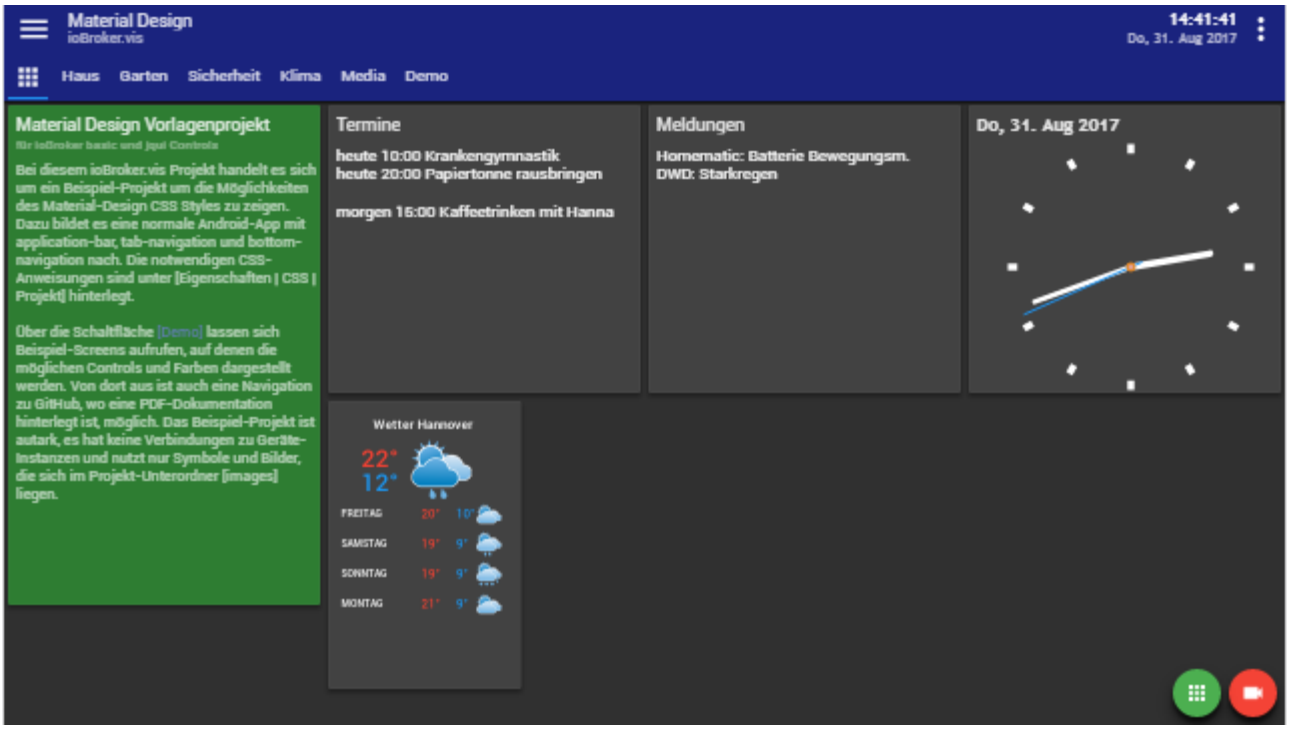
# ioBroker.vis

## im

# Material Design Style

# Inhaltsverzeichnis

# 1.Anleitung zur Material Design CSS




Grundsätzlich wird die Visualisierung mit den bekannten basic und jqui Controls entworfen. Diese erhalten lediglich CSS-Klassenzuweisungen um im Browser im Material Design Style gerendert zu werden. Es gibt nicht für alle Controls entsprechende CSS-Klassen.

Die CSS-Anweisungen müssen dem Projekt auf der CSS-Registerseite unter „Projekt" zugewiesen werden. Sie stehen dann sowohl zur Anzeige im Editor als auch zur Laufzeit zur Verfügung. Sie sind so gestaltet, dass sie sich im Editor nur auf das View-Editfenster auswirken und nicht auf den Rest des Editors. Dazu ist es auch notwendig, ein wenig JS Code auf der Skripte-Registerseite einzufügen (aus einem der Beispiel-herausziehen).

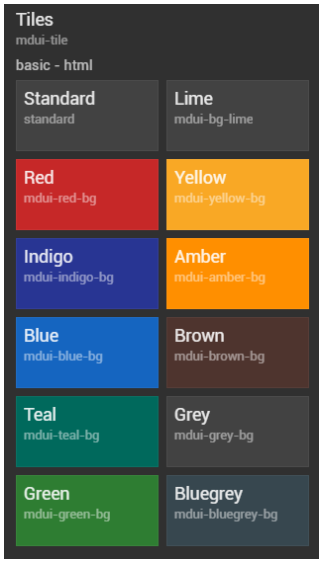
Oder, noch einfacher, eines der am Ende genannten Demo-Projekte als Basis verwenden.

In der folgenden Beschreibung sind notwendige Einstellungen der basic- und jqui-Controls sind mit "MUSS" gekennzeichnet, optionale mit "KANN".

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | ... | Der Wert MUSS gesetzt werden |
| | CSS Klasse | ... | Optional |

## 2.Farben

Als Vorgabe ist der Style insgesamt in schwarz-weiß gehalten, es besteht aber die Möglichkeit über CSS Angaben die Darstellung der Text, Buttons usw. auch farbig zu erhalten. Hierzu stehen spezielle mdui-(color) (Schriftfarben), mdui-(color)-bg (Hintergrundfarben) und mdui-(color)-acc (Akzentfarben) CSS Klassen zur Verfügung. Verwendet werden die durch das Material Design vorgegebenen Farben.




Als Farben (color) stehen zur Verfügung:

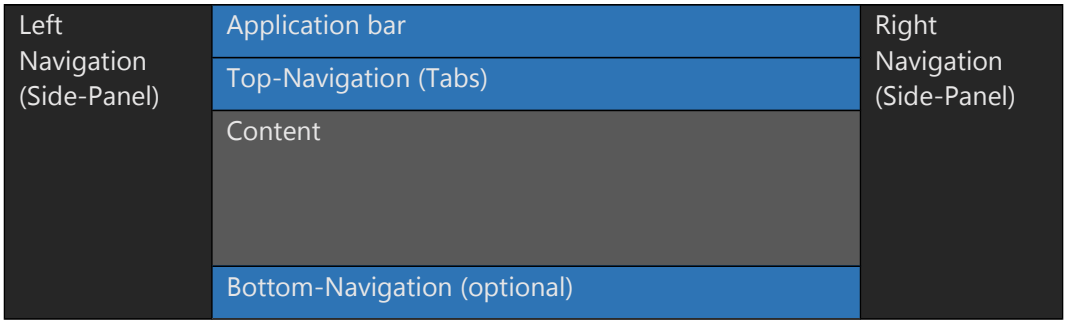
**red, indigo, blue, teal, green, lime, yellow, amber, brown, grey, bluegrey**

**TIPP**
Farben ziehen die Aufmerksamkeit des Betrachters auf sich. Damit sie diese Aufgabe gut erledigen können, dürfen sie nicht in einer Vielzahl von anderen Farben untergehen. Das Grundlayout sollte also bewusst einfarbig gestaltet sein, damit die Signalfarben dann umso deutlicher auffallen.

