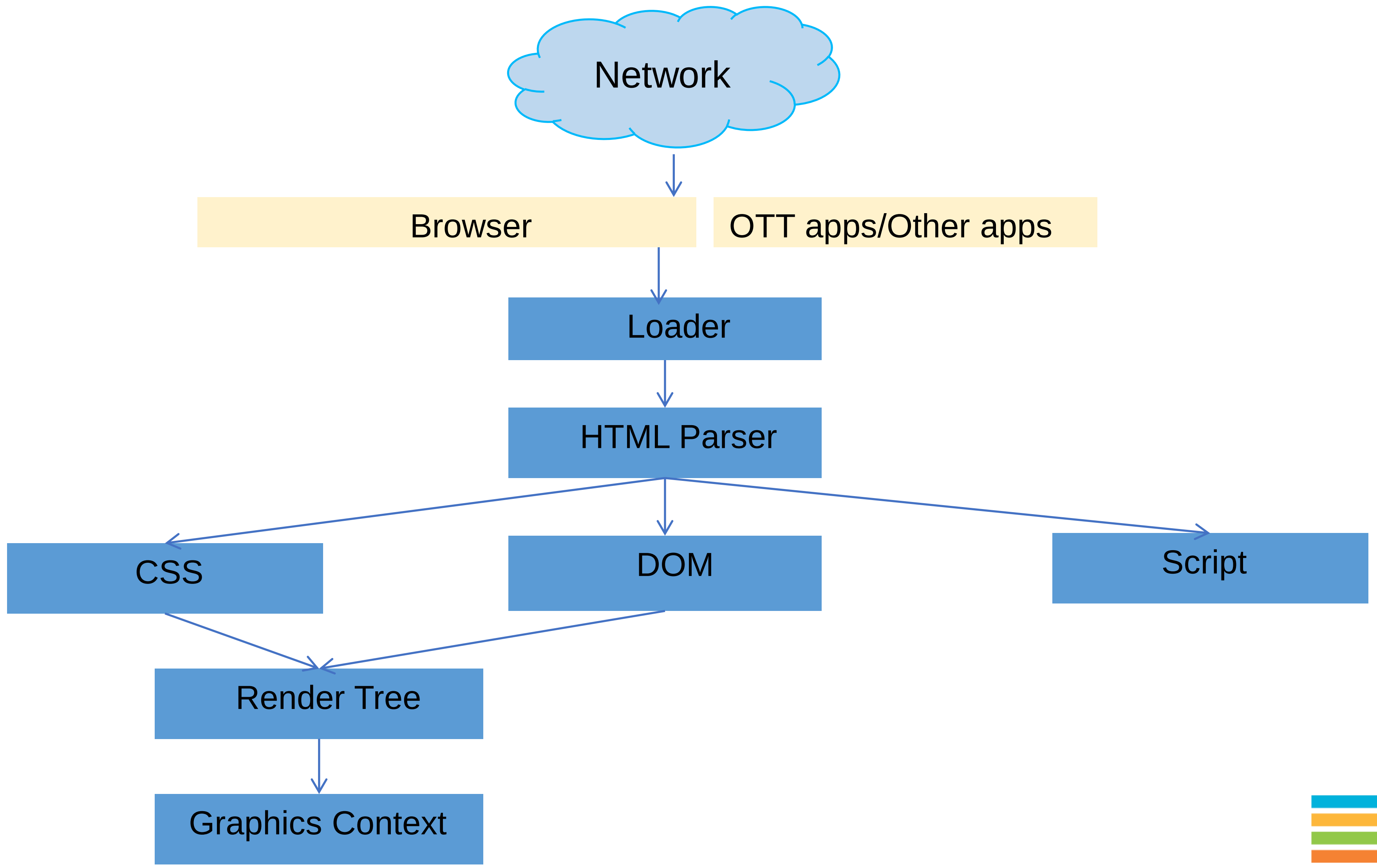
04 Open source since 2005

05 Goals: Open source, compatibility, compliance, stability