WebForm2
HTMLを使った
入力ダイアログ

WebForm2はWindows10以上で、Microsoft Edgeと同じエンジンWebView2コンポーネントを使うWebアプリユーティリティです。
OgO-JavaScriptのプログラムからコントロールします。

WebForm2の仕様

WebForm2はOgOからBrowserを操作するライブラリー。

[機能]
入力DialogをHTMLで実装し
それをOgOのスクリプトで作成・操作し、入力値を取得する。

[メリット]
WebForm2以前は、入力GUIはC++で実装し、開発時間がかかった。また、MFCなど複雑でライブラリーの知識が必要であった。

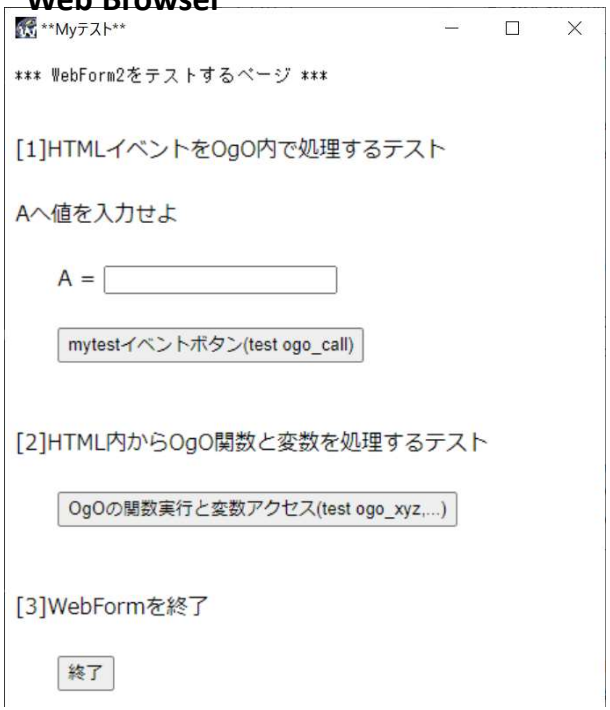
GUIを、つ将来ずっと標準となるHTMLで実装し、制御をOgOスクリプトで実現することは大きな価値がある。

その価値とは、入力画面が早く簡単に作成でき、テクニックはWebでいくらでも探して利用できる。

それとWebの豊富な資源を有効利用できます。

Browser＋HTML ＋OgO

Web Browser



=

== HTMLコード ==

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"/>
<title>WebForm2テスト</title>
<script type="text/javascript">
 ・・・・
</script>
</head>

<body>
<pre>
*** WebFormをテストするページ ***
</pre>
<br>
<div>
[1]HTMLイベントをOgO内で処理するテスト<br>
    <div id="xyz">Aへ値を入力せよ</div><br>
    A = <input type="text" id="A"><br><br>
    <input type="button" id="mytest" value="mytestイベントボタン" onclick="ogo_callt('mytest')"><br>
</div>
   ・・・・・
</body>
</html>
```

作成

関数呼出

イベント

関数呼出

== OgOスクリプト ==

```
var {WebForm2} = require('wbf2');
var opts = {wsize:[500,600], title:"**Myテスト**"};
var wbf = new WebForm2("file:///"+__dirname+"/webform2-sample.html", opts);
wbf.set_env(((()=>{
  async function mytest() {
// html内の変数 abcを操作
    await wbf.set("abc", 12345);
    alert("[ogo->html] abc=" + await wbf.get("abc")); //abc=12345
  // html内の関数をcall
    alert("[ogo->html] add(10,20)=>" + await wbf.call("add", 10,20));
 // htmlのdocumentを操作
    alert("[Aへ値を入力せよ]を[Input value to A]に変更します!");
    await wbf.set("document.getElementById('xyz').innerText", "Input value to
A");//=> <div id="xyz" name="xyz">Input value to A</div>
```