## 3.Layout

Das normale Layout einer Seite (Page) besteht aus sechs Bereichen, wobei bis auf den Content-Bereich alle optional sind.

Aufbau einer Seite:




In der top-navigation werden die Buttons für die Navigation durch die einzelnen Seiten (Views) angezeigt. Je nach Navigationstiefe kann die top-navigation angepasst werden, in dem in den Seiten einfach andere top-navigations eingebunden werden. So besitzt z.B. die Hauptseite eine andere top-navigation als z.B. die Hausseite, in welcher die Räume aufgeführt werden. Views, die als top-navigation genutzt werden, sollten mit "tnav" beginnen, z.B. "tnavMain", "tnavHaus".

Im Content wird später die View mit dem eigentlichen Inhalt der Seite eingeblendet.

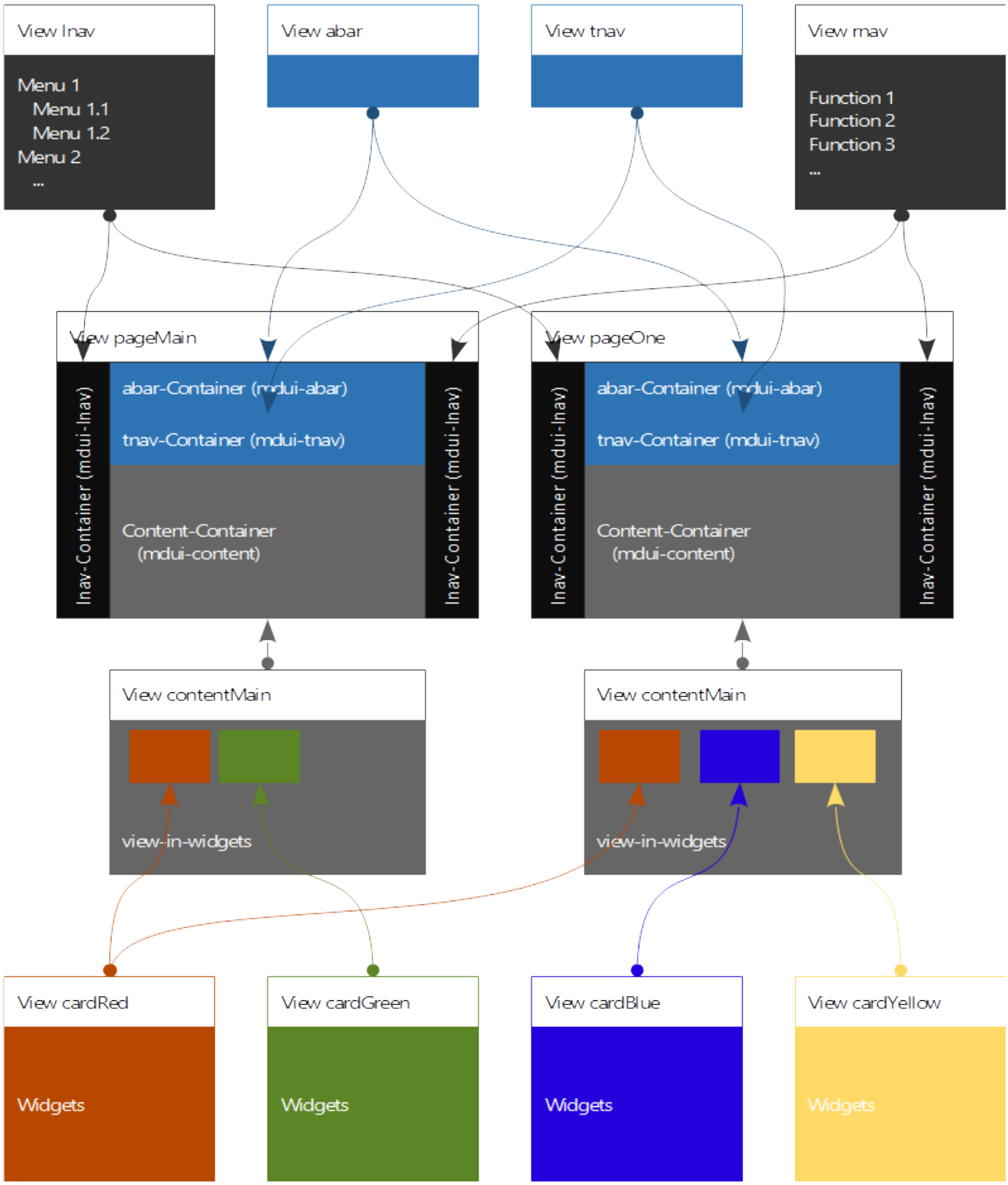
Die bottom-navigation kann nun entweder wie die application-bar oder wie die tab-navigation genutzt werden. Views, die als bottom-navigation genutzt werden, sollten mit "bnav" beginnen, z.B. "bnavMain". Bottom-navigations werden selten verwendet.

Die left-navigation stellt das Application-Menü dar, welches beim Betätigen der Menü-Schaltfläche von links eingeblendet wird. Hier können z.B. noch einmal die Links zu den einzelnen Seiten aufgeführt werden. Views, die als left-navigation genutzt werden, sollten mit "lnav" beginnen, z.B. "lnavMain". Normalerweise gibt es genau eine left-navigation.

Die right-navigation stellt das Funktions-Menü dar, welches beim Betätigen der Funktion-Schaltfläche von rechts eingeblendet wird. Hier können z.B. kontextabhängige Funktionen aufgeführt werden. Views, die als right-navigation genutzt werden, sollten mit "rnav" beginnen, z.B. "rnavMain". Auf unterschiedlichen Seiten können verschiedene right-navigations verwendet werden.

**Wichtig**
Alle genannten Komponenten werden nicht direkt in den Seiten eingefügt, sondern sind jeweils eigene Views, die über das Widget ***basic-view in widget Container*** in die Seiten eingebunden werden! Näheres bei der jeweiligen Beschreibung.