, performance, security, portability, usability, hackability

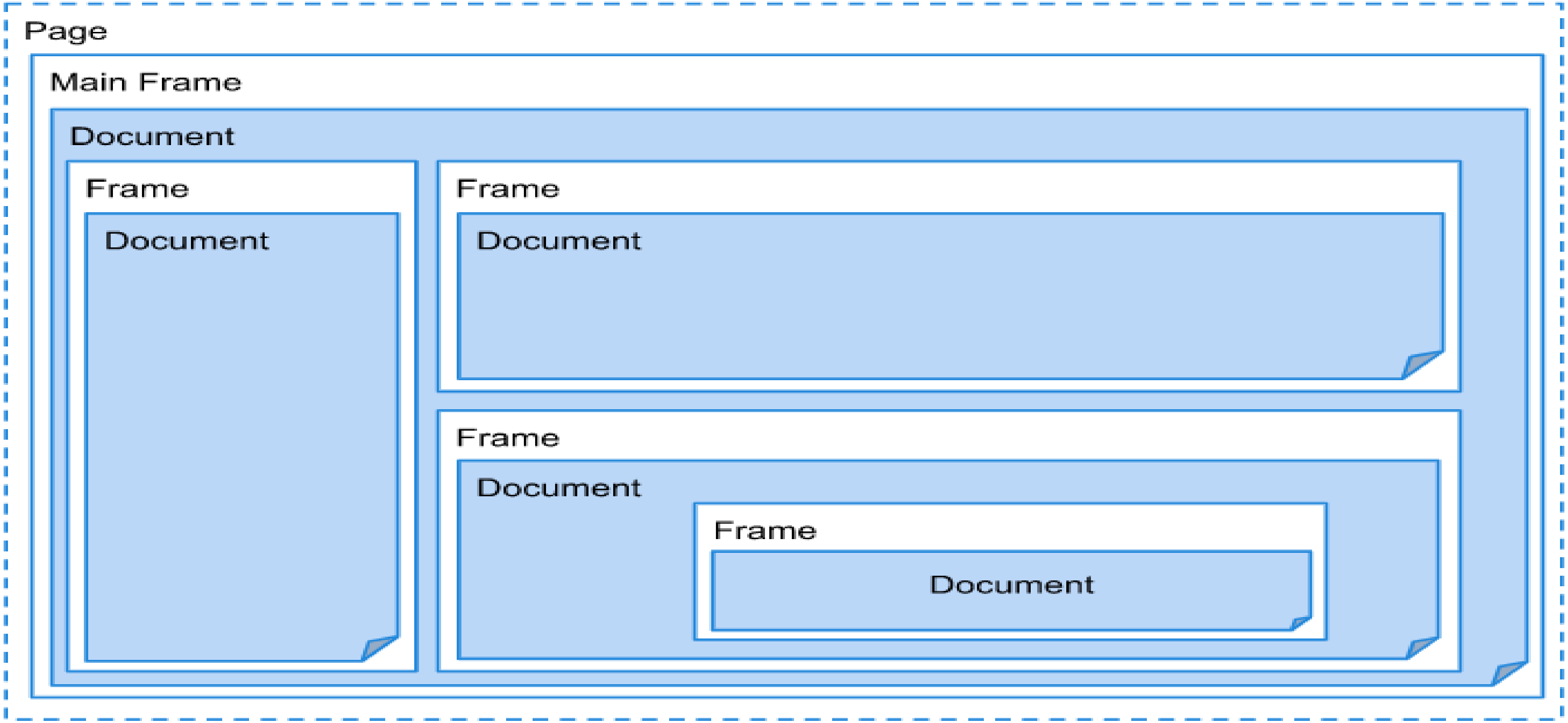
What does webkit do?



