Colorpicker für CIE (Hue, Osram)

<!--

 ioBroker.vis colorpicker Widget-Set

 version: "1.1.1"

 Copyright 2013-2016 hobbyquaker https://github.com/hobbyquaker, bluefox https://github.com/GermanBluefox

-->

<link rel="stylesheet" type="text/css" href="widgets/colorpicker/css/farbtastic.css" />

<link rel="stylesheet" type="text/css" href="widgets/colorpicker/css/spectrum.css" />

<script type="text/javascript" src="widgets/colorpicker/js/jscolor.js"></script>

<script type="text/javascript" src="widgets/colorpicker/js/farbtastic.js"></script>

<script type="text/javascript" src="widgets/colorpicker/js/spectrum.js"></script>

<script type="text/javascript" src="widgets/colorpicker/js/huepiHelper.js"></script>

<script type="text/javascript">

 if (vis.language === 'de'){

 var localization = $.spectrum.localization.de = {

 cancelText: 'Abbrechen',

 chooseText: 'WÃ¤hlen',

 clearText: 'Farbauswahl zurÃ¼cksetzen',

 noColorSelectedText: 'Keine Farbe ausgewÃ¤hlt',

 togglePaletteMoreText: 'Mehr',

 togglePaletteLessText: 'Weniger'

 };

 $.extend($.fn.spectrum.defaults, localization);

 } else if (vis.language === 'ru'){

 var localization = $.spectrum.localization.ru = {

 cancelText: 'Ð¾Ñ‚Ð¼ÐµÐ½Ð°',

 chooseText: 'Ð²Ñ‹Ð±Ñ€Ð°Ñ‚ÑŒ',

 clearText: 'ÑÐ±Ñ€Ð¾ÑÐ¸Ñ‚ÑŒ',

 noColorSelectedText: 'Ð½Ð¸ÐºÐ°ÐºÐ¾Ð¹ Ñ†Ð²ÐµÑ‚ Ð½Ðµ Ð²Ñ‹Ð±Ñ€Ð°Ð½',

 togglePaletteMoreText: 'Ð±Ð¾Ð»ÑŒÑˆÐµ',

 togglePaletteLessText: 'Ð¼ÐµÐ½ÑŒÑˆÐµ'

 };

 $.extend($.fn.spectrum.defaults, localization);

 }

 if (vis.editMode) {

 // Add words for basic widgets

 $.extend(true, systemDictionary, {

 "red-oid": {"en": "Red ID", "de": "Rot ID", "ru": "ÐšÑ€Ð°ÑÐ½Ñ‹Ð¹ ID"},

 "green-oid": {"en": "Green ID", "de": "GrÃ¼n ID", "ru": "Ð—ÐµÐ»Ñ‘Ð½Ñ‹Ð¹ ID"},

 "blue-oid": {"en": "Blue ID", "de": "Blau ID", "ru": "Ð¡Ð¸Ð½Ð¸Ð¹ ID"},

 "divisor": {"en": "divisor", "de": "Divisor", "ru": "Ð”ÐµÐ»Ð¸Ñ‚ÐµÐ»ÑŒ"},

 "decimal": {"en": "Precision", "de": "Nach Komma", "ru": "Ð§Ð¸ÑÐµÐ» Ð¿Ð¾ÑÐ»Ñ Ð·Ð°Ð¿ÑÑ‚Ð¾Ð¹"},

 "command-oid": {"en": "Command ID", "de": "Command ID", "ru": "Command ID"},

 "xy-oid": {"en": "XY ID", "de": "XY ID", "ru": "XY ID"},

 "level-oid": {"en": "Level ID", "de": "Level ID", "ru": "Level ID"},

 "gamut": {"en": "Gamut/Model", "de": "Gamut/Model", "ru": "Gamut/Model"},

 "gamut\_tooltip": {

 "en": "Gamut ID (A,B,C) or Model ID (LCT001 etc.)",

 "de": "Gamut ID (A,B,C) oder Model ID (LCT001 usw.)",

 "ru": "Gamut ID (A,B,C) or Model ID (LCT001 etc.)"

 },

 "transitionTime": {"en": "Transition Time", "de": "Ãœbergangszeit", "ru": "Transition Time"},

 "transitionTime\_tooltip": {

 "en": "Transition time in 1/10s",

 "de": "Ãœbergangszeit in 1/10s",

 "ru": "Transition time in 1/10s"

 },

 "pickerBackground": {"en": "Background Color", "de": "Hintergrundfarbe", "ru": "Background Color"},

 "pickerWidth": {"en": "Picker Width", "de": "Farbwahl Breite", "ru": "Picker Width"},

 "pickerHeight": {"en": "Picker Height", "de": "Farbwahl HÃ¶he", "ru": "Picker Height"},

 "buttonName": {"en": "Button Text", "de": "Buttontext", "ru": "Button Text"},

 "closeButton": {"en": "Close button", "de": "SchlieÃŸen Button", "ru": "Close button"},

 "group\_rgb": {"en": "R.G.B.", "de": "R.G.B.", "ru": "R.G.B."},

 "rgb-oid": {"en": "RGB ID", "de": "RGB ID", "ru": "RGB ID"},

 "group\_hue": {"en": "H.S.L.", "de": "H.S.L.", "ru": "H.S.L."},

 "hue-oid": {"en": "HUE ID", "de": "HUE ID", "ru": "HUE ID"},

 "sat-oid": {"en": "Saturation ID", "de": "SÃ¤ttigung ID", "ru": "ID ÐÐ°ÑÑ‹Ñ‰ÐµÐ½Ð½Ð¾ÑÑ‚ÑŒ"},

