CSCI 4140 – Tutorial 6

YouTube IFrame Player API

Matt YIU, Man Tung (mtyiu@cse)
SHB 118
Office Hour: Tuesday, 3-5 pm

2015.02.26

Outline

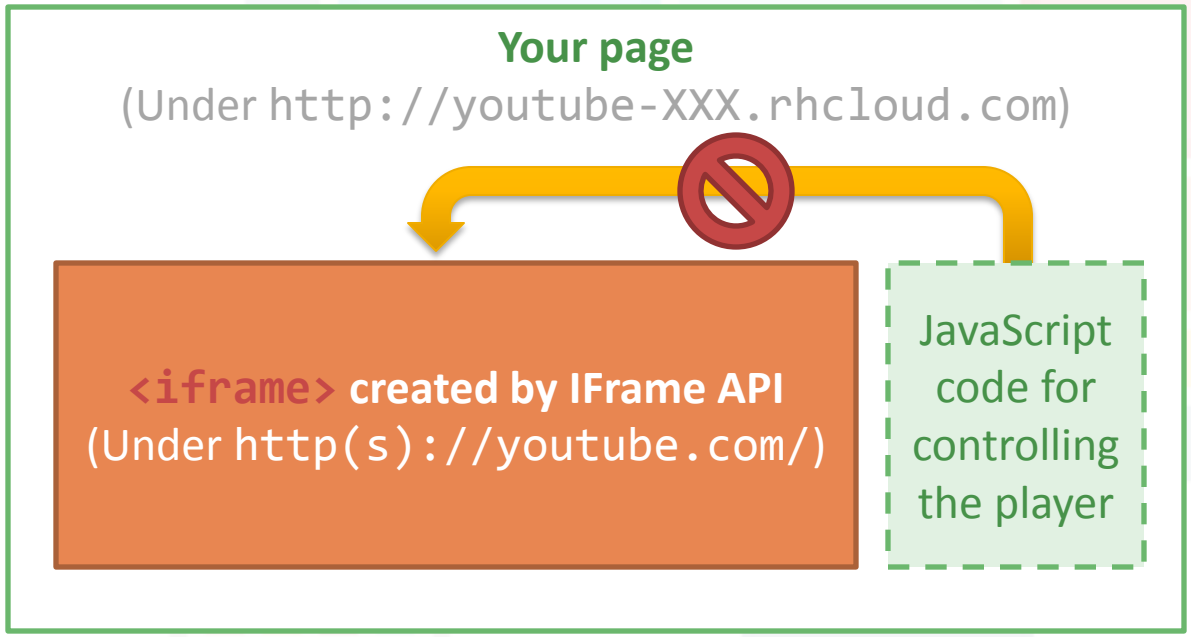
- Overview
- Requirements
- Getting started
- Operations
  - Play, Pause, Stop, Mute, Unmute, Fast Forward, Rewind, Next Video, Previous Video
- The `onStateChange` event
- Destroying the player
- Reference

Overview: What is YouTube IFrame API?

- The **IFrame player API** lets you **embed** and **control** a YouTube video player on your website using **JavaScript**
- It posts content to an `<iframe>` tag on your page
  - It provides more **flexibility** because YouTube can either serve an **HTML5 player** or a Flash player
  - You do not need to care about which player is used!
- The API
  - Provides **operations** for video playback
  - Triggers **events** that can be handled by your **event listeners**

