WiFi Direct in Tizen Web Applications

Łukasz Jagodziński
Samsung Electronics

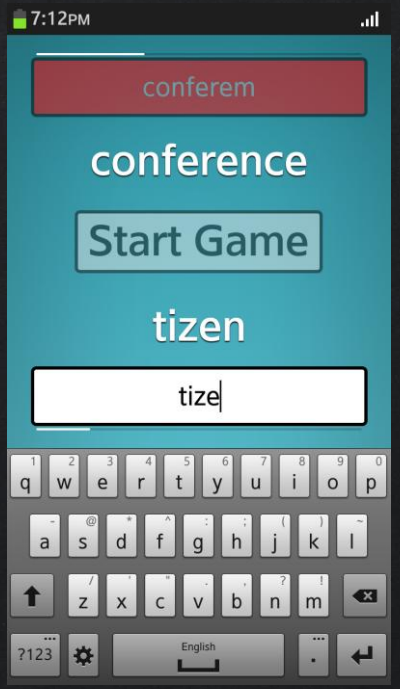
TIZEN™ DEVELOPER CONFERENCE 2014 SAN FRANCISCO

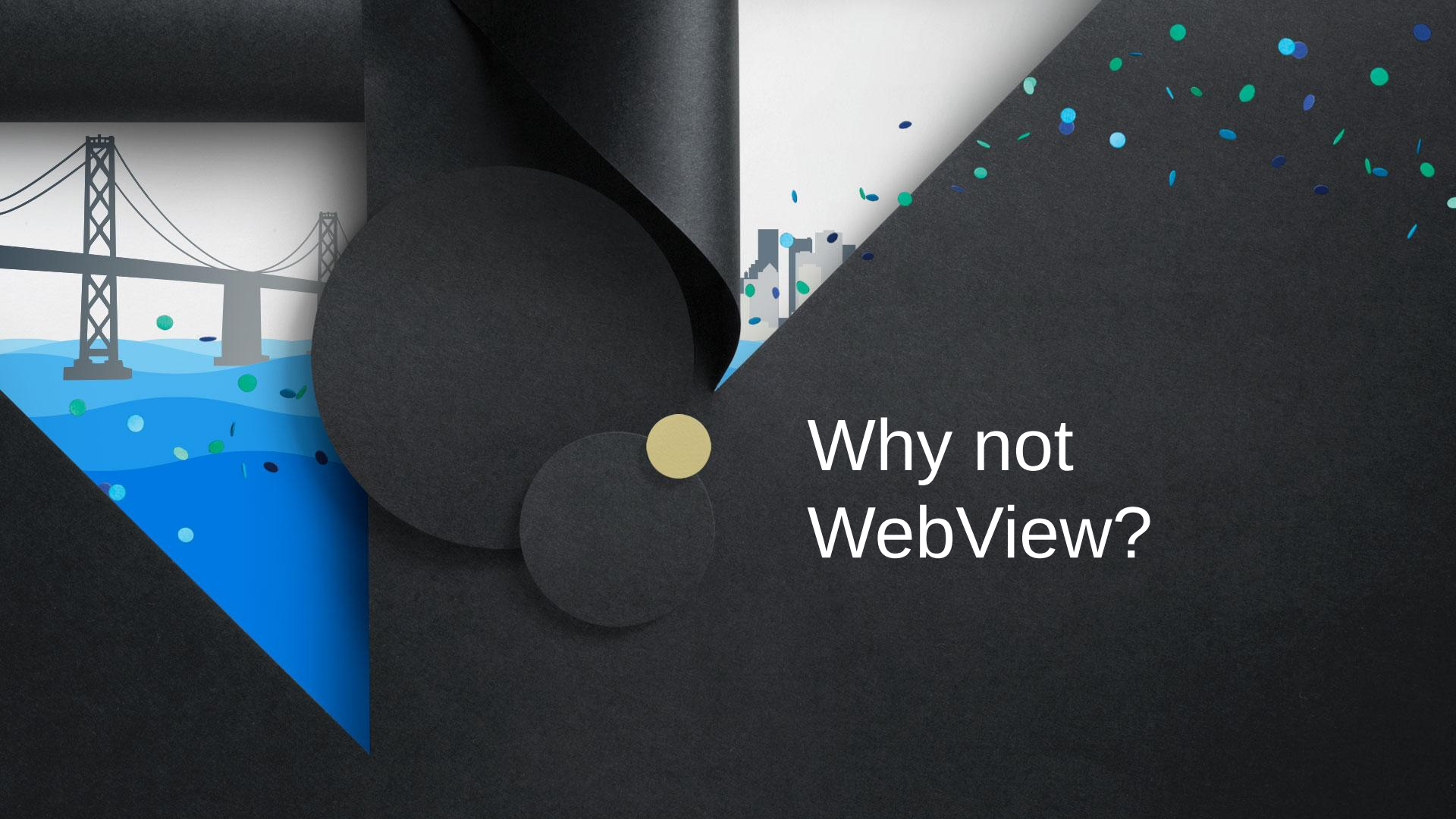

Why
WiFi Direct?

Requirements

- **Multiplayer game without server side management**
- **More than two players playing together**
- **Reusable library**
- **Programming in JavaScript**
- **Fast, realtime communication**
  - Dynamic games
  - Streaming big amount of data

„TypeRace” game demo