# 4.Layout-Zeichnung



# 5.Application bar

## 5.1.Application bar View

Hierbei handelt es sich um eine View, die später in den Application bar Containern verwendet wird. Es werden Buttons und Texte untergebracht, die permanent sichtbar sein sollen. Wie z.B. Menü- und Home-Buttons, Uhrzeit und Tagesanzeige. In der Regel hat ein Projekt genau eine application-bar, es sind aber auch mehrere möglich. Views, die als application-bar genutzt werden, sollten mit "abar" beginnen, z.B. "abarMain". Die Höhe sollte 48px betragen.

Die View sollte z.B. die Schaltflächen zum Öffnen der left- und right-navigation beinhalten, damit diese als solche erkannt werden, muss ihnen jeweils eine CSS-Klasse zugewiesen werden.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-lnavbutton | |

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-rnavbuttton | |

## 5.2.Application bar Container

Auf jeder Seite, auf denen eine Application bar dargestellt werden soll, ist ein Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Application bar-View angezeigt wird. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen, sie sollten alle eine Höhe von 48px haben.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | Width<br>Height | 100%<br>48px | Kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px |
| **Ja** | CSS Klasse | mdui-abar | |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |

# 6.Top-Navigation, Bottom-Navigation

## 6.1.Top/Bottom-Navigation View

Hierbei handelt es sich um Views, die später in den Top/Bottom-Navigation Containern verwendet werden. Normalerweise werden hier Buttons untergebracht, welcher der Navigation dienen. Views, die als Top/Bottom-Navigation genutzt werden, sollten mit "tnav" bzw. „bnav“ beginnen, z.B. "tnavMain". Die Höhe sollte 48px betragen.

Die in den Views verwendeten **Buttons** sollten die folgenden Einstellungen erhalten:

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height | auto<br>32px | |
| | CSS Klasse | mdui-flatbutton | Reine Textbuttons |
| | CSS Klasse | mdui-float<br>mdui-float-right | Damit werden die Buttons automatisch links (rechts) angeordnet. Insbesondere wenn man im Button-Text mit Breitenbedingungen arbeitet, ist dieses sehr sinnvoll. Da das floaten in Abhängigkeit der Erstellungsreihenfolge passiert, muss man evtl. über Kopieren/Einfügen die Buttons in die richtige Reihenfolge bringen – ioBroker hat m.W. dafür keine eigene Funktion. |
| | Text | „Buttontext“ | Wenn man hier Rücksicht auf ein responsive Design nehmen möchte, kann man den Text so gestalten, dass er bei Bildschirmauflösungen von max. 480px und mehr als 480px unterschiedlich dargestellt wird. Hierzu sind die CSS Klassen mdui-show480 und mdui-hide480 zu nutzen. Bsp: EGErdgeschoß |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |
| | CSS Klasse | mdui-(color) | Schriftfarbe |

**TIPP**
Um Controls am rechten Rand so zu verankern, dass sie mit der Screen-Breite skalieren, kann man bei der "left" Angabe "calc(100% - nnnpx)" angeben, mit nnn = Breite des Controls.

## 6.2.Top/Bottom-Navigation Container

Auf jeder Seite, auf denen eine solche Navigation dargestellt werden soll, sind entsprechende Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Top/Bottom-Navigation Views angezeigt

werden. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen, sie sollten alle eine Höhe von 48px haben.

**Top-Navigation:**

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height<br>Top<br>Left | 100%<br>48px<br>48px<br>0px | Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px |
| **Ja** | CSS Klasse | mdui-tnav | Top-Navigation |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |
| | CSS Klasse | mdui-(color)-acc | Akzentfarbe um den aktuellen Navigations-Button zu kennzeichnen |

**Bottom-Navigation:**

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height<br>Top<br>Left | 100%<br>48px<br>calc(100% - 48px)<br>0px | Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px<br>Top kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 672px |
| **Ja** | CSS Klasse | mdui-bnav | Top-Navigation |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |
| | CSS Klasse | mdui-(color)-acc | Akzentfarbe um den aktuellen Navigations-Button zu kennzeichnen |

# 7.Content

## 7.1.Content View

Hierbei handelt es sich den View, der später in den Content Containern verwendet wird und den echten Inhalt der Seiten bekommt. Zu jedem Seiten-View gibt es also genau einen Content-View. Views, die als Content genutzt werden, sollten mit "cont" beginnen, z.B. "contMain".

In den Content Views kann man entweder die Widgets direkt hinein setzen, oder, wenn man ein responsive Design haben möchten, fügt man nur Widget ***basic-view in widget Container*** ein, die als Platzhalter für die (Card-) Views dienen, welche dann die Widgets enthalten. Dieses ist notwendig, da ansonsten kein automatischen „floaten" des Inhalts möglich ist.

Diesen Widget ***basic-view in widget Container***: können zugewiesen werden:

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width | 312 px | Beim responsive Design sollte man immer „mobile first" im Blick haben, also die Breite der einzelnen Contaoner so wählen, dass sie auch noch auf den gewünschten Endgeräten komplett dargestellt werden können. Bei heutigen Smartphones, die eine Auflösung von 720x1280px haben, wären das 360px (wegen der doppelten Pixeldichte). |
| | CSS Klasse | mdui-float<br>mdui-float-right | Damit werden die Container automatisch links (rechts) angeordnet. Da das „floaten" in Abhängigkeit der Erstellungsreihenfolge passiert, muss man evtl. über Kopieren/Einfügen die Buttons in die richtige Reihenfolge bringen – ioBroker hat m.W. dafür keine eigene Funktion. |
| | CSS Klasse | mdui-card | Darstellung des Containers als „Card", also mit Schatten |
| | CSS Klasse | mdui-tile | Darstellung des Containers als „Tile" |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |

## 7.2.Content Container

Auf jeder Seite gibt es genau ein Widget ***basic-view in widget Container*** welches den Content View darstellen wird.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height | 100%<br>calc(100% - 2 * 48px) | Width kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px |

| | | | |
|---|---|---|---|
| | Top<br>Left | 96px<br>0px | Wenn mit Bottom-Navigation, dann ist Height: calc(100% - 3 * 48px) |
| **Ja** | CSS Klasse | mdui-content | |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |

# 8.Left-Navigation, Right-Navigation

## 8.1.Left/Right-Navigation View

Hierbei handelt es sich um Views, die später in den Left/Right-Navigation Containern verwendet werden. Normalerweise werden hier Buttons untergebracht, welcher der internen und externen Navigation dienen. Views, die als Left/Right-Navigation genutzt werden, sollten mit "lnav" bzw. „rnav“ beginnen, z.B. "lnavMain".