```
   // html textテキスト入力値を取得
   alert("[ogo->html document A] A="+await wbf.get_value('A
  }
  function myclose() {
   wbf.close();
  }
  let ans_sum;
  function sum(n) {
   let ans = 0;
   for (let i=1; i<=n; i++) ans += i;
   ans_sum = ans;
   return ans;
  }
  return (_s,..._a)=>eval(_s);
})());
wbf.show();
```

WebForm2
Browser＋HTMLをOgOで
操作するライブラリ

- WebForm2ライブラリを使うための準備

  var {WebForm2} = require(‘wbf2’);

WebForm2作成

- var wbf = new WebForm2(url[, ops]);

url : HTMLのリソースを指定
opts={title, wsize, wpos, howner, icon, curdir, env, menu}
  title: Windowタイトル名
  wsize:[width, height]
  wpos:[left, top, width, height]
  visible: wbf作成後、表示(true)/非表示(false)。省略された場合 非表示(false) 。
  howner: wbfの親Windowハンドル 、-1ならアプリメインのWindowハンドルに内部変換
  icon: アイコンファイルパス名
  curdir: wbfカレントディレクトリ
  env: HTMLからアクセス可能な環境オブジェクト (後でwbf.set_env(…)で設定可能)
  menu: true/false メニューバーの表示/非表示
  html: , url: , uri: 第一引数のurlをこのプロパティ名で指定可能

```
var  wbf = new  WebForm2('file:///webform2-sample.html', {wsize:[600, 700]});
            /*基本的にWindowは非表示。表示するには
              諸々の設定メソッド関数を実行後、
              wbf.show();  または wbf.navigate(...);
              を実行すれば、Windowが表示されます.*/
```

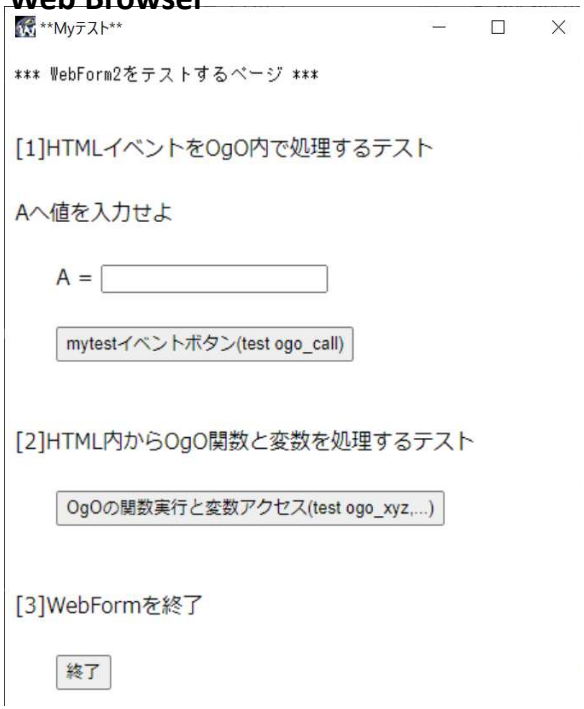
- 位置・サイズ変更

```
var  wpos = wbf.wpos();   //(pos[0],pos[1]):左上  (pos[2],pos[3]):幅、高さ
wbf. wpos (top,left, W, H);  // W:幅、H:高さ
```

- Windowハンドルの取得

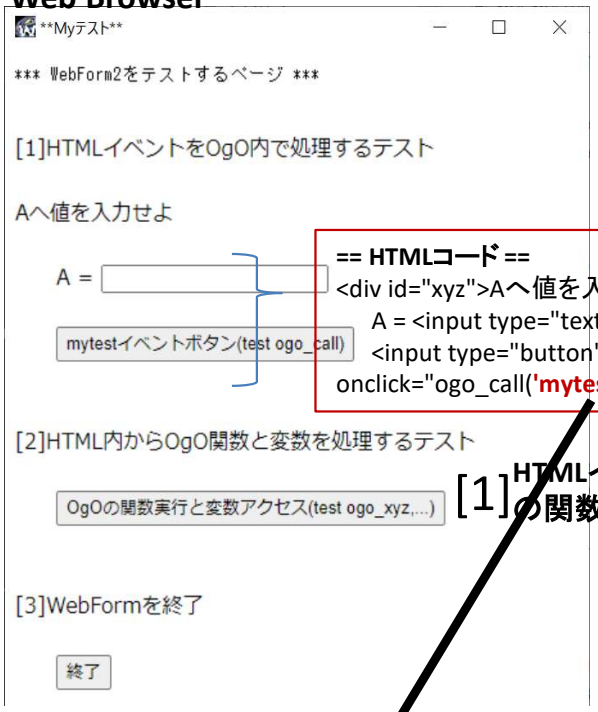
```
var  hwnd = wbf.hwnd;
```

Web Browser



イベント処理とHTMLの操作(in OgO)

Web Browser



== HTMLコード ==