 "bri-oid": {"en": "Brightness ID", "de": "Helligkeit ID", "ru": "ID Ð¯Ñ€ÐºÐ¾ÑÑ‚ÑŒ"}

 });

 }

 vis.binds.colorpicker = {

 version: "1.1.1",

 showVersion: function () {

 if (vis.binds.colorpicker.version) {

 console.log('Version vis-colorpicker: ' + vis.binds.colorpicker.version);

 vis.binds.colorpicker.version = null;

 }

 },

 jscolor: function (el, data) {

 data = data.\_data || data;

 var $this = $(el);

 var handler = [];

 var factor = parseFloat(data.factor) || 1;

 var decimal = parseInt(data.decimal || 0, 10) || 0;

 var picker = new jscolor($this[0], {

 valueElement: data.showValue || false,

 onFineChange: function () {

 for (var h = 0; h < handler.length; h++) {

 handler[h]();

 }

 },

 closable: !!data.closeText,

 closeText: data.closeText

 });

 $this.click(function () {

 picker.show();

 });

 if (data['hue-oid'] && data['hue-oid'] !== 'nothing\_selected' &&

 data['sat-oid'] && data['sat-oid'] !== 'nothing\_selected' &&

 data['bri-oid'] && data['bri-oid'] !== 'nothing\_selected') {

 var hue = data['hue-oid'];

 var sat = data['sat-oid'];

 var bri = data['bri-oid'];

 handler.push(function () {

 vis.setValue(hue, parseFloat(picker.hsv[0]).toFixed(decimal));

 vis.setValue(sat, parseFloat((picker.hsv[1] / factor).toFixed(decimal)));

 vis.setValue(bri, parseFloat((picker.hsv[2] / factor).toFixed(decimal)));

 });

 var htimer = null;

 var hueChanged = function () {

 if (htimer) clearTimeout(htimer);

 htimer = setTimeout(function () {

 htimer = null;

 picker.fromHSV(

 Math.round(parseFloat(vis.states[hue + '.val'] || 0)),

 Math.round(parseFloat(vis.states[sat + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[bri + '.val'] || 0) \* factor)

 );

 }, 10);

 };

 vis.states.bind(hue + '.val', hueChanged);

 vis.states.bind(sat + '.val', hueChanged);

 vis.states.bind(bri + '.val', hueChanged);

 hueChanged();

 }

 if (data['red-oid'] && data['red-oid'] !== 'nothing\_selected' &&

 data['green-oid'] && data['green-oid'] !== 'nothing\_selected' &&

 data['blue-oid'] && data['blue-oid'] !== 'nothing\_selected') {

 var r = data['red-oid'];

 var g = data['green-oid'];

 var b = data['blue-oid'];

 handler.push(function () {

 vis.setValue(r, parseFloat((picker.rgb[0] / factor).toFixed(decimal)));

 vis.setValue(g, parseFloat((picker.rgb[1] / factor).toFixed(decimal)));

 vis.setValue(b, parseFloat((picker.rgb[2] / factor).toFixed(decimal)));

 });

 var timer = null;

 var rgbChanged = function () {

 if (timer) clearTimeout(timer);

 timer = setTimeout(function () {

 timer = null;

 picker.fromRGB(

 Math.round(parseFloat(vis.states[r + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[g + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[b + '.val'] || 0) \* factor)

 );

 }, 10);

 };

 vis.states.bind(r + '.val', rgbChanged);

 vis.states.bind(g + '.val', rgbChanged);

 vis.states.bind(b + '.val', rgbChanged);

 rgbChanged();

 }

 if (data['rgb-oid'] && data['rgb-oid'] !== 'nothing\_selected') {

 var rgb = data['rgb-oid'];

 handler.push(function () {

 vis.setValue(rgb, picker.toHEXString());

 });

 var \_rgbChanged = function () {

 picker.fromString(vis.states[rgb + '.val'] || '');

 };

 vis.states.bind(rgb + '.val', \_rgbChanged);

 \_rgbChanged();

 }

 },

 rgb2hex: function (r,g,b) {

 return '#' + ('0' + r.toString(16)).slice(-2) + ('0' + g.toString(16)).slice(-2) + ('0' + b.toString(16)).slice(-2);

 },

 hex2rgb: function (hex) {

 var r = parseInt(hex.substr(1, 2), 16);

 var g = parseInt(hex.substr(3, 2), 16);

 var b = parseInt(hex.substr(5, 2), 16);

 return {r: r, g: g, b: b};

 },

 farbtastic: function (el, data) {

 var $this = $(el);

 $this.hide();

 data = data.\_data || data;

 var handler = [];

 var factor = parseFloat(data.factor) || 1;

 var decimal = parseInt(data.decimal || 0, 10) || 0;

 var ft = $.farbtastic($this.prev(), function () {

 for (var h = 0; h < handler.length; h++) {

 handler[h](this.color);