Requirements

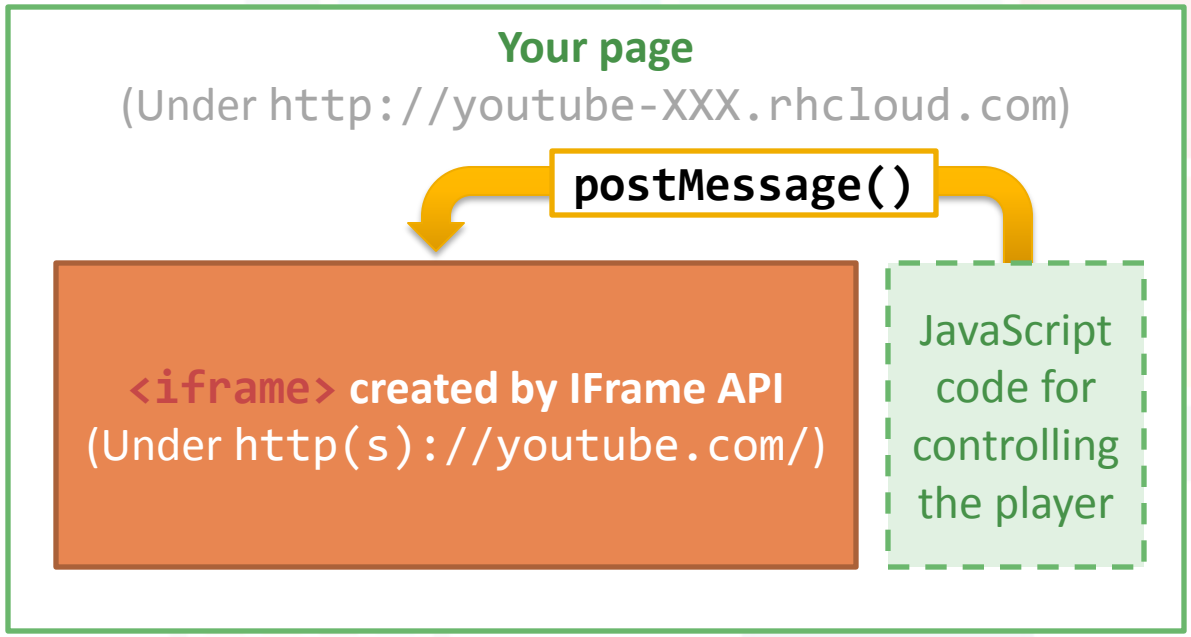
- Browsers that support the HTML5 **postMessage** feature
  - Chrome / Safari / Firefox support it (forget about IE 7 or below...)
  - Why HTML5?



Before HTML5, it is impossible for the scripts on a site to access resources (e.g., access to DOM) from other sites (different origin) due to security reasons.
This is called the **same-origin policy (SOP)**.

Requirements

- Browsers that support the HTML5 **postMessage** feature
  - Chrome / Safari / Firefox support it (forget about IE 7 or below...)
  - Why HTML5?



HTML5 defines the `window.postMessage` method which enables cross-origin communication. For example, the YouTube site tells the browser that it is going to receive messages from other sites. Then, a site on OpenShift can send messages to this YouTube site.

Requirements

- Browsers that support the HTML5 **`postMessage`** feature
  - Chrome / Safari / Firefox support it (forget about IE 7 or below...)
- **Viewport** that is at least 200px by 200px
  - This ensures that the player can be fully displayed
  - In Assignment 2, you don't need to care about this as the player is only displayed on desktop view (i.e., device width = **`md`** or **`lg`** in Bootstrap)
- Implemented **`onYouTubeIframeAPIReady`** JS function
  - The API calls this function when the page has finished downloading the JavaScript for the player API

- Now let's see how to use the API

Getting started: Sample HTML page

```
<!DOCTYPE html>
<html>
    <body>
        <div id="player"></div>
        <script src="player.js"></script>
    </body>
</html>
```

youtube-iframe-api/index.html

Tell the browser that you are using HTML5 (also required by Bootstrap).

This `<div>` serves as a placeholder, which will be replaced by the `<iframe>` and video player later.

Embed the JavaScript code to the page.
Note: It is a good practice to separate the JavaScript from the document content.

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

This code loads the IFrame API code asynchronously.

youtube-iframe-api/player.js

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

Once the IFrame API code is loaded, the `onYouTubeIframeAPIReady()` function is executed. This function creates an `<iframe>` and YouTube player.

youtube-iframe-api/player.js

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_a
var firstScriptTag = document.getElementsBy
firstScriptTag.parentNode.insertBefore( tag
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

youtube-iframe-api/player.js

The first parameter specifics either the DOM element or the id of the HTML element where the API will insert the <iframe> tag containing the player.