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width | max 288px | Da die Left/Right-Navigation Container später mit einer Breite von 288px angezeigt werden, sollte diese im View eingehalten werden |
| | Height | | Egal, da später im Left/Right-Navigation Container vertikal gescrollt wird |
| | Top<br>Left | 8px<br>8px | |

TIPP
Manchmal möchte man in den Sidebars (Left/Right-Navigation) ein komplettes Menü mit vielen Einträgen unterbringen. Um die Übersichtlichkeit zu wahren und um häufiges Scrollen zu vermeiden, kann man dort mit Baumstruktur-Menüs arbeiten. Bei diesen handelt es sich im Prinzip um 2-stufige Menüs, wobei die jeweils 2.Stufe sich erst öffnet, wenn auf der 1.Stufe eine Schaltfläche betätigt wurde.
Weiter im eigenen Kapitel „ 18. Baumstruktur-Menüs in der Sidebar“

## 8.2.Left/Right-Navigation Container

Auf jeder Seite, auf denen eine solche Navigation dargestellt werden soll, sind entsprechende Widget ***basic-view in widget Container*** einzufügen, in welchem dann die Left/Right-Navigation Views angezeigt werden. Den Containern (nicht den Views!) auf den Seiten wird dann die zugehörige CSS Klasse zugewiesen.

**Left-Navigation Container**

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height<br>Top<br>Left | | Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am linken Rand angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden. |
| **Ja** | CSS Klasse | mdui-lnav | Left-Navigation |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |

**Right-Navigation Container**

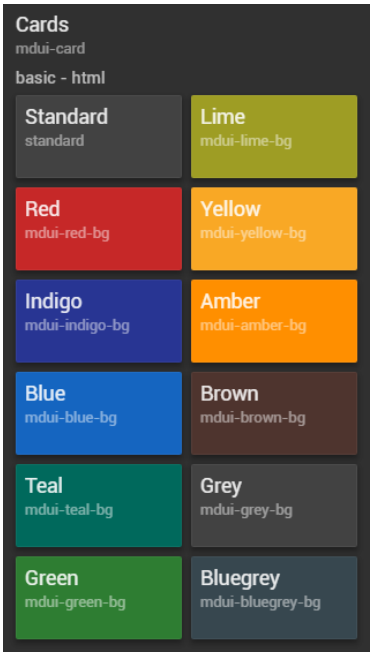
| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| | Width<br>Height<br>Top<br>Left | | Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am linken Rand angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden. |
| **Ja** | CSS Klasse | mdui-rnav | Left-Navigation |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe (siehe dort) |

## 9.Cards

Cards können, müssen aber nicht verwendet werden. Mit Cards lassen sich Widgets optisch gruppieren. Für eine Card wird das Widget ***basis-html*** verwendet.

Im **responsive Design** wird die Card nicht dem View mit den anzuzeigenden Werten zugewiesen, sondern jeweils dem Container im Content, in welchem der View angezeigt wird.

Sollen Cards Titel und Untertitel haben, so können hierfür die unter „Labels" beschriebenen CSS-Styles verwendet werden.



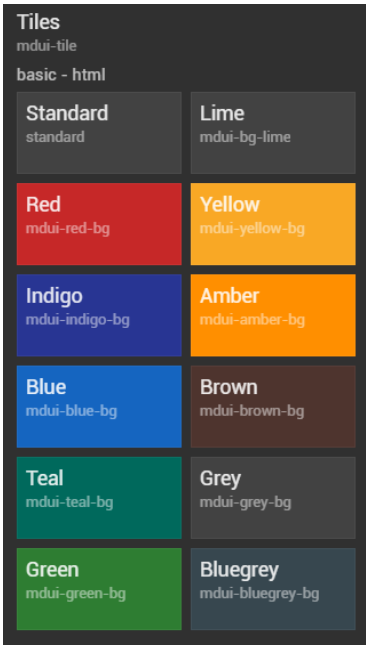

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-card | |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein |

## 10.Tiles

Tiles können, müssen aber nicht verwendet werden. Mit Cards lassen sich Widgets optisch gruppieren. Für eine Card wird das Widget ***basis-html*** verwendet. Gegenüber den Cards besitzen sie keinen Schatten.

Im **responsive Design** wird die Card nicht dem View mit den anzuzeigenden Werten zugewiesen, sondern jeweils dem Container im Content, in welchem der View angezeigt wird.

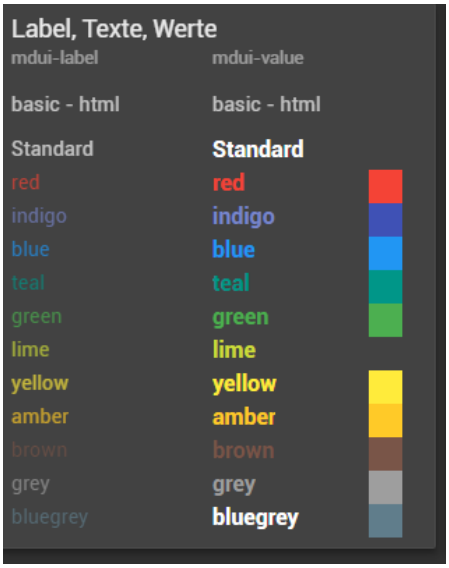
Sollen Cards Titel und Untertitel haben, so können hierfür die unter „Labels" beschriebenen CSS-Styles verwendet werden.

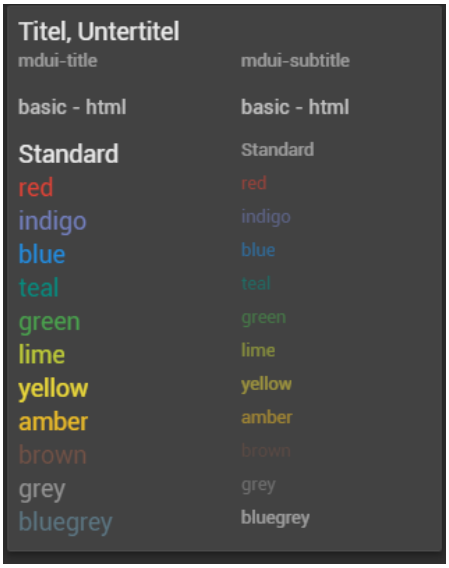



| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-tile | |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein |

## 11.Labels

Für ein Label (Text) wird das ***basis-html*** Widget verwendet. Es kann sowohl für die Darstellung von Titeln, Untertiteln, Labels als auch von Werten verwendet werden. Labels werden etwas dunkler und kleiner als Werte dargestellt.




| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-label oder<br>mdui-value oder<br>mdui-title oder<br>mdui-subtitle | Label-Text<br>Wert<br>Titel, Überschrift<br>Untertitel |
| | CSS Klasse | mdui-(color) | Schriftfarbe. (color) kann eine der Farben (siehe dort) sein |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein |

**TIPP**
Da bei jedem Attribut auch auf ioBroker Variable zugriffen werden kann, ist auch eine Farbsteuerung über Variable möglich. Hierzu eine Zeichenvariable deklarieren und ihr den gewünschten Farbwert "red", "green", ... zuweisen. In der CSS Klasse dann mdui-{meine Variable} verwenden.