 }

 });

 if (data['hue-oid'] && data['hue-oid'] !== 'nothing\_selected' &&

 data['sat-oid'] && data['sat-oid'] !== 'nothing\_selected' &&

 data['bri-oid'] && data['bri-oid'] !== 'nothing\_selected') {

 var hue = data['hue-oid'];

 var sat = data['sat-oid'];

 var bri = data['bri-oid'];

 handler.push(function (hex) {

 var color = vis.binds.colorpicker.hex2rgb(hex);

 vis.setValue(hue, parseFloat(picker.hsv[0]).toFixed(decimal));

 vis.setValue(sat, parseFloat((picker.hsv[1] / factor).toFixed(decimal)));

 vis.setValue(bri, parseFloat((picker.hsv[2] / factor).toFixed(decimal)));

 });

 var htimer = null;

 var hueChanged = function () {

 if (htimer) clearTimeout(htimer);

 htimer = setTimeout(function () {

 htimer = null;

 ft.setColor(vis.binds.colorpicker.rgb2hex(

 Math.round(parseFloat(vis.states[hue + '.val'] || 0)),

 Math.round(parseFloat(vis.states[sat + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[bri + '.val'] || 0) \* factor)

 ));

 }, 10);

 };

 vis.states.bind(hue + '.val', hueChanged);

 vis.states.bind(sat + '.val', hueChanged);

 vis.states.bind(bri + '.val', hueChanged);

 hueChanged();

 }

 if (data['red-oid'] && data['red-oid'] !== 'nothing\_selected' &&

 data['green-oid'] && data['green-oid'] !== 'nothing\_selected' &&

 data['blue-oid'] && data['blue-oid'] !== 'nothing\_selected') {

 var r = data['red-oid'];

 var g = data['green-oid'];

 var b = data['blue-oid'];

 handler.push(function (hex) {

 var color = vis.binds.colorpicker.hex2rgb(hex);

 vis.setValue(r, parseFloat((color.r / factor).toFixed(decimal)));

 vis.setValue(g, parseFloat((color.g / factor).toFixed(decimal)));

 vis.setValue(b, parseFloat((color.b / factor).toFixed(decimal)));

 });

 var timer = null;

 var rgbChanged = function () {

 if (timer) clearTimeout(timer);

 timer = setTimeout(function () {

 timer = null;

 ft.setColor(vis.binds.colorpicker.rgb2hex(

 Math.round(parseFloat(vis.states[r + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[g + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[b + '.val'] || 0) \* factor)

 ));

 }, 10);

 };

 vis.states.bind(r + '.val', rgbChanged);

 vis.states.bind(g + '.val', rgbChanged);

 vis.states.bind(b + '.val', rgbChanged);

 rgbChanged();

 }

 if (data['rgb-oid'] && data['rgb-oid'] !== 'nothing\_selected') {

 var rgb = data['rgb-oid'];

 handler.push(function (hex) {

 vis.setValue(rgb, hex);

 });

 var \_rgbChanged = function () {

 ft.setColor(vis.states[rgb + '.val'] || '');

 };

 vis.states.bind(rgb + '.val', \_rgbChanged);

 \_rgbChanged();

 }

 },

 spectrum: function (el, data) {

 data = data.\_data || data;

 var $this = $(el);

 var handler = [];

 var factor = parseFloat(data.factor) || 1;

 var decimal = parseInt(data.decimal || 0, 10) || 0;

 $this.spectrum({

 preferredFormat: 'rgb'

 });

 $this.change(function() {

 for (var h = 0; h < handler.length; h++) {

 handler[h]();

 }

 });

 if (data['hue-oid'] && data['hue-oid'] !== 'nothing\_selected' &&

 data['sat-oid'] && data['sat-oid'] !== 'nothing\_selected' &&

 data['bri-oid'] && data['bri-oid'] !== 'nothing\_selected') {

 var hue = data['hue-oid'];

 var sat = data['sat-oid'];

 var bri = data['bri-oid'];

 handler.push(function () {

 var hsl = $this.spectrum('get').toHsl();

 vis.setValue(hue, parseFloat((hsl.h).toFixed(decimal)));

 vis.setValue(sat, parseFloat((hsl.s \* factor).toFixed(decimal)));

 vis.setValue(bri, parseFloat((hsl.l \* factor).toFixed(decimal)));

 });

 var htimer = null;

 var hueChanged = function () {

 if (htimer) clearTimeout(htimer);

 htimer = setTimeout(function () {

 htimer = null;

 $this.spectrum('set', {

 h: Math.round(parseFloat(vis.states[hue + '.val'] || 0)),

 s: Math.round(parseFloat(vis.states[sat + '.val'] || 0) / factor),

 l: Math.round(parseFloat(vis.states[bri + '.val'] || 0) / factor)

 });

 }, 10);

 };

 vis.states.bind(hue + '.val', hueChanged);

 vis.states.bind(sat + '.val', hueChanged);

 vis.states.bind(bri + '.val', hueChanged);

 hueChanged();

 }

 if (data['red-oid'] && data['red-oid'] !== 'nothing\_selected' &&

 data['green-oid'] && data['green-oid'] !== 'nothing\_selected' &&

 data['blue-oid'] && data['blue-oid'] !== 'nothing\_selected') {

 var r = data['red-oid'];

 var g = data['green-oid'];

 var b = data['blue-oid'];

 handler.push(function () {

 var color = $this.spectrum('get').toRgb();

 vis.setValue(r, parseFloat((color.r / factor).toFixed(decimal)));

 vis.setValue(g, parseFloat((color.g / factor).toFixed(decimal)));

 vis.setValue(b, parseFloat((color.b / factor).toFixed(decimal)));

 });

 var timer = null;

 var rgbChanged = function () {

 if (timer) clearTimeout(timer);

 timer = setTimeout(function () {

 timer = null;