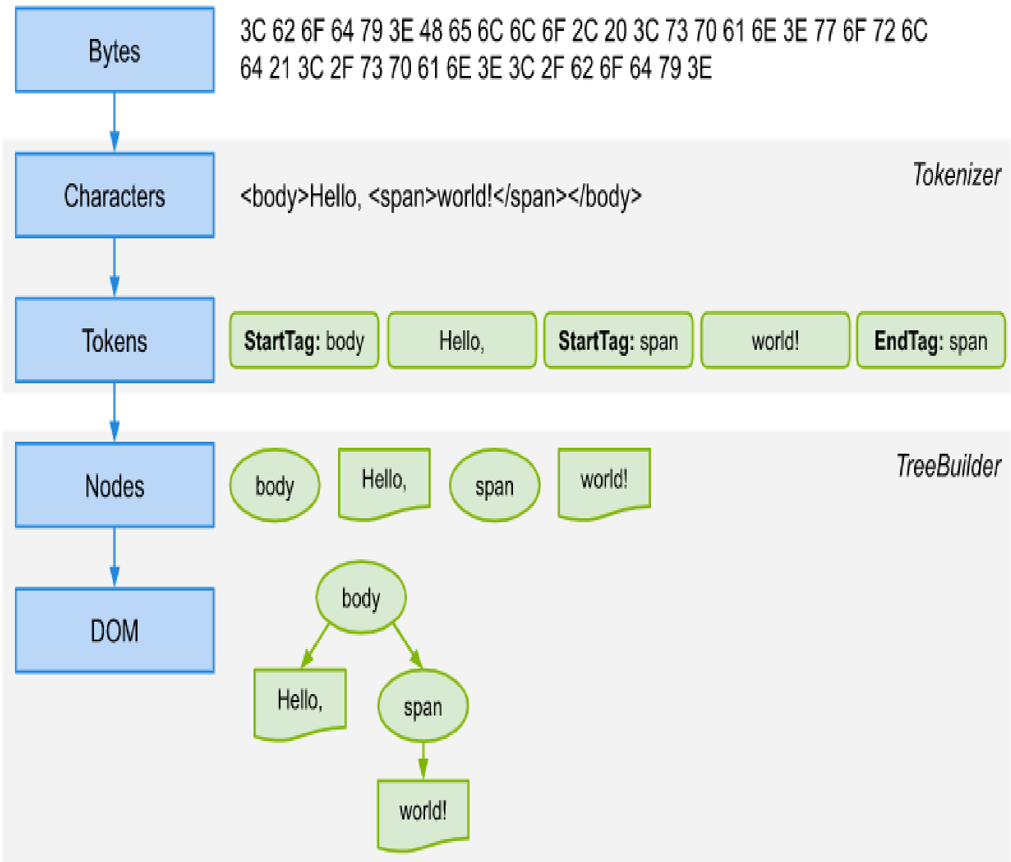
Life of a webpage



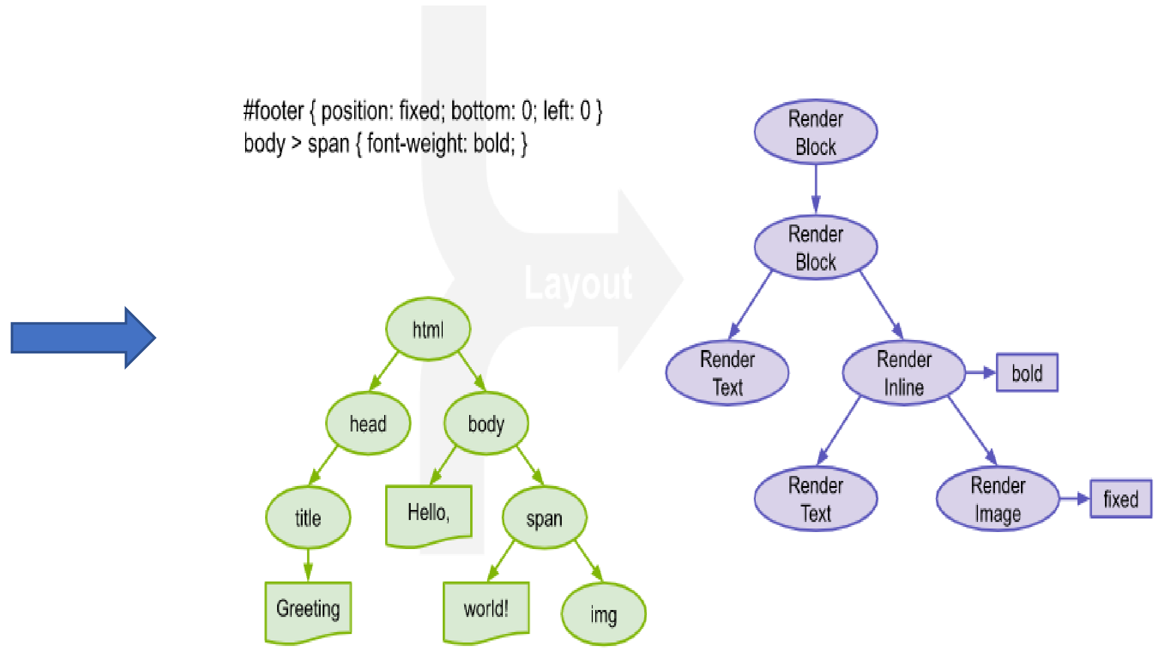
Page, Frame, Document



Render Tree



Bytes
3C 62 6F 64 79 3E 48 65 6C 6C 6F 2C 20 3C 73 70 61 6E 3E 77 6F 72 6C
64 21 3C 2F 73 70 61 6E 3E 3C 2F 62 6F 64 79 3E
Characters
<body>Hello, world!</body>
Tokenizer
Tokens
StartTag: body
Hello,
StartTag: span
world!
EndTag: span
Nodes
body
Hello,
span
world!
TreeBuilder
DOM
body
Hello,
span
world!