```
<div id="xyz">Aへ値を入力せよ</div><br>
  A = <input type="text" id="A"><br><br>
  <input type="button" id="mytest" value="mytestイベントボタン"
onclick="ogo_call('mytest')"><br>
```

== HTML内JavaScriptコード ==

```
<script type="text/javascript"><!—
let abc = 100;
 function add(a,b) { return a + b; }
```

```
async function test() {
  let ret = await ogo_call("sum",100);
  alert("[html->ogo]sum(100) => " +ret);
  alert("[html->ogo] ans_sum = " + await
ogo_get("ans_sum"));
}
--></script>
```

[1] HTMLイベントをOgO内の関数callする

[2]

変数アクセス

v = await wbf.get (変数名); // 取得

await wbf.set(変数名, value); // 設定

関数呼び出し

ret = await wbf.call (関数名, 引数,...);

要素オブジェクトのvalue属性値

wbf.get("**document**.getElementById(**ID名**)); //取得

wbf.set("**document**.getElementById(**ID名**)", **value**); //設定

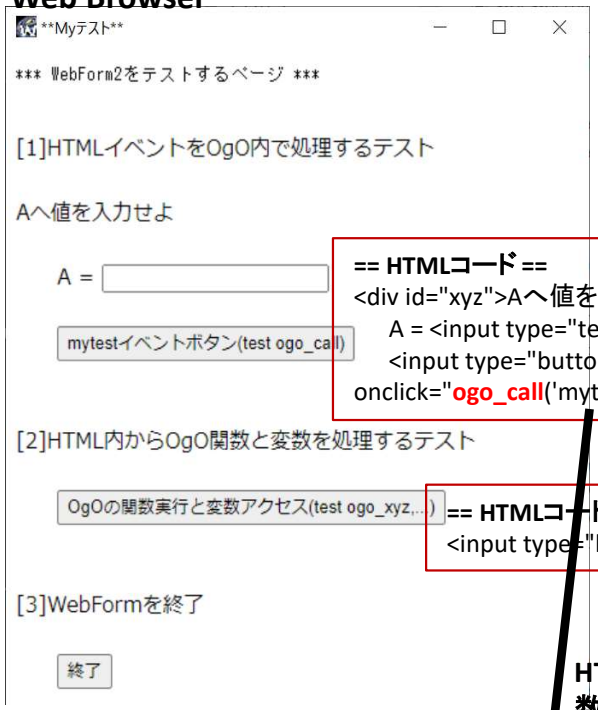
== OgOスクリプト ==

```
let {WebForm2} = require('wbf2');
let opts = {wsize:[500,600], title:"**Myテスト**"};
let wbf = new WebForm2("file:///"+__dirname+"/webform2-sample.html", opts);

wbf.set_env(((()=>{
 async function mytest() {

  // html内の変数 abcを操作
  await wbf.set("abc", 12345);
  alert("[ogo->html] abc=" + await wbf.get("abc")); //abc=12345

  // html内の関数をcall
  alert("[ogo->html] add(10,20)=>" + await wbf.call("add", 10,20));
```

```
  // htmlのdocumentを操作
  alert("[Aへ値を入力せよ]を[Input value to A]に変更します!");
  await wbf.set("document.getElementById('xyz').innerText", "Input value to A
  // html textテキスト入力値を取得
  alert("[ogo->html document A] A="+await wbf.get_value('A'));
 }
 ......
 return (_s,..._a)=>eval(_s);
})());

wbf.show();
```

HTML内からOgOを操作(in HTML)

Web Browser



== HTMLコード ==

```
<div id="xyz">Aへ値を入力せよ</div><br>
   A = <input type="text" id="A"><br><br>
   <input type="button" id="mytest" value="mytestイベントボタン"
onclick="ogo_call('mytest')"><br>
```

== HTMLコード ==

```
<input type="button" value="OgOの関数実行と変数アクセス" onclick="test()">
```

== HTML内JavaScriptコード ==