 $this.spectrum('set', {

 r: Math.round(parseFloat(vis.states[r + '.val'] || 0) \* factor),

 g: Math.round(parseFloat(vis.states[g + '.val'] || 0) \* factor),

 b: Math.round(parseFloat(vis.states[b + '.val'] || 0) \* factor)

 });

 }, 10);

 };

 vis.states.bind(r + '.val', rgbChanged);

 vis.states.bind(g + '.val', rgbChanged);

 vis.states.bind(b + '.val', rgbChanged);

 rgbChanged();

 }

 if (data['rgb-oid'] && data['rgb-oid'] !== 'nothing\_selected') {

 var rgb = data['rgb-oid'];

 handler.push(function () {

 vis.setValue(rgb, $this.spectrum('get').toHexString());

 });

 var \_rgbChanged = function () {

 $this.spectrum('set', vis.states[rgb + '.val'] || '');

 };

 vis.states.bind(rgb + '.val', \_rgbChanged);

 \_rgbChanged();

 }

 },

 spectrumHomematic: function (el, data) {

 data = data.\_data || data;

 var $this = $(el);

 var handler = [];

 $this.spectrum({

 homematic: true,

 preferredFormat: "convert360To200",

 showInput: false,

 color: "hsv(360,0%,100%)",

 showPalette: true,

 palette: ["white"]

 });

 $this.change(function() {

 for (var h = 0; h < handler.length; h++) {

 handler[h]();

 }

 });

 if (data['color-oid'] && data['color-oid'] !== 'nothing\_selected') {

 var color = data['color-oid'];

 handler.push(function () {

 vis.setValue(color, $this.spectrum('get').toHMString());

 });

 var \_colorChanged = function () {

 var currentColor = vis.states[color + '.val'] === 0 ? 0 : vis.states[color + '.val'] || 200;

 var saturation = "100%";

 if (currentColor >= 200) {

 saturation = "0%";

 currentColor = 200;

 }

 var hsvVal = Math.round(currentColor / 200 \* 360);

 $this.spectrum('set',"hsv(" + hsvVal + "," + saturation + ",100%)");

 };

 vis.states.bind(color + '.val', \_colorChanged);

 \_colorChanged();

 }

 },

 jscolorcie: function (el, data) {

 data = data.\_data ? JSON.parse(JSON.stringify(data.\_data)) : data;

 data.pickerWidth = parseInt(data.pickerWidth) || 100;

 data.pickerHeight = parseInt(data.pickerHeight) || 100;

 data.gamut = typeof data.gamut === 'string' ? data.gamut : 'default';

 var gamutZoom = true;

 data.transitionTime = parseInt(data.transitionTime) || 0;

 data.buttonName = typeof data.buttonName === 'string' ? data.buttonName : '';

 data.closeButton = typeof data.closeButton === 'string' ? data.closeButton : '';

 var factor = parseFloat(data.factor) || 1;

 var decimal = parseInt(data.decimal || 0, 10) || 0;

 var $this = $(el);

 $this.html(data.buttonName);

 var handler = [];

 var picker = new jscolor($this[0], {

 valueElement: false,

 onFineChange: function () {

 for (var h = 0; h < handler.length; h++) {

 handler[h]();

 }

 },

 mode: 'CIE',

 gamut: data.gamut,

 gamutZoom: gamutZoom,

 width: data.pickerWidth,

 height: data.pickerHeight,

 closable: !!data.closeButton,

 closeText: data.closeButton,

 padding: 0,

 shadow: false,

 borderWidth: 0,

 backgroundColor: data.pickerBackground,

 insetColor: '#000'

 });

 if (data['command-oid'] && data['command-oid'] !== 'nothing\_selected' &&

 data['xy-oid'] && data['xy-oid'] !== 'nothing\_selected' &&

 data['level-oid'] && data['level-oid'] !== 'nothing\_selected') {

 var cmd\_id = data['command-oid'];

 var xy\_id = data['xy-oid'];

 var level\_id = data['level-oid'];

 handler.push(function () {

 if (cmd\_id !== 'nothing\_selected') {

 // var cmd = '{"transitiontime":' + data.transitionTime + ',"xy":"' + picker.xy[0] + ',' + picker.xy[1] + '",' + '"level":' + Math.round(picker.xy[2] \* 100) + '}';

 //var cmd = picker.xy[0] + ',' + picker.xy[1];

 vis.setValue(xy\_id, picker.xy[0] + ',' + picker.xy[1]);

 vis.setValue(level\_id, Math.round(picker.xy[2] \* 254));

 }

 });

 var onChange = function () {

 var xy = vis.states[xy\_id + '.val'].toString();

 //var level = vis.states[level\_id + '.val'] || 0;

 var level = vis.states[level\_id + '.val'] || 0;

 xy = xy.split(',');

 if (xy.length === 2) {

 picker.fromXY(parseFloat(xy[0]), parseFloat(xy[1]), parseInt(level) / 254);

 }

 };

 vis.states.bind(xy\_id + '.val', onChange);

 vis.states.bind(level\_id + '.val', onChange);

 onChange();