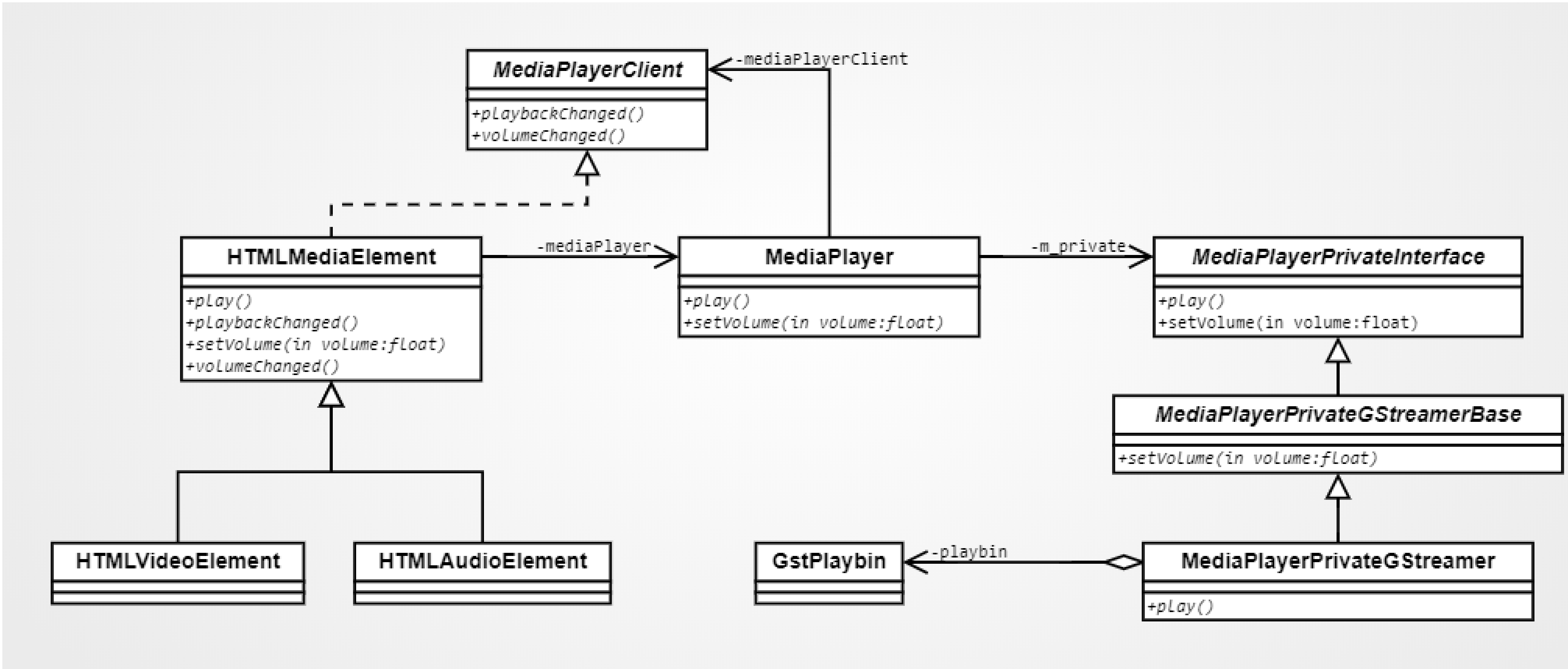


#footer { position: fixed; bottom: 0; left: 0 }
body > span { font-weight: bold; }
Layout
html
head
body
title
Hello,
span
Greeting
world!
img
Render Block
Render Block
Render Text
Render Inline
bold
Render Text
Render Image
fixed