```
<script type="text/javascript">
・・・
async function test() {
  let ret = await ogo_call("sum",100);
  alert("sum(100) => " +ret);
  alert("ans_sum = " + await ogo_get("ans_sum"));
}
</script>
```

[1]

[0]

HTMLイベントをOgO内の関数callする

[2]

変数アクセス

v = await **ogo_get**(変数名); // 取得

await **ogo_set**(変数名, value); // 設定

関数呼び出し

ret = await **ogo_call**(関数名, 引数);

== OgOスクリプト ==

```
wbf.set_env((()=>{
  async function mytest() {
    ・・・・
    ・・・・
    return (_s,..._a)=>eval(_s);
  }
})());
```

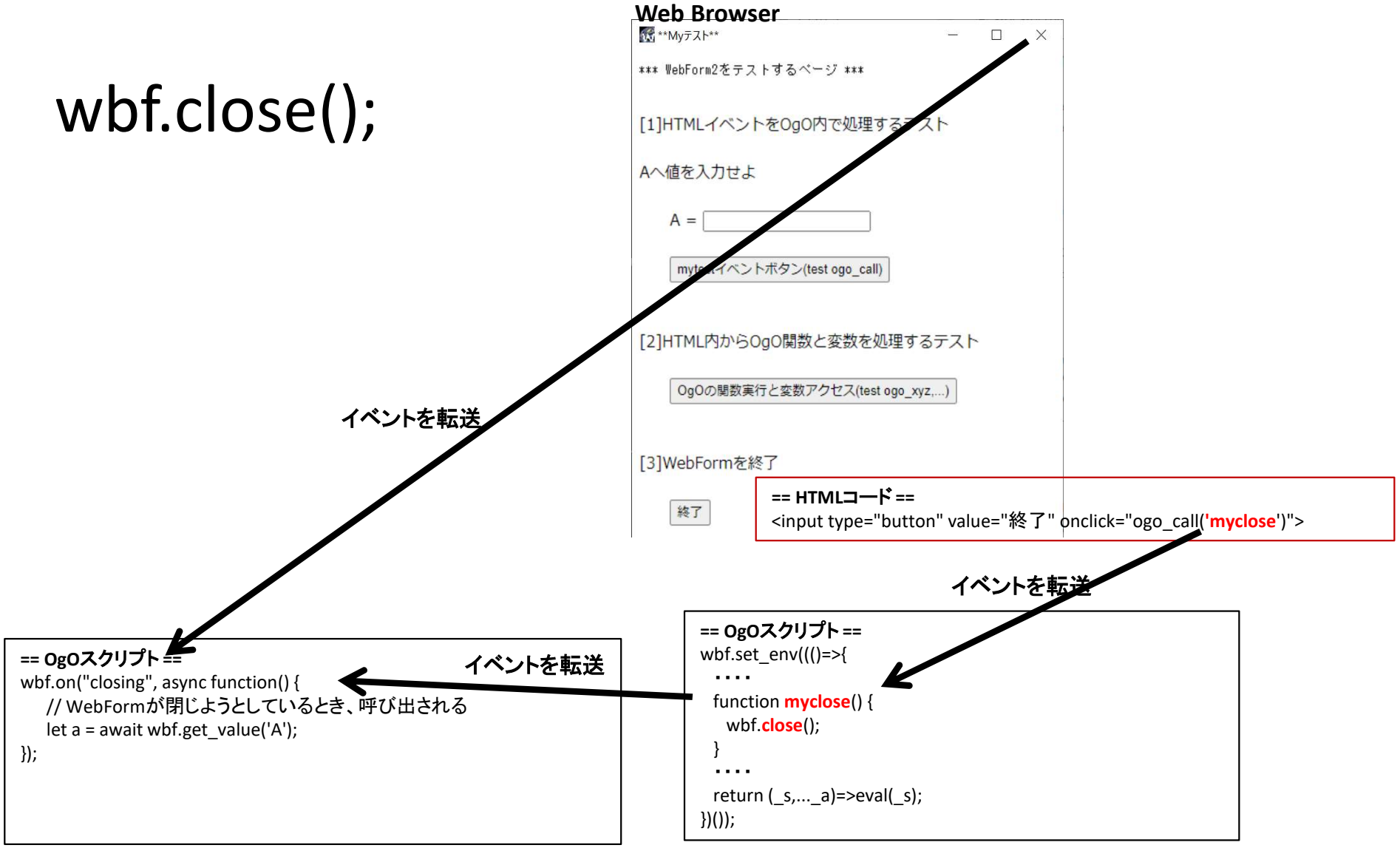
== OgOスクリプト ==