 }

 if (data['hue-oid'] && data['hue-oid'] !== 'nothing\_selected' &&

 data['sat-oid'] && data['sat-oid'] !== 'nothing\_selected' &&

 data['bri-oid'] && data['bri-oid'] !== 'nothing\_selected') {

 var hue = data['hue-oid'];

 var sat = data['sat-oid'];

 var bri = data['bri-oid'];

 handler.push(function () {

 vis.setValue(hue, parseFloat(picker.hsv[0]).toFixed(decimal));

 vis.setValue(sat, parseFloat((picker.hsv[1] / factor).toFixed(decimal)));

 vis.setValue(bri, parseFloat((picker.hsv[2] / factor).toFixed(decimal)));

 });

 var htimer = null;

 var hueChanged = function () {

 if (htimer) clearTimeout(htimer);

 htimer = setTimeout(function () {

 htimer = null;

 picker.fromHSV(

 Math.round(parseFloat(vis.states[hue + '.val'] || 0)),

 Math.round(parseFloat(vis.states[sat + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[bri + '.val'] || 0) \* factor)

 );

 }, 10);

 };

 vis.states.bind(hue + '.val', hueChanged);

 vis.states.bind(sat + '.val', hueChanged);

 vis.states.bind(bri + '.val', hueChanged);

 hueChanged();

 }

 if (data['red-oid'] && data['red-oid'] !== 'nothing\_selected' &&

 data['green-oid'] && data['green-oid'] !== 'nothing\_selected' &&

 data['blue-oid'] && data['blue-oid'] !== 'nothing\_selected') {

 var r = data['red-oid'];

 var g = data['green-oid'];

 var b = data['blue-oid'];

 handler.push(function () {

 vis.setValue(r, parseFloat((picker.rgb[0] / factor).toFixed(decimal)));

 vis.setValue(g, parseFloat((picker.rgb[1] / factor).toFixed(decimal)));

 vis.setValue(b, parseFloat((picker.rgb[2] / factor).toFixed(decimal)));

 });

 var timer = null;

 var rgbChanged = function () {

 if (timer) clearTimeout(timer);

 timer = setTimeout(function () {

 timer = null;

 picker.fromRGB(

 Math.round(parseFloat(vis.states[r + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[g + '.val'] || 0) \* factor),

 Math.round(parseFloat(vis.states[b + '.val'] || 0) \* factor)

 );

 }, 10);

 };

 vis.states.bind(r + '.val', rgbChanged);

 vis.states.bind(g + '.val', rgbChanged);

 vis.states.bind(b + '.val', rgbChanged);

 rgbChanged();

 }

 if (data['rgb-oid'] && data['rgb-oid'] !== 'nothing\_selected') {

 var rgb = data['rgb-oid'];

 handler.push(function () {

 vis.setValue(rgb, picker.toHEXString());

 });

 var \_rgbChanged = function () {

 picker.fromString(vis.states[rgb + '.val'] || '');

 };

 vis.states.bind(rgb + '.val', \_rgbChanged);

 \_rgbChanged();

 }

 },

 huepickerxy: function (el, data) {

 data = data.\_data ? JSON.parse(JSON.stringify(data.\_data)) : data;

 data.gamut = typeof data.gamut === 'string' ? data.gamut : 'default';

 data.transitionTime = parseInt(data.transitionTime) || 0;

 var gamutZoom = true;

 var $this = $(el);

 //build canvas

 var canvas = document.createElement('canvas');

 var img = document.createElement("img");

 canvas.appendChild(img);

 function redraw() {

 canvas.width = $this.width();

 canvas.height = $this.height();

 var ctx = canvas.getContext('2d');

 ctx.clearRect(0, 0, canvas.width, canvas.height);

 ctx.fillStyle = "#FFFFFF";

 ctx.fillRect(0, 0, canvas.width, canvas.height); // Need to Fill Canvas otherwise its unable to be modified...

 var XYImagedata = ctx.getImageData(0, 0, canvas.width, canvas.height);

 for (var x = 0; x < canvas.width; x++) { // Every Pixel on ScanLine

 for (var y = 0; y < canvas.height; y++) { // Every ScanLine

 var Index = (x + y \* canvas.width) \* 4; // Index of Pixel

 var xy;

 if (gamutZoom){

 xy = huepi.HelperZoomXYforModel(x / canvas.width, 1 - (y / canvas.height), data.gamut);

 }else {

 xy = [

 x / canvas.width,

 1 - (y / canvas.height)

 ]

 }

 var XYColor = huepi.HelperXYtoRGBforModel(xy[0], xy[1], 1, data.gamut);

 var TestColor = huepi.HelperXYtoRGB(xy[0], xy[1], 1);

 if (TestColor.Red !== XYColor.Red || TestColor.Green !== XYColor.Green || TestColor.Blue !== XYColor.Blue){

 XYImagedata.data[Index] = XYColor.Red \* 240;

 XYImagedata.data[Index + 1] = XYColor.Green \* 240;

 XYImagedata.data[Index + 2] = XYColor.Blue \* 240;

 }else {

 XYImagedata.data[Index] = XYColor.Red \* 255;

 XYImagedata.data[Index + 1] = XYColor.Green \* 255;

 XYImagedata.data[Index + 2] = XYColor.Blue \* 255;

 }

 }

 }

 ctx.putImageData(XYImagedata, 0, 0);

 img.src = canvas.toDataURL("image/png");

 var xyGet = vis.states[xy\_id + '.val'] || '0.5,0.5';

 if (typeof xyGet !== 'string') xyGet = '0.5,0.5';

 xyGet = xyGet.split(',');

 xyGet = huepi.HelperUnzoomXYforModel(xyGet[0], xyGet[1], data.gamut);

 var centerX = xyGet[0] \* canvas.width;

 var centerY = canvas.height - (xyGet[1] \* canvas.height);

 var radius = 6;

 ctx.beginPath();

 ctx.arc(centerX, centerY, radius, 0, 2 \* Math.PI, false);

 ctx.lineWidth = 2;

 ctx.strokeStyle = '#000000';

 ctx.stroke();