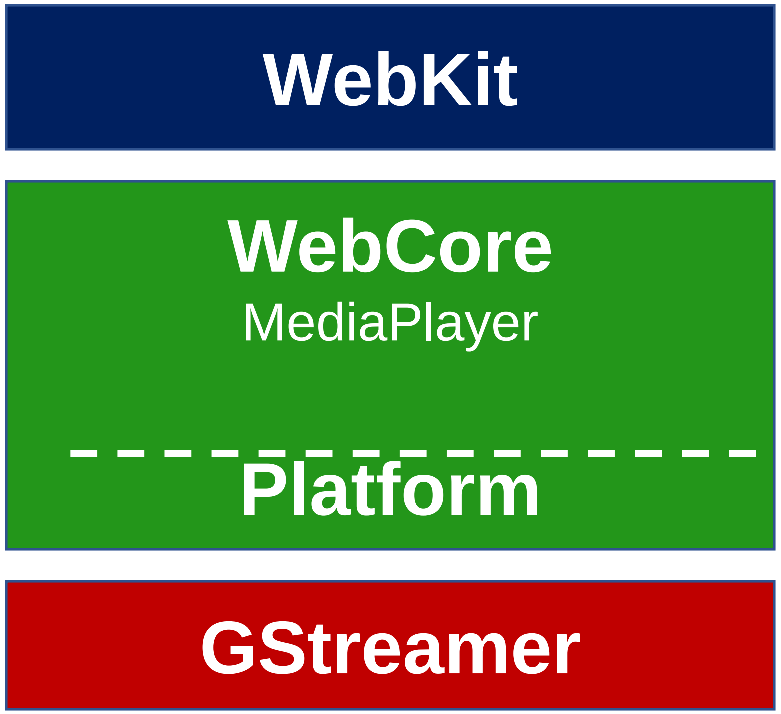
HTML video tags

```
<html> <body>
<div style="text-align:center">
  <button onclick="playPause()">Play/Pause</button>
  <button onclick="makeBig()">Big</button>
  <button onclick="makeSmall()">Small</button>
  <button onclick="makeNormal()">Normal</button>
  <br><br>
  <video id="video1" width="420">
    <source src="mov_bbb.mp4" type="video/mp4">
    <source src="mov_bbb.ogg" type="video/ogg">
</video>
</div>
<script>

var myVideo = document.getElementById("video1");

function playPause() {
  if (myVideo.paused)
    myVideo.play();
  else
    myVideo.pause();
}
</script>

</body> </html>
```

Webkit and Gstreamer class relations



Webkit multimedia architecture



HTML media playback in webkit

- Webkit- APIs – Set of interfaces which an application invokes, a thin layer
- Webcore
  - Platform independent blocks
    - Mediaplayer
  - Corresponding platform implementations for
    - Parsing
    - Layout
    - Network, painting
    - Media playback
- Java script core and platform specific bits

Gstreamer mediaplayer implementation

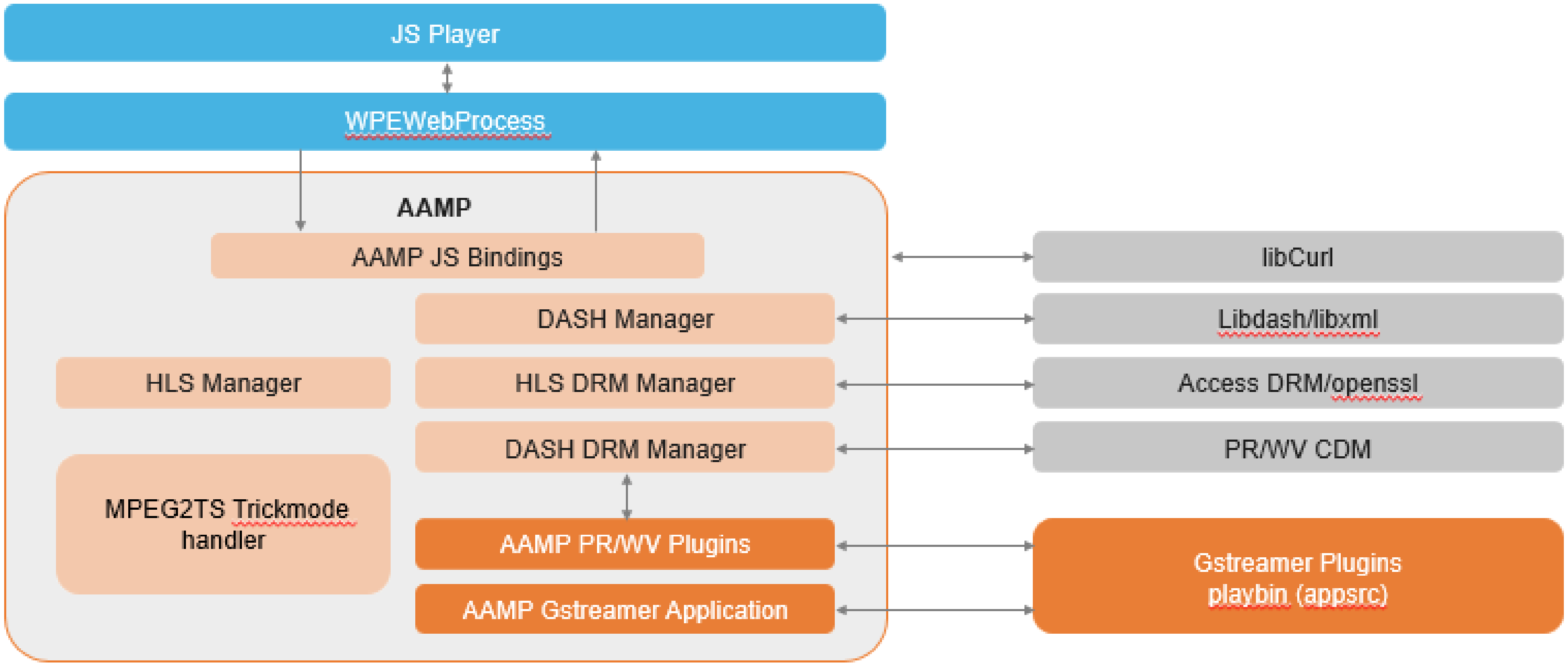
- With playbin
- Custom video sink and source elements
- Basic trick-modes support
- On-disk buffering
- Fullscreen video display
- Frame-accurate seeking
- Basic metrics reporting
- WebAudio
- Hardware decoding support with VA-API (gst 1.2.x)
- Video accelerated compositing
- Codec installer support