```
var ans_sum;

function sum(n) {
  let ans = 0;
  for (let i=1; i<=n; i++) ans += i;
  ans_sum = ans  return ans;
}
```

WebFormを終了

- webform2オブジェクト.close()

  wbf.close();



WebForm API

- WebFormオブジェクト

in OgO

var {WebForm2} = require('wbf2');
new WebForm2() --- コンストラクタ, WebForm2.allClose() --- static function
navigate(url), show(), hide(), close(), isClosed(), wpos(…), title(nam), set_env(env) --- method
hwnd, howner --- property
on/addEventListener(eventname, handler), off/removeEventListener(eventname), messageReceiver(hanndler) --- event

- OgOからHTML内の関数、変数、DOMを扱う(非同期)

in OgO

get(name), set(name,value) --- HTML内のJS変数へのアクセス
call (fname, …args) --- HTML内のJS関数呼び出し
exec(script, …args) --- HTML内でスクリプトを実行
postMessage(data) --- HTMLへデータを送る
get_value(id), set_value((id.value), get_number(id), set_number(id,num) --- HTML id要素アクセス
get_checked(id), set_checked(id, checked), get_radio(name), set_radio(name,id)
get_option(id), set_option(if,opid), get_attribute(id,attri), set_attribute(id,attri,value)
get_innerHTML(id), set_innerHTML(id, html)

- HTMLからOgO内の関数、変数を扱う(非同期)

in HTML

ogo_get(name), ogo_set(name,value) --- OgO内の変数へアクセス
ogo_call(fname, …args) --- OgO内の関数呼び出し
ogo_exec(script,…args) --- OgO内でスクリプトを実行
ogo_postMessage(data) --- OgOへデータを送る

WebForm2オブジェクト

in OgO

- WebFormライブラリ呼び出し
  let {WebForm2} = require('wbf2');
- WebFormコンストラクタ new WebForm2(...)
  let wbf = new WebForm2(url[, ops]); --- 指定url のコンテンツを表示するWebForm2オブジェクトを作成する。
  基本的にまだWindowは表示されない。
  wbf.show()またはwbs.navigate(url) を実行すれば表示される。
  url : HTMLのリソースを指定
  opts={title, wsize, wpos, howner, icon, curdir, env, menu}
  title: Windowタイトル名
  wsize:[width, height]
  wpos:[left, top, width, height]
  visible: wbf作成後、表示(true)/非表示(false)。省略された場合 非表示(false) 。
  howner: wbfの親Windowハンドル -1 ならアプリメインWindowハンドル
  icon: アイコンファイルパス名
  curdir: wbfカレントディレクトリ
  env: wbf.set_env(...)に渡せる環境オブジェクト(HTMLのJavaScriptからアクセスできる)
  menu: true/false メニューバーの表示/非表示
  html: , url: , uri: 第一引数をこのプロパティ名で指定可能

in OgO

- wbf.navigate(url)
  urlのコンテンツを表示する。urlのコンテンツをload完了したら、
  イベント" navigationCompleted"が発生します。
- wbf.close()
  WebForm2を閉じる。この関数が実行されると、Windowをclose中、イベント"closing" が発生します。
- wbf.isClosed()
  WebForm2が閉じられてればtrueを返し、まだ使えればfalseを返します。
- wbf.show(visible=true)
  visible=trueなら表示、visible=falseなら非表示
  省略ならvisible=trueと同じで表示する。
- wbf.hide()
  wbf.show(false)と同じ。非表示する

メソッド

- ret=wbf.title() --- タイトル取得
  wbf.title(name) ---タイトル設定
- pos = wbf.wpos () --- (pos[0],pos[1]):左上、(pos[0],pos[1]):幅・高さ
  wbf.position(left,top, width,height) --- 位置とサイズを設定
- wbf.set_env((()=>{
  //HTMLからアクセス許可する変数、関数を記述
  ・・・・・
  return (_s,…_a)=>eval(_s);
  })());

- wbf.hwnd (ReadOnly)
  WebForm2のbrowserWindowハンドル

in OgO

プロパティ

- wbf.howner
  WebForm2のbrowserWindowはhowner Windowの前面に必ず表示する。
  Howner=0なら、どのWindowに対しても前面/後面の関係はなさない。

- wbf.on/addEventListener (eventname, func)
  イベント(eventname)とそのを処理関数(func)を登録する。
  登録できるイベントは次の3つ。
  イベント navigationCompleted:
  function (args) { //args={isSuccess, WebErrorStatus, NavigationId}
  ….
  }
  イベント closing:
  function () {
  …. //HTML内の要素にアクセス可能