Bsp: mdui-{javascript.0.farbwert-temperatur}

## 12.Buttons

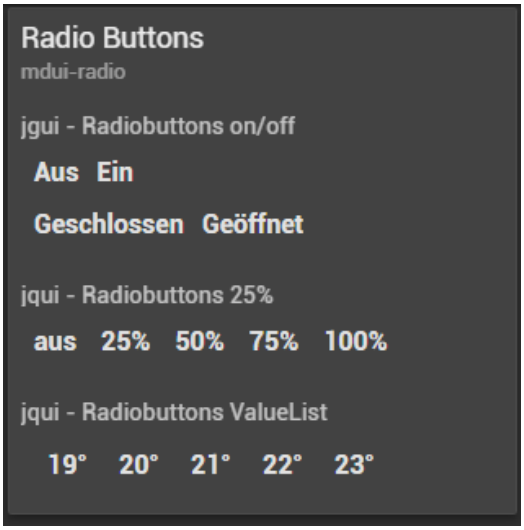
Für einen Button können verschiedene ***jqui-Button*** Widgets verwendet werden. Bei den meisten funktioniert die Darstellung korrekt, wenn eine der Button CSS Klassen zugewiesen wird. Im Material Design gibt es drei Button Arten: flat-button, raised-button und floating-button.



| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-flat-button oder mdui-raised-button oder mdui-floating-button | |
| | CSS Klasse | mdui-(color) | Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe für flat ist „blau“, für die anderen "weiß". |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "blau". |

## 13.Radio Buttons

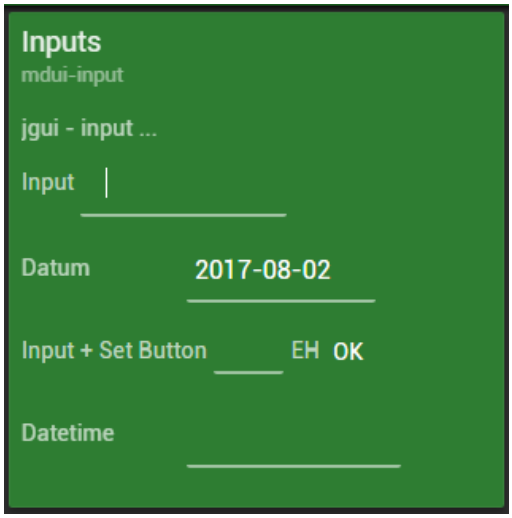
Für Radio-Eingaben wird eines der ***jqui-Radio...*** Widgets verwendet. Die Darstellung der wählbaren Optionen erfolgt als normaler, fetter Text. Die jeweils aktive Option wird in der Akzentfarbe dargestellt und unterstrichen.




| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-radio | |
| | CSS Klasse | mdui-(color) | Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß". |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent". |
| | CSS Klasse | mdui-(color)-acc | Akzentfarbe um die aktuelle Auswahl zu kennzeichnen. Vorgabe ist „blau“. |

# 14.Inputs

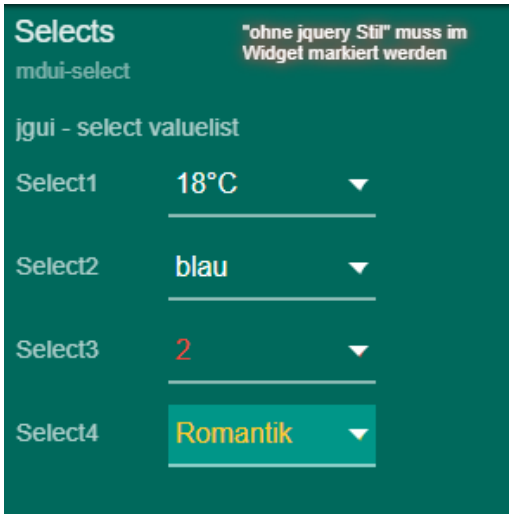
Für einen Input können verschiedene ***jqui-Input*** Widgets verwendet werden. Bei den meisten funktioniert die Darstellung korrekt. Material Design typisch werden Inputs nur durch eine untere Rahmenlinie angezeigt, die beim Fokus farbig wird.




| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-input | |
| | CSS Klasse | mdui-(color) | Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß". |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent". |

## 15.Selects

Für ein Select wird das ***jqui-Select Valuelist*** Widgets verwendet, hier muss die Eigenschaft „Ohne jQuery Stil“ gewählt sein. Material Design typisch wird ein Select nur durch eine untere Rahmenlinie und einer Pfeil-Nach-Unten  Schaltfläche angezeigt. Die sich öffnende Select-Liste beim [Tap] ist abhängig vom Browser.

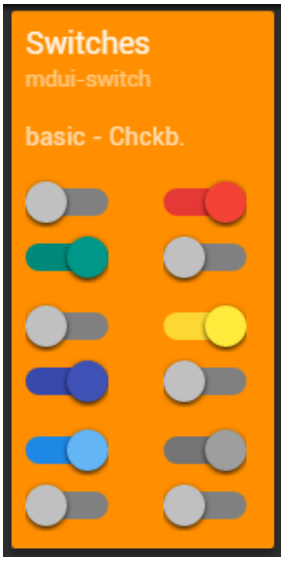



| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-select | |
| | CSS Klasse | mdui-(color) | Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß". |
| | CSS Klasse | mdui-(color)-bg | Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent". |

Wichtig
Damit der Select funktioniert, muss die die Eigenschaft „Ohne jQuery Stil“ im Widget markiert werden!

## 16.Switches

Für einen Switch wird das Widget ***basis-bool checkbox*** verwendet.



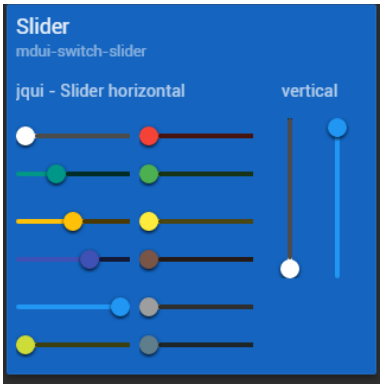

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-switch | |
| **Ja** | HTML anhängen | <label for="(id)_checkbox"> </label> | **Wichtig!** Wobei (id) durch die ID des Controls selbst ersetzt werden muss, Bsp: "w00033_checkbox" |
| | CSS Klasse | mdui-(color)-acc | Hiermit kann festgelegt werden, welche Farbe im EIN Zustand verwendet werden soll. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß". |

Wichtig
Damit der Switch funktioniert, muss die oben angegebenen HTML-anhängen Erweiterung eingegeben werden.