Why not
WebView?

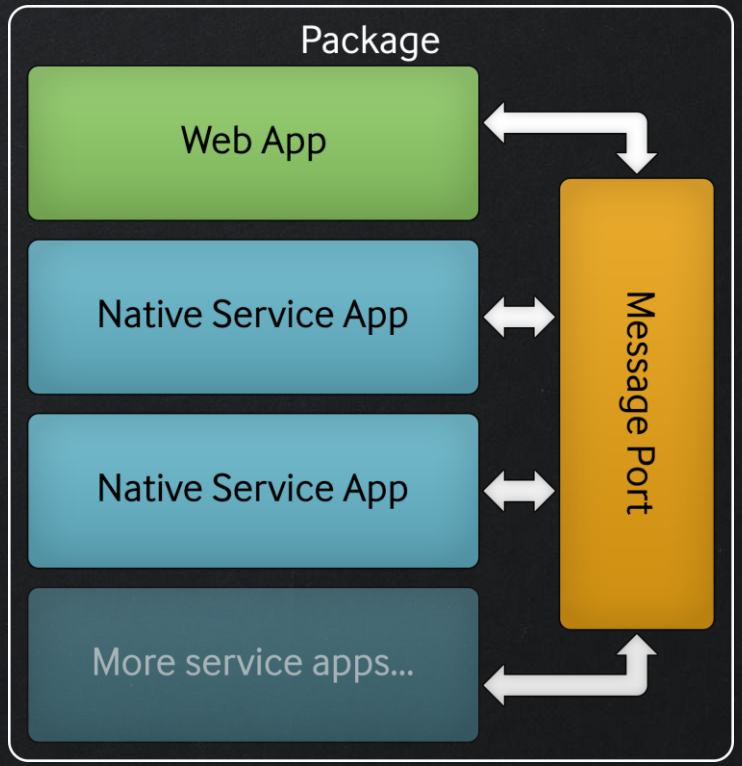
Disadvantages of WebView

- Debugging is harder
- Native application in the center
- Can’t create reusable library
- Lack of some APIs

Solution: Hybrid Application

What is Hybrid Application?

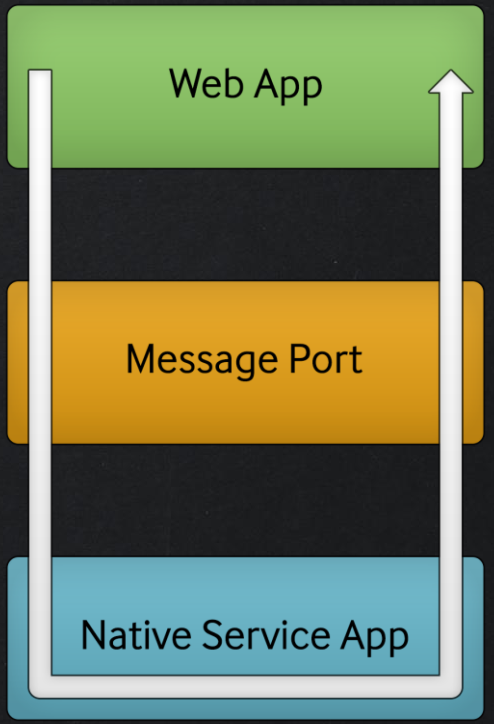
- Two or more applications (native and web) in one package
- Web application in the center
- Native application as a service that exports some API
- Message Port communication between applications