 $this.html(canvas);

 }

 $this.parent().parent().resize(function() {

 redraw();

 });

 $this.show(function() {

 redraw();

 });

 if (data['command-oid'] && data['command-oid'] !== 'nothing\_selected' &&

 data['xy-oid'] && data['xy-oid'] !== 'nothing\_selected') {

 var cmd\_id = data['command-oid'];

 var xy\_id = data['xy-oid'];

 $this.click(function (e) {

 var xy = [e.offsetX / canvas.width, 1 - (e.offsetY / canvas.height)];

 if (gamutZoom) {

 xy = huepi.HelperZoomXYforModel(xy[0], xy[1], data.gamut);

 }

 var cmd = '{"transitiontime":' + data.transitionTime + ',"xy":"' + xy[0] + ',' + xy[1] + '"}';

 vis.setValue(cmd\_id, cmd);

 });

 var onChange = function () {

 redraw();

 };

 vis.states.bind(xy\_id + '.val', onChange);

 }

 },

 hueindicatorxy: function (el, data) {

 data = data.\_data ? JSON.parse(JSON.stringify(data.\_data)) : data;

 data.gamut = typeof data.gamut === 'string' ? data.gamut : 'default';

 var $this = $(el);

 if (data['xy-oid'] && data['xy-oid'] !== 'nothing\_selected') {

 var xy\_id = data['xy-oid'];

 var onChange = function () {

 var xyGet = vis.states[xy\_id + '.val'] || '0.5,0.5';

 if (typeof xyGet !== 'string') xyGet = '0.5,0.5';

 xyGet = xyGet.split(',');

 var rgb = huepi.HelperXYtoRGBforModel(xyGet[0], xyGet[1], 1, data.gamut);

 $this.css("background-color", huepi.HelperRGBToHEX(rgb));

 };

 vis.states.bind(xy\_id + '.val', onChange);

 onChange();

 }

 },

 huepickerct: function (el, data) {

 data = data.\_data ? JSON.parse(JSON.stringify(data.\_data)) : data;

 data.transitionTime = parseInt(data.transitionTime) || 0;

 var ctMin = 2000;

 var ctMax = 6500;

 var $this = $(el);

 //build canvas

 var canvas = document.createElement('canvas');

 var img = document.createElement("img");

 canvas.appendChild(img);

 function redraw() {

 canvas.width = $this.width();

 canvas.height = $this.height();

 var ctx = canvas.getContext('2d');

 ctx.clearRect(0, 0, canvas.width, canvas.height);

 ctx.fillStyle = "#FFFFFF";

 ctx.fillRect(0, 0, canvas.width, canvas.height); // Need to Fill Canvas otherwise its unable to be modified...

 var XYImagedata = ctx.getImageData(0, 0, canvas.width, canvas.height);

 for (var x = 0; x < canvas.width; x++) { // Every Pixel on ScanLine

 var ctColor = huepi.HelperColortemperaturetoRGB(ctMin + (x / canvas.width \* (ctMax - ctMin)));

 for (var y = 0; y < canvas.height; y++) { // Every ScanLine

 var Index = (x + y \* canvas.width) \* 4; // Index of Pixel

 XYImagedata.data[Index] = ctColor.Red \* 255;

 XYImagedata.data[Index + 1] = ctColor.Green \* 255;

 XYImagedata.data[Index + 2] = ctColor.Blue \* 255;

 }

 }

 ctx.putImageData(XYImagedata, 0, 0);

 img.src = canvas.toDataURL("image/png");

 var ctGet = parseInt(vis.states[ct\_id + '.val']) || 153;

 ctGet = Math.max(2000, Math.min(6500, Math.round(Math.pow(10,6)/ctGet)));

 ctGet = canvas.width / ((ctMax-ctMin) / (ctGet-ctMin));

 ctx.beginPath();

 ctx.lineWidth = 2;

 ctx.strokeStyle = '#000000';

 ctx.moveTo(ctGet, 0);

 ctx.lineTo(ctGet, canvas.height);

 ctx.stroke();

 $this.html(canvas);

 }

 $this.parent().parent().resize(function() {

 redraw();

 });

 $this.show(function() {

 redraw();

 });

 if (data['command-oid'] && data['command-oid'] !== 'nothing\_selected' &&

 data['ct-oid'] && data['ct-oid'] !== 'nothing\_selected') {

 var cmd\_id = data['command-oid'];

 var ct\_id = data['ct-oid'];

 $this.click(function (e) {

 var ct = e.offsetX / canvas.width;

 ct = Math.round(ctMin + (ctMax-ctMin) \* ct);

 ct = Math.max(153, Math.min(500, Math.round(Math.pow(10,6)/ct)));

 var cmd = '{"transitiontime":' + data.transitionTime + ',"ct":"' + ct + '"}';

 vis.setValue(cmd\_id, cmd);

 });

 var onChange = function () {

 redraw();

 };

 vis.states.bind(ct\_id + '.val', onChange);