# 17.Slider

## 17.1.Farbige Slider

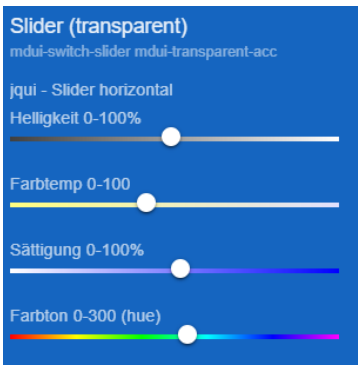
Für einen Slider wird das Widget ***jqui-slider vertical*** bzw. Widget ***jqui-slider horizontal*** verwendet.




| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-slider | |
| | CSS Klasse | mdui-(color)-acc | Hiermit kann festgelegt werden, welche Farbe für den Button und den aktiven Bereich verwendet werden soll. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß". |
| | CSS Klasse | mdui-transparent-acc | Transparente Darstellung des Silders-Weges, nur das Slider-Handle wird dargestellt. Hierdurch ist es möglich einen beliebigen Hintergrund zu wählen. |

## 17.2.Transparente Slider

Wird dem Slider-Widget die CSS-Klasse mdui-transparent-acc zugewiesen, so wird nur das Slider-Handle (der Anfasser) gezeichnet, der Slider-Weg ist transparent. Dieses ermöglicht einen individuellen Slider-Hintergrund, z.B. um eine Farbauswahl, Helligkeit usw darstellen zu können.




Beispiele für Hintergründe (jeweils ein eigenes basic-HTML hinter dem Slider mit background-Eigenschaft):

Helligkeit: linear-gradient(to right, #404040, #ffffff)

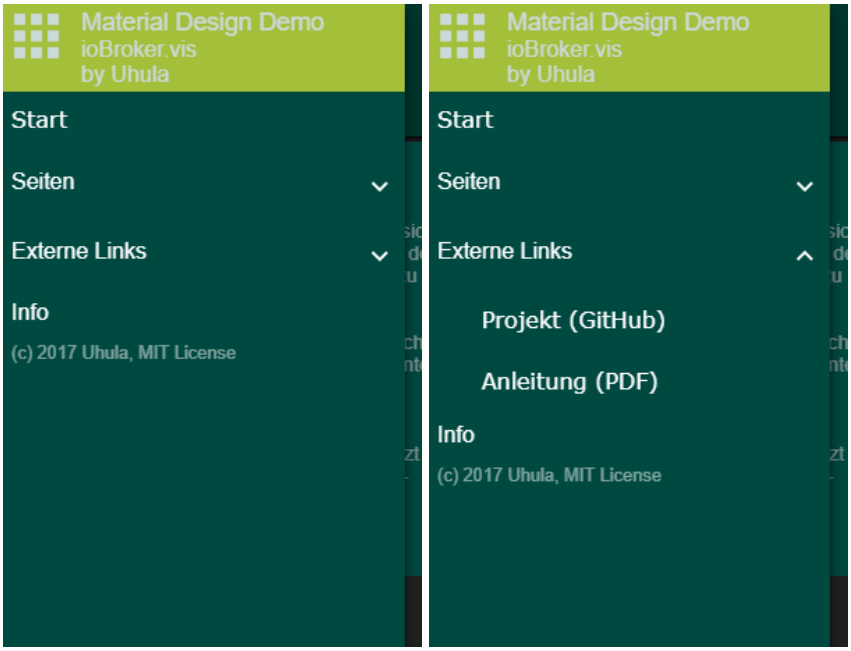
Farbtemperatur: linear-gradient(to right, #FFFF80, #E0E0FF)

Sättigung: linear-gradient(to right, white, blue)

Farbton: inear-gradient(to right, #ff0000, #ffff00, #00ff00, #00ffff, #0000ff, #ff00ff)

# 18.Baumstruktur-Menüs in der Sidebar

Manchmal möchte man in den Sidebars (Left/Right-Navigation) ein komplettes Menü mit vielen Einträgen unterbringen. Um die Übersichtlichkeit zu wahren und um häufiges Scrollen zu vermeiden, kann man dort mit Baumstruktur-Menüs arbeiten. Bei diesen handelt es sich im Prinzip um 2-stufige Menüs, wobei die jeweils 2.Stufe sich erst öffnet, wenn auf der 1.Stufe eine Schaltfläche betätigt wurde.




Linkes Bild: Mit zwei Baumstruktur-Einträgen, alle geschlossen
Rechtes Bild: Nach [Klick] auf „Externe Links“.

Damit sich die Menü-Einträge so verhalten, müssen ihnen spezielle CSS-Klassen zugeordnet werden. Da ioBroker.vis alle Widgets mit absoluten Positionen rendert, dieses aber hier nicht gewollt ist (die Menüs sollen ja zusammenfallen), werden alle Widgets mit einer Breite von >50% und mdui-float versehen.

## 18.1.Menü-Eintrag 1.Ebene, der eine 2.Ebene öffnen soll

Als Widget dürfen hier keine solche verwendet werden, die einen Link bzw. View nachladen – das würde nicht gehen, da ihnen weder ein Link noch ein View zugewiesen werden. Sondern es werden normale ***basic-HTML*** Widgets verwendet und ihr Inhalt wird als HTML angegeben. Entweder nur als Text „Menübezeichnung“, oder kombiniert ein Text und ein Icon:

```
Menütext<img src="iconpfad" width="24px" height="24px" style="margin-
right:12px;float:right;"></img>
```

Tipp
Wenn der Inhalt ein <img> enthält, wird dieses bei jedem Umschalten um 180° gedreht. Es bieten sich also Icons wie ein „Pfeil nach unten“ an.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-toggle | Damit wird das Widget als toggle-Widget gekennzeichnet, welches andere Widgets anzeigen/verstecken kann |
| **Ja** | CSS Klasse | mdui-group-(name) | Angabe des Gruppennamens, welchen auch die anzuzeigenden/zu versteckenden Menüeinträge |

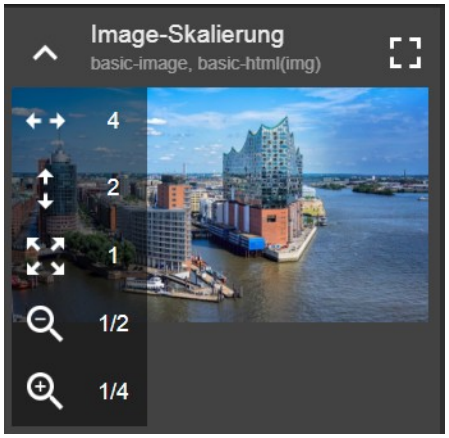
| | | | |
|---|---|---|---|
| | | | bekommen. Hierdurch „weiß“ das Widget, welches es beim [Tap] bearbeiten soll. Es muss darauf geachtet werden, dass unterschiedliche Untermenüs auch unterschiedliche Gruppennamen erhalten. |
| **Ja** | CSS Klasse | mdui-float | |
| **Ja** | Width | >50% | Führt im Zusammenspiel mit mdui-float dazu, dass sich die Einträge automatisch anordnen |
| | CSS Klasse | mdui-flatbutton | |