Message Port

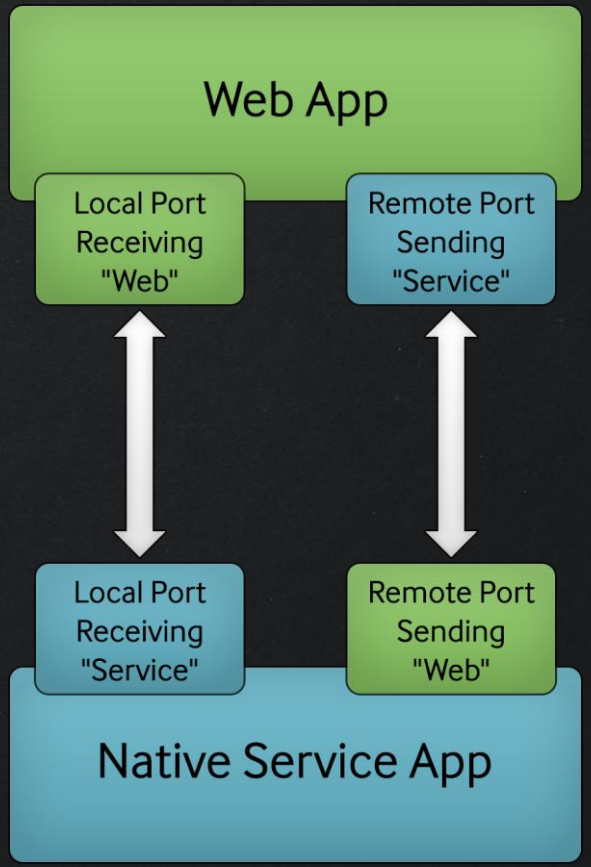
Communication speed

- Round trip time < 5 ms



Local and remote ports

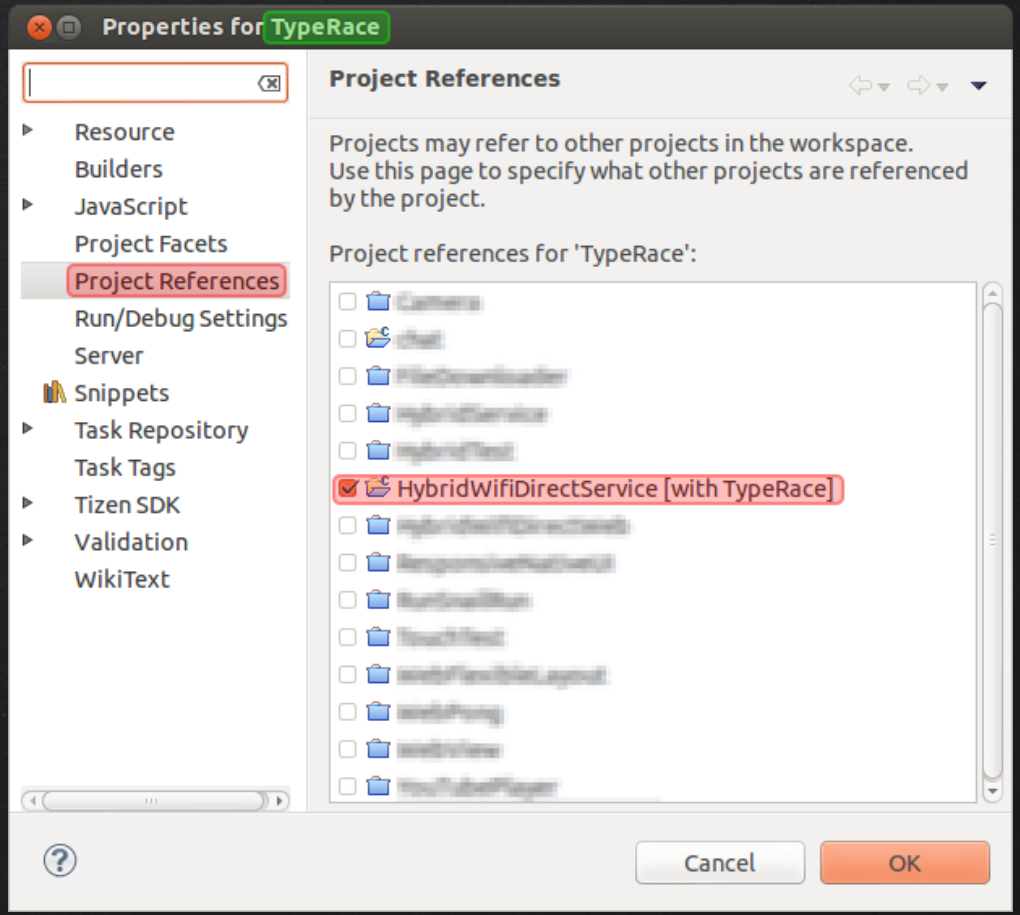
- **Ports:**
  - Local – receiving port
  - Remote – sending port
- **Bidirectional communication**