 }

 },

 hueindicatorct: function (el, data) {

 data = data.\_data ? JSON.parse(JSON.stringify(data.\_data)) : data;

 var $this = $(el);

 if (data['ct-oid'] && data['ct-oid'] !== 'nothing\_selected') {

 var ct\_id = data['ct-oid'];

 var onChange = function () {

 var ctGet = parseInt(vis.states[ct\_id + '.val']) || 153;

 ctGet = Math.max(2000, Math.min(6500, Math.round(Math.pow(10,6)/ctGet)));

 var rgb = huepi.HelperColortemperaturetoRGB(ctGet);

 $this.css("background-color", huepi.HelperRGBToHEX(rgb));

 };

 vis.states.bind(ct\_id + '.val', onChange);

 onChange();

 }

 }

 };

 if (vis.editMode) {

 vis.binds.colorpicker.changedHUE = function (widgetID, view, newId, fields) {

 var cmd = vis.objects[newId];

 var changed = [];

 // If it is real object and state

 if (cmd && cmd.common && cmd.type === 'state') {

 var light = vis.objects[newId.split('.').slice(0,-1).join('.')];

 var xy = vis.objects[newId.split('.').slice(0,-1).join('.') + '.xy'];

 var level = vis.objects[newId.split('.').slice(0,-1).join('.') + '.level'];

 if (xy && xy.type === 'state' && !vis.views[view].widgets[widgetID].data['xy-oid']) {

 changed.push('xy-oid');

 vis.views[view].widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 vis.widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 }

 if (level && level.type === 'state' && !vis.views[view].widgets[widgetID].data['level-oid']) {

 changed.push('level-oid');

 vis.views[view].widgets[widgetID].data['level-oid'] = newId.split('.').slice(0,-1).join('.') + '.level';

 vis.widgets[widgetID].data['level-oid'] = newId.split('.').slice(0,-1).join('.') + '.level';

 }

 if (light.native.modelid !== undefined && !vis.views[view].widgets[widgetID].data.gamut) {

 changed.push('gamut');

 vis.views[view].widgets[widgetID].data.gamut = light.native.modelid;

 vis.widgets[widgetID].data.gamut = light.native.modelid;

 }

 }

 return changed;

 };

 vis.binds.colorpicker.changedHUEPickerXY = function (widgetID, view, newId, fields) {

 var cmd = vis.objects[newId];

 var changed = [];

 // If it is real object and state

 if (cmd && cmd.common && cmd.type === 'state') {

 var light = vis.objects[newId.split('.').slice(0,-1).join('.')];

 var xy = vis.objects[newId.split('.').slice(0,-1).join('.') + '.xy'];

 if (xy && xy.type === 'state' && !vis.views[view].widgets[widgetID].data['xy-oid']) {

 changed.push('xy-oid');

 vis.views[view].widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 vis.widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 }

 if (light.native.modelid !== undefined && !vis.views[view].widgets[widgetID].data.gamut) {

 changed.push('gamut');

 vis.views[view].widgets[widgetID].data.gamut = light.native.modelid;

 vis.widgets[widgetID].data.gamut = light.native.modelid;

 }

 }

 return changed;

 };

 vis.binds.colorpicker.changedHUEPickerCT = function (widgetID, view, newId, fields) {

 var cmd = vis.objects[newId];

 var changed = [];

 // If it is real object and state

 if (cmd && cmd.common && cmd.type === 'state') {

 var light = vis.objects[newId.split('.').slice(0,-1).join('.')];

 var ct = vis.objects[newId.split('.').slice(0,-1).join('.') + '.ct'];

 if (ct && ct.type === 'state' && !vis.views[view].widgets[widgetID].data['ct-oid']) {

 changed.push('ct-oid');

 vis.views[view].widgets[widgetID].data['ct-oid'] = newId.split('.').slice(0,-1).join('.') + '.ct';

 vis.widgets[widgetID].data['ct-oid'] = newId.split('.').slice(0,-1).join('.') + '.ct';

 }

 }

 return changed;

 };

 vis.binds.colorpicker.changedHUEIndicatorXY = function (widgetID, view, newId, fields) {

 var xy = vis.objects[newId];

 var changed = [];

 // If it is real object and state

 if (xy && xy.common && xy.type === 'state') {

 var light = vis.objects[newId.split('.').slice(0,-1).join('.')];

 if (xy && xy.type === 'state' && !vis.views[view].widgets[widgetID].data['xy-oid']) {

 changed.push('xy-oid');

 vis.views[view].widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 vis.widgets[widgetID].data['xy-oid'] = newId.split('.').slice(0,-1).join('.') + '.xy';

 }

 if (light.native.modelid !== undefined && !vis.views[view].widgets[widgetID].data.gamut) {

 changed.push('gamut');

 vis.views[view].widgets[widgetID].data.gamut = light.native.modelid;

 vis.widgets[widgetID].data.gamut = light.native.modelid;

 }

 }

 return changed;

 };

 }

 vis.binds.colorpicker.showVersion();

</script>

<script id="tplRGBSpectrum"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplRGBSpectrum.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="RGB spectrum"

 data-vis-attrs="rgb-oid/id;"

 data-vis-attrs0="group.rgb;red-oid/id;green-oid/id;blue-oid/id;"

 data-vis-attrs1="group.hue;hue-oid/id;sat-oid/id;bri-oid/id;"

 data-vis-attrs2="group.extra;factor[1];decimal[0]/slider,0,5,1;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <input style="width: 100%; height: 100%" <%= (el) -> vis.binds.colorpicker.spectrum(el, data) %> />