## 18.2.Menü-Eintrag 2.Ebene

Hier werden nun normale ***Link/View/Navigations***-Widgets verwendet.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-group-(name) | Angabe des Gruppennamens, wie im zugehörigen Menü-Eintrag der 1.Ebene |
| **Ja** | CSS Klasse | mdui-float | |
| **Ja** | Width | >50% | Führt im Zusammenspiel mit mdui-float dazu, dass sich die Einträge automatisch anordnen |
| | CSS Klasse | mdui-flatbutton | |
| | CSS Klasse | mdui-hide | Kann/Sollte angegeben werden, wenn der Menüeintrag als Vorgabe versteckt sein soll. Wirkt sich nur zur Laufzeit aus, nicht im Designer. |

# 19.Bild skalieren, als Vollbild




Häufig werden ***basic-HTML*** oder ***basic-image*** oder ***basic-iFrame*** Widgets verwendet um Bilder von z.B. IP Kameras darzustellen. Wenn man diese in unterschiedlichen Größen betrachten will, muss man jeweils eigene Widgets abrufen. Über die hier beschriebene Methode erübrigt sich dieses, da ein direktes Skalieren des innen liegenden Images (<img>) ermöglicht wird.

Als Schaltflächen werden normale ***basic-HTML*** Wigdets verwendet. Über die Zuweisung von CSS-Klassen wird ihnen ihre Bedeutung zugewiesen.

Beispiel: (Download) https://github.com/Uhula/ioBroker-Material-Design-Style/blob/master/video/image_scale.mp4

Technisch wird das <img> über eine CSS-Anweisung (transform:scale(x)) in der Darstellung verändert. Hierzu müssen sowohl die MD CSS-Anweisungen als auch das MD Skript in das ioBroker.vis Projekt kopiert werden.

Weiterhin ist es möglich die komplette View, auf welcher das Image platziert wurde, als Vollbild darzustellen, siehe hierzu Unterkapitel 3.

## 19.1.Skalierungs-Schaltflächen

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-target-(widgetID) | Angabe der Widget-ID des basic-HTML oder basic-image Widgets, in welchem das <img> skaliert werden soll. Bsp: mdui-target-w00127 |
| **Ja** | CSS Klasse | mdui-scale-(type) | Angabe der Skalierung, die angewendet werden soll. (type)=<br>*fit*=Das <img> wird eingepasst<br>*hfit*= Das <img> wird horizontal eingepasst<br>*vfit*= Das <img> wird vertikal eingepasst<br>*in*= Es wird um den Faktor 1,41 hineingezoomt<br>*out*= Es wird um den Faktor 1,41 herausgezoomt<br>*xxx*= Es wird auf xxx% gezoomt<br>Bsp: mdui-scale-100 |
| | CSS Klasse | mdui-flatbutton | |
| | CSS Klasse | mdui-hide | Kann/Sollte angegeben werden, wenn die Schaltfläche als Vorgabe versteckt sein soll. Wirkt sich nur zur Laufzeit aus, nicht im Designer. |

## 19.2.Skalierungs-Schaltflächen verstecken/anzeigen

Möchte man die Skalierungs-Schaltflächen nur bei Bedarf einblenden, so kann dieses über eine weitere ***basic-HTML*** Widget Schaltfläche geschehen, welche sinnvollerweise nur aus einem Icon besteht und oberhalb angeordnet wird. Siehe im Beispielbild das \/ Symbol.

```
<img src="iconpfad"></img>
```

Tipp
Wenn der Inhalt ein <img> enthält, wird dieses bei jedem Umschalten um 180° gedreht. Es bieten sich also Icons wie ein „Pfeil nach unten" an.

Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-toggle | Damit wird das Widget als toggle-Widget gekennzeichnet, welches andere Widgets anzeigen/verstecken kann |
| **Ja** | CSS Klasse | mdui-target-(widgetID) | Angabe der Widget-ID, die auch in den Skalierungs-Widgets verwendet wurde. Alle Widgets mit dem gleichen mdui-target-(widgetID) Eintrag werden beim [Tap] versteckt/gezeigt. |
| | CSS Klasse | mdui-flatbutton | |

## 19.3.Vollild-Modus

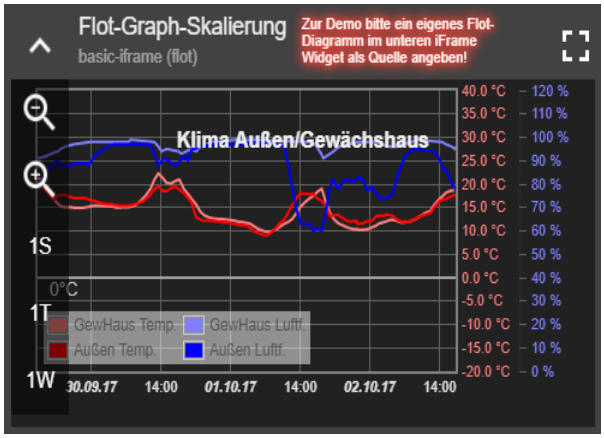
Bisher konnte man erreichen, dass das <img> innerhalb seines Platzes skaliert wird. Befindet es sich jedoch in einem eigenen (card-) View, welcher auf einem (content.)View angezeigt wird, so kann man dieses (card-) View über eine weitere ***basic-HTML*** Widget Schaltfläche im Vollbildmodus darstellen; und auch wieder zurück.

Technisch wird hierfür der View aus seinem HTML-Kontext entfernt und direkt in den vis-container eingehängt und bekommt über direkte CSS-Anweisungen das Vollbildformat.

Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-fullscreen | Damit wird das Widget als Vollbild-Widget gekennzeichnet, welches sein View als Vollbild anzeigen kann. Es wird automatisch das View verwendet, in welchem es sich befindet. |
| | CSS Klasse | mdui-flatbutton | |

# 20.FLOT Diagramm Zeitspanne setzen, Vollbild




Ein FLOT Diagramm wird in einem ***basic-iFrame*** Widgets dargestellt. Zwar ist es möglich, hier einen „Finger-ZOOM" für die Bestimmung der Zeitspanne vorzunehmen, jedoch ist es dann eher ein Zufall, eine gewünschte Zeitspanne wie z.B. „1 Woche" zu erwischen. Besser geht dieses, wenn man dafür eigene Schaltflächen hätte. Und genau diese werden hier beschrieben.