WiFi Direct
Library

Library components

- **Library consists of:**
  - HybridWifiDirectService service application
  - tizen.hybrid.js files
  - tizen.wifidirect.js file



Usage

```
tizen.wifidirect.init(initCF);

tizen.wifidirect.addEventListener('activated', activatedCF);
tizen.wifidirect.activate(activateCF);

tizen.wifidirect.addEventListener('groupCreated', groupCreatedCF);
tizen.wifidirect.createGroup(createGroupCF);

tizen.wifidirect.addEventListener('scanCompleted', scanCompletedCF);
tizen.wifidirect.scan(scanCF);

tizen.wifidirect.addEventListener('connected', connectedCF);
tizen.wifidirect.connect(deviceInfo, connectCF);

tizen.wifidirect.initSocket(initSocketCF);

tizen.wifidirect.addEventListener('messageReceived', messageReceivedCF);
tizen.wifidirect.sendBroadcast('Hello World!', sendBroadcastCF);
tizen.wifidirect.sendMessage('192.168.0.10', 'Hello World!', sendBroadcastCF);
```

Functions list

```
getServiceId()
getServiceName()
getLocalMessagePortName()
getServicePortName()

init()
activate()
deactivate()
isActivated()
isDiscoverable()
scan()
cancelScan()
connect()
cancelConnect()
disconnect()
```

```
createGroup()
leaveGroup()
getGroupClientInfoList()
getGroupOwnerInfo()
getGroupSettingInfo()
setGroupSettingInfo()
setLocalDeviceName()
getLocalDeviceInfo()
getOperatingChannel()
getWpsConfigurationModePreference()
setWpsConfigurationModePreference()

initSocket()
sendBroadcast()
sendMessage()

addEventListener()
onReceive()
```

Request codes

```
var _Request = {
    INIT:         100,
    // Activation
    ACTIVATE:     0,
    DEACTIVATE:   1,
    // Scanning
    SCAN:         2,
    CANCEL_SCAN:  3,
    /* ... */
};
```

```
struct Request {
    static const int INIT        = 100;
    // Activation
    static const int ACTIVATE    = 0;
    static const int DEACTIVATE  = 1;
    // Scanning
    static const int SCAN        = 2;
    static const int CANCEL_SCAN = 3;
    /* ... */
};
```

Response codes

```
var _Response = {
    INIT:             100,
    // Activation
    ACTIVATE:         0,
    ACTIVATED:        1,
    DEACTIVATE:       2,
    DEACTIVATED:      3,
    // Scanning
    SCAN:             4,
    CANCEL_SCAN:      5,
    SCAN_COMPLETED:   6,
    /* ... */
};
```

```
struct Response {
    static const int INIT           = 100;
    // Activation
    static const int ACTIVATE       = 0;
    static const int ACTIVATED      = 1;
    static const int DEACTIVATE     = 2;
    static const int DEACTIVATED    = 3;
    // Scanning
    static const int SCAN           = 4;
    static const int CANCEL_SCAN    = 5;
    static const int SCAN_COMPLETED = 6;
    /* ... */
};
```

Message format

- Messages sent as JSON strings

```
// Stringify data and send to
// service application.
remoteMessagePort.sendMessage([{
    key: 'request',
    value: request
}, {
    key: 'data',
    value: JSON.stringify(data)
}], localMessagePort);
```