Gstreamer in AAMP

- AAMP : Advanced adaptive mediaplayer
- Supports HLS and DASH formats
- Aamp is light weight player compared earlier players used in RDK
- Uses GStremer pipeline
- Two modes of player operation – GStreamer used in both
  - AAMP as a player
  - AAMP as a Plugin

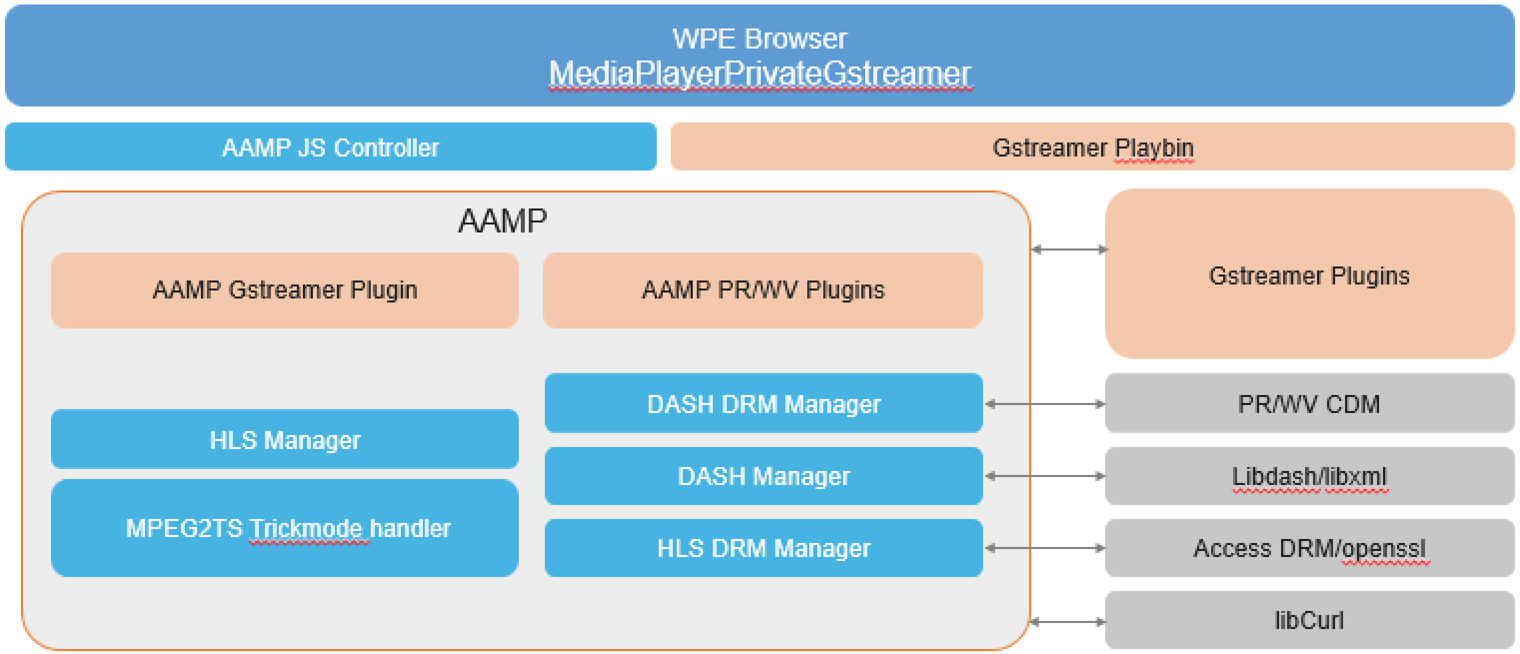
AAMP as a player

- Java Script Player or application directly creates AAMP Player Instance