Als Schaltflächen werden normale ***basic-HTML*** Wigdets verwendet. Über die Zuweisung von CSS-Klassen wird ihnen ihre Bedeutung zugewiesen.

Beispiel: (Download) https://github.com/Uhula/ioBroker-Material-Design-Style/blob/master/video/flot_timespan.mp4

Technisch wird die Eigenschaft „Quelle" , welche die FLOT-URL enthält, via Javascript manipuliert. Hierzu müssen sowohl die MD CSS-Anweisungen als auch das MD Skript in das ioBroker.vis Projekt kopiert werden.

Weiterhin ist es möglich die komplette View, auf welcher das FLOT-Diagramm platziert wurde, als Vollbild darzustellen, siehe hierzu Unterkapitel 3.

## 20.1.Zeitspannen-Schaltflächen

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-target-(widgetID) | Angabe der Widget-ID des basic-iFrame Widgets, in welchem das FLOT-Diagramm platziert wurde. Bsp: mdui-target-w00271 |
| **Ja** | CSS Klasse | mdui-timespan-(time) | Angabe der Zeitspanne. (time)=<br>*inc*= Die Zeitspanne wird verdoppelt<br>*dec*= Die Zeitspanne wird halbiert<br>*xxxxx*= Zeitspanne in Minuten:<br>*60*=Eine Stunde<br>*1440*= Ein Tag<br>*10080*= Eine Woche usw.<br>Bsp: mdui-timespan-1440 |
| | CSS Klasse | mdui-flatbutton | |
| | CSS Klasse | mdui-hide | Kann/Sollte angegeben werden, wenn die Schaltfläche als Vorgabe versteckt sein soll. Wirkt sich nur zur Laufzeit aus, nicht im Designer. |

## 20.2.Zeitspannen-Schaltflächen verstecken/anzeigen

Möchte man die Skalierungs-Schaltflächen nur bei Bedarf einblenden, so kann dieses über eine weitere ***basic-HTML*** Widget Schaltfläche geschehen, welche sinnvollerweise nur aus einem Icon besteht und oberhalb angeordnet wird. Siehe im Beispielbild das \/ Symbol.

```
<img src="iconpfad"></img>
```

Tipp
Wenn der Inhalt ein <img> enthält, wird dieses bei jedem Umschalten um 180° gedreht. Es bieten sich also Icons wie ein „Pfeil nach unten" an.

Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-toggle | Damit wird das Widget als toggle-Widget gekennzeichnet, welches andere Widgets anzeigen/verstecken kann |
| **Ja** | CSS Klasse | mdui-target-(widgetID) | Angabe der Widget-ID, die auch in den Skalierungs-Widgets verwendet wurde. Alle Widgets mit dem gleichen mdui-target-(widgetID) Eintrag werden beim [Tap] versteckt/gezeigt. |
| | CSS Klasse | mdui-flatbutton | |

## 20.3.Vollild-Modus

Befindet sich das FLOT-Diagramm in einem eigenen (card-) View, welcher auf einem (content.)View angezeigt wird, so kann man dieses (card-) View über eine weitere ***basic-HTML*** Widget Schaltfläche im Vollbildmodus darstellen; und auch wieder zurück.

Technisch wird hierfür der View aus seinem HTML-Kontext entfernt und direkt in den vis-container eingehängt und bekommt über direkte CSS-Anweisungen das Vollbildformat.

Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

| Muss? | Eigenschaft | Werte | Beschreibung |
|---|---|---|---|
| **Ja** | CSS Klasse | mdui-fullscreen | Damit wird das Widget als Vollbild-Widget gekennzeichnet, welches sein View als Vollbild anzeigen kann. Es wird automatisch das View verwendet, in welchem es sich befindet. |
| | CSS Klasse | mdui-flatbutton | |

# 21.Flashen – Blinken – Pulsieren

... folgt ...

# 22.Bargraphanzeigen

... folgt ...

# 23.Tabellen

... folgt ...

# 24.Demo Projekte

Mit auf GitHub befinden sich drei Demo-Projekte, welche eine ioBroker.vis Visualisierung ähnlich einer Android-App mit Application-bar, Tab-Navigation und Bottom-Navigation bieten. Die Controls dieser Demo-Projekte sind nicht mit echten Instanzen verbunden. Die verwendeten Symbole (Icons) befinden sich alle im Projektunterordner „images".

Die Demo-Projekte, also die ZIP-Datei, können in ioBroker.vis unter „Projekt importieren" eingelesen werden.

**MD_Demo.zip**

Ein ioBroker Projekt in welchem alle Möglichkeiten des Material Design Styles dargestellt werden.

**MD_Simple.zip**

Ein ioBroker Projekt welches als Basis für ein eigenes Projekt verwendet werden kann. Es ist mit allen notwendigen Views für die Navigation ausgestattet. Die Seiten sind neutral gehalten.

# 25.Änderungen

| | | |
|---|---|---|
| 24.09.2017 | UH | Kapitel „Select“ hinzugefügt<br>Kapitel „Transparente Slider“ hinzugefügt |
| 02.10.2017 | UH | Kapitel „FLOT Diagramme“ hinzugefügt |
| | | |

# 26.Lizenz

### The MIT License (MIT)

Copyright (c) 2017ff Uhula

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### Deutsche Übersetzung

Copyright (c) 2017ff Uhula

Hiermit wird unentgeltlich jeder Person, die eine Kopie der Software und der zugehörigen Dokumentationen (die "Software") erhält, die Erlaubnis erteilt, sie uneingeschränkt zu nutzen, inklusive und ohne Ausnahme mit dem Recht, sie zu verwenden, zu kopieren, zu verändern, zusammenzufügen, zu veröffentlichen, zu verbreiten, zu unterlizenzieren und/oder zu verkaufen, und Personen, denen diese Software überlassen wird, diese Rechte zu verschaffen, unter den folgenden Bedingungen:

Der obige Urheberrechtsvermerk und dieser Erlaubnisvermerk sind in allen Kopien oder Teilkopien der Software beizulegen.

DIE SOFTWARE WIRD OHNE JEDE AUSDRÜCKLICHE ODER IMPLIZIERTE GARANTIE BEREITGESTELLT, EINSCHLIEßLICH DER GARANTIE ZUR BENUTZUNG FÜR DEN VORGESEHENEN ODER EINEM BESTIMMTEN ZWECK SOWIE JEGLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEGLICHEN SCHADEN ODER SONSTIGE ANSPRÜCHE HAFTBAR ZU MACHEN, OB INFOLGE DER ERFÜLLUNG EINES VERTRAGES, EINES DELIKTES ODER ANDERS IM ZUSAMMENHANG MIT DER SOFTWARE ODER SONSTIGER VERWENDUNG DER SOFTWARE ENTSTANDEN.