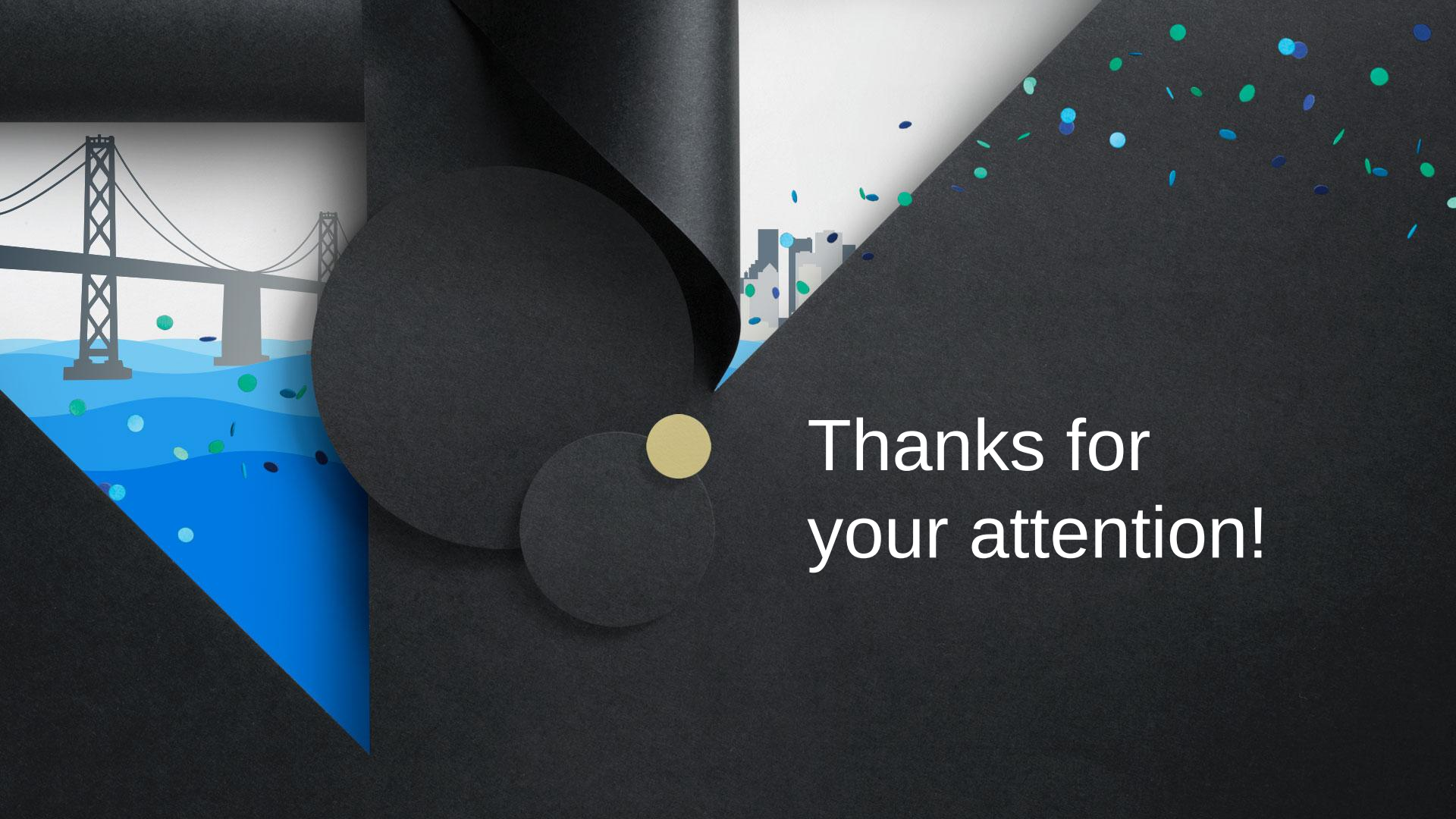
```
// Convert JSON object to JSON string and send
// data to web application.
String pJsonStr = JsonUtility::Stringify(pJson);
r = SendToWeb(pJsonStr);

// Fragment of 'SendToWeb' function
HashMap* pMap = new HashMap(SingleObjectDeleter);
pMap->Construct();
pMap->Add(
    new String(L"json"),
    new String(pJsonStr)
);
__pMessagePort->SendMessage(pMap);
```

Resources

- Wi-Fi Direct and Sockets in Tizen web applications - Article
  https://developer.tizen.org/documentation/articles/wi-fi-direct-and-sockets-tizen-web-applications
- Wi-Fi Direct™ Connectivity - Dev Guide
  https://developer.tizen.org/dev-guide/2.2.1/org.tizen.native.appprogramming/html/guide/net/wi-fi_direct_connectivity.htm


Any questions?


Thanks for
your attention!


TIZEN™
DEVELOPER
CONFERENCE
2014
SAN FRANCISCO