- AAMP manages the GStreamer pipeline
- Audio /Video buffers are pushed to playbin using appsrc



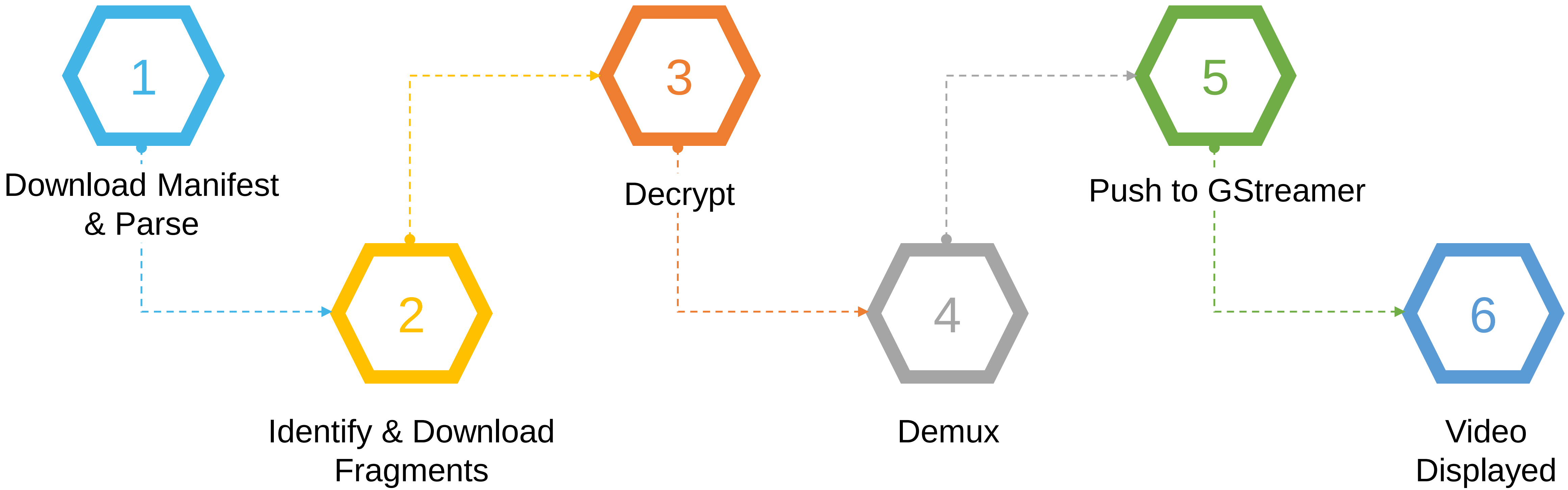
AAMP as a plugin

- Enables HTML5 based playback
- WebKit manages the GStreamer pipeline
- AAMPs GStreamer plugin ‘Gstaamp’ is loaded in the GStreamer pipeline
- Audio /Video buffers are pushed to ‘Gstaamp’s srcpads