In this example, this refers to the element:

```
<div id="player"></div>
```

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

The second parameter is an object which specifies player options.

youtube-iframe-api/player.js

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

youtube-iframe-api/player.js

Let's disable the control bar inside the player. We only control the video playback on our remote in Assignment 2.

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

Specify the ID of the YouTube video to be loaded by the player. In Assignment 2, this should be the first entry in the playlist.

youtube-iframe-api/player.js

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIfr
    player = new YT.P
        height : '390
        width : '640
        playerVars: {
        videoId : 'MX
        events : {
            'onReady
            'onStateC
        }
    } );
}
```

youtube-iframe-api/player.js

In case you don't know, the video ID of an YouTube video can be retrieved from its URL:

https://www.youtube.com/watch?v=vJMmt7FyI2s

The GET parameter of "v" (i.e., v=XXXXXXXXXX) contains the video ID.

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTa
firstScriptTag.parentNode.insertBefore( tag,
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

Event name

Event listener

To set up event listeners, pass the function to the "events" object.

Note: Read the complete source code for the implementation of the event listeners (i.e., onPlayerReady() & onPlayerStateChange()).

youtube-iframe-api/player.js

Getting started: Sample JavaScript

```
var tag = document.createElement( 'script' );
tag.src = "https://www.youtube.com/iframe_api";
var firstScriptTag = document.getElementsByTagName( 'script' )[ 0 ];
firstScriptTag.parentNode.insertBefore( tag, firstScriptTag );
var player;
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

youtube-iframe-api/player.js

Remember to save the `player` object returned from new `YT.Player()` such that you can control the player later.

Operations

- After getting a **reference** to the player object (stored in the `player` variable), you can use the player API methods to control the video playback
- Operations required in Assignment 2:
  - Play
  - Pause
  - Stop
  - Mute
  - Unmute
  - Fast forward
  - Rewind
  - Previous video
  - Next video

Operations: Play, Pause, Stop, Mute, Unmute

- Suppose the player has loaded a video
- Methods for performing these operations:
  - Play:

```
player.playVideo();
```

  - Pause:

```
player.pauseVideo();
```

  - Stop:

```
player.stopVideo();
```

  - Mute:

```
player.mute();
```

  - Unmute:

```
player.unMute();
```

Operations: Fast Forward, Rewind

- Implementing **fast forward** and **rewind** is a bit tricky as the API does not provide these methods directly
- However, there is a **`seekTo()`** method which allows you to seek to a **specified time** in the video

```
player.seekTo(seconds:Number, allowSeekAhead:Boolean);
```

Time to which the player should advance (in seconds)

Allow or disallow the player to seek beyond the currently buffered portion of the video
For simplicity, please set it to `true`!

- We can also use **`getCurrentTime()`** to get the **elapsed time** of the currently playing video in seconds

```
currentTime = player.getCurrentTime();
```

Operations: Fast Forward, Rewind

- Combining these two methods:

Note: This is a our simplified version of fast forward and rewind in Assignment 2.

```
When the user click the "Fast Forward" button:
    currentTime = player.getCurrentTime();
    player.seekTo( currentTime + 2.0 ); // Seek to 2 seconds later
End

When the user click the "Rewind" button:
    currentTime = player.getCurrentTime();
    player.seekTo( currentTime - 2.0 ); // Seek to 2 seconds before
End
```

Pseudocode

- Note: You can assume that the user will never "fast forward" or "rewind" out of the video, i.e.,

2 < currentTime < Video length (in seconds) - 2

Operations: Next Video, Previous Video

- User manages the playlist in our application (by either adding or removing videos from the playlist)

Warning: Don't use **`nextVideo()`** or **`previousVideo()`**

  - They are for loading and playing the next/previous video in a ***YouTube*** playlist
    - Example: https://www.youtube.com/watch?v=V9Gljy4q3wk&list=PL8F997915F5135BF3
  - This is totally different from the playlist managed in our application
  - So, how to implement this feature?