  }
  イベント messageReceived:
  function (data) {
  …. //dataはHTMLからogo_postMessage(data)で送られたdata
  }

イベント

- wbf.off/removeEventListener (eventname)
  wbf.on/addEventListener(...)で登録したイベントとその関数を削除する。
- wbf.messageReceiver(function(data) {
  …. //dataはHTMLからogo_postMessage(data)で送られたdata
  }
  これは
  wbf.on/addEventListener ("messageReceived", function(data) {
  …. //dataはHTMLからogo_postMessage(data)で送られたdata
  }
  と同じ。

in OgO

OgOからHTML内の関数,変数,DOMを扱う(非同期)

in OgO

- HTML内のJS変数・要素へのアクセス

ret = await wbf.get(name) (ret=await wbf.exec("name"))
--- HTML内JS変数・要素の値を取得
await wbf.set(name, value) (await wbf.exec("name=value")
--- HTML内JS変数・要素に値valueを代入

- HTML内のJS関数呼び出し

ret = await wbf.call(fname, …args) (ret=await wbf.exec("fname(…args)"))

- HTML内でJSスクリプトを実行

ret = await wbf.exec(script, …args)
…args引数はスクリプト内で、_a[0],_a[1],…でアクセスできます。
retはscript最後実行された式の値です。

- HTMLへデータを送る

wbf.postMessage(data)
dataは,数値、文字列、Object, 配列, null値
HTML側は
ogo_messageReceiver(function(data) {
・・・・
});
で受け取ります。

- HTML要素のvalue値

ret = await wbf.get_value(id) (ret=await wbf.get(document.getElementById(id).value)
await wbf.set_value(id, value) (await wbf.set(document.getElementById(id).value, value)

- HTML要素のvalue数値

ret = await wbf.get_number(id) (ret=+await wbf.get(document.getElementById(id).value)
await wbf.set_number(id, num) (await wbf.set(document.getElementById(id).value, num)

- HTML要素のchecke属性値

ret = await wbf.get_checked(id) (ret=await wbf.get(document.getElementById(id).checked)
await wbf.set_checked(id, checked) (await wbf.set(document.getElementById(id).checked, checked)

- (nameのradio)HTML要素群からselect id

ret = await wbf.get_radio (name) --- checkedされてる要素のidを取得
await wbf.set_radio(name, id) --- name要素内のid要素にcheckedする

- selectタグ要素内にあるoption要素を選択

ret = wbf.get_option (id) --- 選択されたoption要素のidを取得
wbf.set_option(id, opid) --- opidのoption要素を選択状態にする

- HTML要素の属性値へのアクセス

ret = await wbf.get_attribute(id, attri) (ret=await wbf.exec("document.getElementById(id).getAttribute(attri)"))
await wbf.get_attribute(id, attri, value) (await wbf.exec("document.getElementById(id).setAttribute(attri, value)"))

- HTML要素内のHTMLの取得と設定

ret=await wbf.get_innerHTML(id) (ret=await wbf.get_attribute(id,"innerHTML"))
await wbf.set_innerHTML(id, html) (await wbf.set_attribute(id,"innerHTML", html)))

HTMLからOgO内の関数、変数を扱う(非同期)

in HTML

- OgO内のJS変数へのアクセス

ret = await ogo_get(varname)
--- OgO内JS変数 varnameの値を取得
varname="objname.prop.propc" もOK
await ogo_set(varname, value)
--- OgO内JS変数 varnameに値valueを代入
varname="objname.prop.propc" もOK

- OgO内のJS関数呼び出し

ret = await ogo_call(funcname, ...args)
--- OgO内のJS関数 (名前 funcname)を呼び出し実行する。
funcname ="objname.prop.func" もOK
funcname ="**.**prop.func" であった場合、OgOスクリプトのwebform**.**prop.func(引数)が実行される

- OgO内でJSスクリプトを実行

ret=await ogo_exec(script, ...args)
--- args引数はスクリプト内で、a[0], a[1],…でアクセスできます。
retはscript最後実行された式の値です。

- OgOへデータを送る

ret=await ogo_postMessage (data)
--- dataは,数値、文字列、Object, 配列, null値
OgO側は
wbf.messageReceiver(function(data) {
・・・・
});
で受け取ります。

OgOからHTML内へのアクセス補助ユーティリティ

in HTML

- OgOに提供する実行環境をセットする