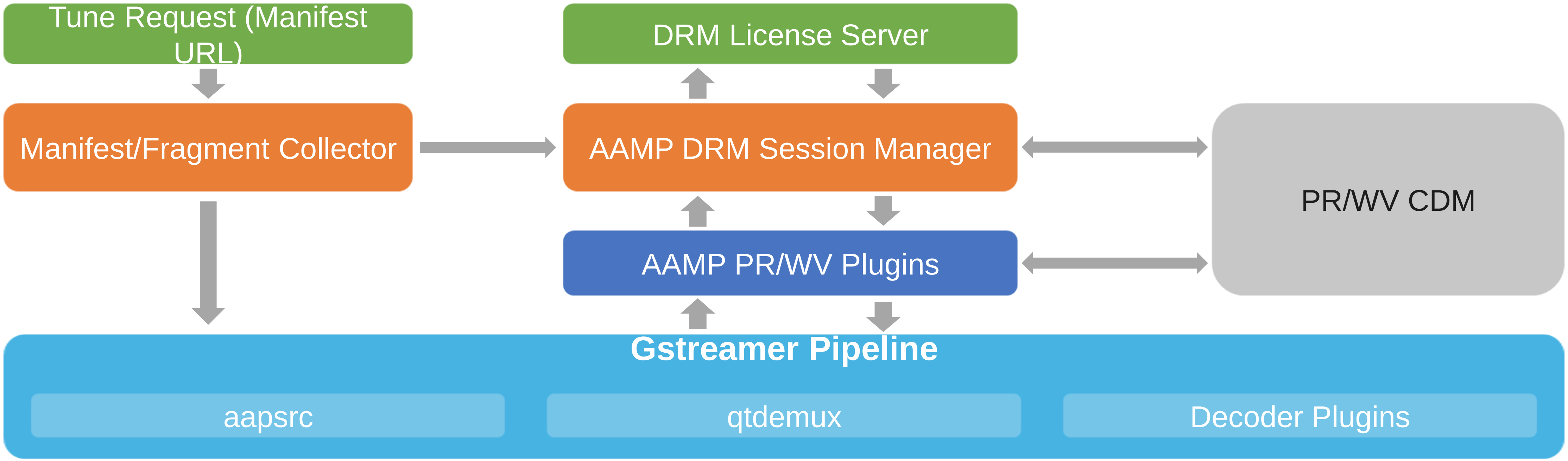
HLS playback

- Downloads main manifest and playlist manifest for normal, trick and playlist for different bitrates
- For HLS, decryption happens before AAMP's default demux
- After demux, it is pushed to gstreamer audio and video pipelines



DASH playback

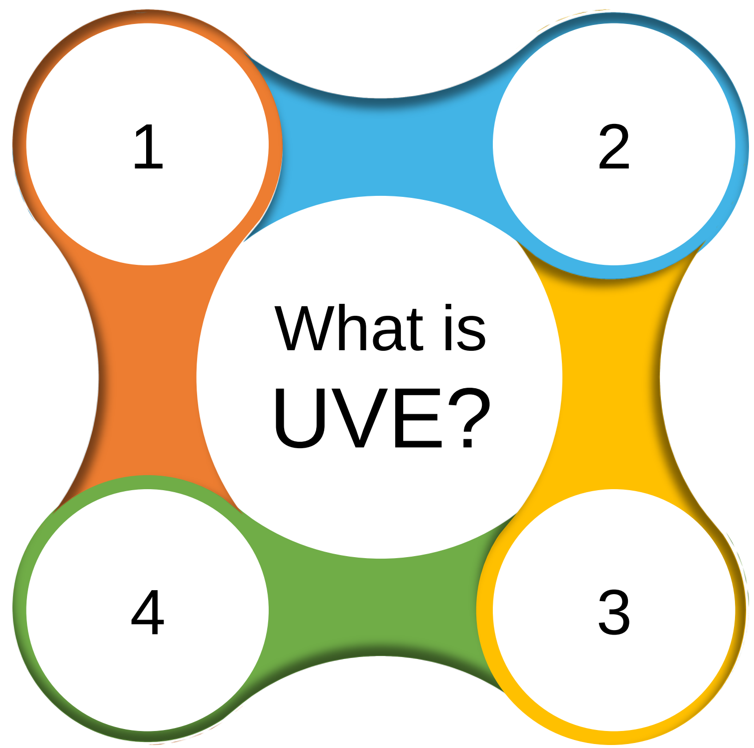
- Single manifest for Dash
- Passes fragments to gstreamer pipeline
- Qtdemux identifies the stream is enryped and hands over to PR/WV
- PR/WV gives info for DRM session creation



Unified Video Engine(UVE)

AAMPs JavaScript binding interfaces

Interface targeting OTT app vendors for migrating apps to STB



Already used by ChannelStore, PlutoTV and MDU Camera apps

Easy to integrate to any JavaScript application

Thank You