- **Answer:** Use the **`player.loadVideoById()`** method

Operations: Next Video, Previous Video

- Let's say we store the playlist in an array called "`playlist`"
- The video currently playing corresponds to the *i*-th entry of the playlist

```
When the user click the "Next Video" button:
    var id = playlist[ i + 1 ]; // Get the next video ID in the playlist
    player.loadVideoById( id ); // Change the current video
    i++; // Update current index of the playlist
End

When the user click the "Previous Video" button:
    var id = playlist[ i - 1 ]; // Get the previous video ID in the playlist
    player.loadVideoById( id ); // Change the current video
    i--; // Update current index of the playlist
End
```

Pseudocode

The onStateChange event

- Note that the player will just stop when the currently playing video ends (which is not the expected behavior)
  - However, the player should **play the next video in the playlist**!
  - How can we **detect** that a video has stopped playing?

```
function onYouTubeIframeAPIReady() {
    player = new YT.Player( 'player', {
        height : '390',
        width : '640',
        playerVars: { 'controls' : 0 },
        videoId : 'M7lc1UVf-VE',
        events : {
            'onReady': onPlayerReady,
            'onStateChange': onPlayerStateChange
        }
    } );
}
```

Do you remember the event listeners set in the `onYouTubeIframeAPIReady()` method?
We will use the `onStateChange` event to detect the change in player state .

The onStateChange event

```
function onPlayerStateChange(event){
   switch( event.data ) {
      case YT.PlayerState.ENDED:
          // ...
          break;
      case YT.PlayerState.PLAYING:
          // ...
          break;
      case YT.PlayerState.PAUSED:
          // ...
          break;
      case YT.PlayerState.BUFFERING:
          // ...
          break;
      case YT.PlayerState.CUED:
          // ...
          break;
      default:
          // ...
   }
}
```

- This is the event listener of the event "onStateChange"
- The event is triggered when the player changes its state (e.g., from playing to pause/stop)
- The argument of the event listener (**event**) has an attribute called "data"
- By checking its value, we know the current state of the player, and we can act accordingly

The onStateChange event

```
function onPlayerStateChange(event){
    switch( event.data ) {
        case YT.PlayerState.ENDED:
            // ...
            break;
        case YT.PlayerState.PLAYING:
            // ...
            break;
        case YT.PlayerState.PAUSED:
            // ...
            break;
        case YT.PlayerState.BUFFERING:
            // ...
    }
}
```

You need to load the next video in the playlist when the current state of the player is "ENDED".

Note: For other state changes, you may handle or ignore them (depends on your implementation)

- This is the event listener of the event "onStateChange"
- The event is triggered when the player changes its state (e.g., from playing to pause/stop)
- The argument of the event listener (**event**) has an attribute called "data"
- By checking its value, we know the current state of the player, and we can act accordingly

Destroying the player

- When the device width is below **`md`** or **`lg`** in Bootstrap (i.e., tablet and mobile view), the player should not be displayed
- If the player is already loaded, you are strongly suggested **destroying** the player when it is not shown
  - This avoids the video player to keep on playing videos even it is not displayed
- To destroy a player object:

```
player.destroy();
```

- Remember to re-create the player object when the page is restored to **desktop view** later

Destroying the player

- To detect the width of the browser window, use

```
window.innerWidth
```

- When the device width is below **md** or **lg** in Bootstrap (i.e., tablet and mobile view), the player should not be displayed
  - i.e., the player is displayed only if **window.innerWidth >= 992**

```
if ( window.innerWidth >= 992 ) {
   Display the video player
} else {
   player.destroy(); // Destroy the video player
}
```

Pseudocode

References

- YouTube Player API Reference for iframe Embeds:
  - https://developers.google.com/youtube/iframe_api_reference?hl=en
  - Most of the contents of this tutorial come from this page
- YouTube Player Demo:
  - https://developers.google.com/youtube/youtube_player_demo

– End –