 </div>

 </div>

</script>

<script id="tplSpectrumHomematic"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplRGBSpectrum.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="Homematic spectrum"

 data-vis-attrs="color-oid/id;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <input style="width: 100%; height: 100%" <%= (el) -> vis.binds.colorpicker.spectrumHomematic(el, data) %> />

 </div>

 </div>

</script>

<script id="tplRGBFarbtastic"

 class="vis-tpl"

 type="text/ejs"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplRGBFarbtastic.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="farbtastic"

 data-vis-attrs="rgb-oid/id;"

 data-vis-attrs0="group.rgb;red-oid/id;green-oid/id;blue-oid/id;"

 data-vis-attrs1="group.hue;hue-oid/id;sat-oid/id;bri-oid/id;"

 data-vis-attrs2="group.extra;factor[1];decimal[0]/slider,0,5,1;"

 >

 <div class="vis-widget <%== this.data.attr('class') %>" style="width: 196px; height: 196px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <div></div>

 <input <%= (el) -> vis.binds.colorpicker.farbtastic(el, data) %> />

 </div>

 </div>

</script>

<script id="tplHUEjscolor"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplHUEjscolor.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="Philips HUE"

 data-vis-attrs="command-oid/id/changedHUE;xy-oid/id;level-oid/id;gamut;transitionTime[4]/number,0,30000,1;pickerWidth[100]/number,0,500,1;pickerHeight[100]/number,0,500,1;pickerBackground/color;buttonName[HUE]/html;closeButton[close];"

 data-vis-attrs0="group.rgb;red-oid/id;green-oid/id;blue-oid/id;divisor[1];decimal[0]/slider,0,5,1;rgb-oid/id;"

 data-vis-attrs1="group.hue;hue-oid/id;sat-oid/id;bri-oid/id;"

 >

 <div class="vis-widget <%== this.data.attr('class') %>" style="width:100px;height:100px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body" style="display:table;">

 <div class="ui-button ui-state-default ui-corner-all" style="width:100%;height:100%;display:table-cell;" <%= (el) -> vis.binds.colorpicker.jscolorcie(el, data) %> ></div>

 </div>

 </div>

</script>

<script id="tplHUEPickerXY"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplHUEPickerXY.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="Philips HUE XY-Picker"

 data-vis-attrs="command-oid/id/changedHUEPickerXY;xy-oid/id;gamut;transitionTime[4]/number,0,30000,1;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="width:100px;height:100px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <div class="hue-canvas" style="width:100%;height:100%;cursor:crosshair;" <%= (el) -> vis.binds.colorpicker.huepickerxy(el, data) %> ></div>

 </div>

 </div>

</script>

<script id="tplHUEIndicatorXY"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplHUEIndicatorXY.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="value,color"

 data-vis-name="Philips HUE XY-Indicator"

 data-vis-attrs="xy-oid/id/changedHUEIndicatorXY;gamut;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="width:100px;height:100px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <div class="hue-indicator-xy" style="width:100%;height:100%;" <%= (el) -> vis.binds.colorpicker.hueindicatorxy(el, data) %> ></div>

 </div>

 </div>

</script>

<script id="tplHUEPickerCT"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplHUEPickerCT.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="Philips HUE CT-Picker"

 data-vis-attrs="command-oid/id/changedHUEPickerCT;ct-oid/id;transitionTime[4]/number,0,30000,1;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="width:100px;height:100px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <div class="hue-canvas-ct" style="width:100%;height:100%;cursor:crosshair;" <%= (el) -> vis.binds.colorpicker.huepickerct(el, data) %> ></div>

 </div>

 </div>

</script>

<script id="tplHUEIndicatorCT"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplHUEIndicatorCT.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="value,color"

 data-vis-name="Philips HUE CT-Indicator"

 data-vis-attrs="ct-oid/id;"

>

 <div class="vis-widget <%== this.data.attr('class') %>" style="width:100px;height:100px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body">

 <div class="hue-indicator-xy" style="width:100%;height:100%;" <%= (el) -> vis.binds.colorpicker.hueindicatorct(el, data) %> ></div>

 </div>

 </div>

</script>

<script id="tplJscolor"

 type="text/ejs"

 class="vis-tpl"

 data-vis-prev='<img src="widgets/colorpicker/img/Prev\_tplJscolor.png"></img>'

 data-vis-set="colorpicker"

 data-vis-type="ctrl,color"

 data-vis-name="RGBColor"

 data-vis-attrs="title[RGB:&nbsp];closeText[X];rgb-oid/id;"

 data-vis-attrs0="group.rgb;red-oid/id;green-oid/id;blue-oid/id;"

 data-vis-attrs1="group.hue;hue-oid/id;sat-oid/id;bri-oid/id;"

 data-vis-attrs2="group.extra;factor[1];decimal[0]/slider,0,5,1;"

 >

 <div class="vis-widget <%== this.data.attr('class') %>" style="width: 96px; height: 50px;" id="<%= this.data.attr('wid') %>">

 <div class="vis-widget-body" style="display: inline-block">

 <table style="border: 0; border-spacing: 0; width: 100%; height: 100%" class="vis-no-spaces"><tr class="vis-no-spaces">

 <td class="vis-no-spaces"><%== this.data.attr('title') %></td>

 <td class="vis-no-spaces"><input style="width: calc(100% - 5px); height: 100%" readonly <%= (el) -> vis.binds.colorpicker.jscolor(el, this.data) %> /></td>

 </div>

 </div>

</script>