```
ogo_set_env((()=>{
    //OgOからアクセス許可する変数、関数を記述
     .....
    return (_s,..._a)=>eval(_s);
})());
```

上記の実行環境をセットしなくてもHTML内のJSグローバル変数にアクセスできるよう初期設定してます。

- OgOからwbf.postMessage(data)で送られたデータを受け取る

```
ogo_messageReceiver(function(data) {
   dataが受け取った値
}
```

サンプル・チェックプログラム

- webform2-sample.js, webform2-sample.html

== OgOスクリプト == webform2-sample.js

```
let {WebForm2} = require('wbf2');

let opts = {wsize:[500,600], title:"**Myテスト**"};
let wbf = new WebForm2("file:///"+__dirname+"/webform2-sample.html", opts); //非表示 諸々の設定後表示 wbf.show();

wbf.on("navigationCompleted",
 async function(args) {
  await wbf.set_value("A", 987654);
 }
);

wbf.on("closing",
 async function() {
  let a = await wbf.get_value('A');
  /*
  alert(a);
  */
 }
);

wbf.set_env(((()=>{
 async function mytest() {
  alert("[ogo]イベント mytest");

  // html内の変数 abcを操作
  await wbf.set("abc", 12345);
  alert("[ogo->html] abc=" + await wbf.get("abc")); //abc=12345

 // html内の関数をcall
  alert("[ogo->html] add(10,20)=>" + await wbf.call("add", 10,20));

  // htmlのdocumentを操作
  alert("[Aへ値を入力せよ]を[Input value to A]に変更します!");
  await wbf.set("document.getElementById('xyz').innerText", "Input value to A");//=> <div id="xyz" name="xyz">Input value to A</div>
  // html textテキスト入力値を取得
  alert("[ogo->html document A] A="+await wbf.get_value('A'));  //or await wbf.get("document.getElementById('A').value")

 }

 function myclose() {
  wbf.close();
 }

 //----
 let ans_sum;

 function sum(n) {
  let ans = 0;
  for (let i=1; i<=n; i++) ans += i;
  ans_sum = ans;
  return ans;
 }

 return (_s,..._a)=>eval(_s);
})());

wbf.show();  //諸々の設定した最後に表示
```

== HTMLコード == webform2-sample.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>

<title>WebForm2テスト</title>

<script type="text/javascript">
<!--
//{
 let abc = 100;
 function add(a,b) { return a + b; }
//}
async function test() {
 let ret = await ogo_call("sum",100);
 alert("[html->ogo]sum(100) => " +ret);
 alert("[html->ogo] ans_sum = " + await ogo_get("ans_sum"));
}
-->
</script>

</head>
<body>
<pre>
*** WebForm2をテストするページ ***
</pre>
<br>
<div>
[1]HTMLイベントをOgO内で処理するテスト<br>
    <div id="xyz">Aへ値を入力せよ</div><br>
    A = <input type="text" id="A" test="test..."><br><br>
    <input type="button" id="mytest" value="mytestイベントボタン(test ogo_call)"
onclick="ogo_call('mytest')"><br>
</div>
<br><br>
<div>
[2]HTML内からOgO関数と変数を処理するテスト<br>
  <br>
    <input type="button" value="OgOの関数実行と変数アクセス(test ogo_xyz,...)"
onclick="test()"><br>

</div>
<br><br>
<div>
[3]WebFormを終了<br>
  <br>
    <input type="button" value="終了" onclick="ogo_call('myclose')"><br>
</div>

</body>
</html>
```