

**ioBroker.vis**

**im**

# **Material Design CSS - Style**

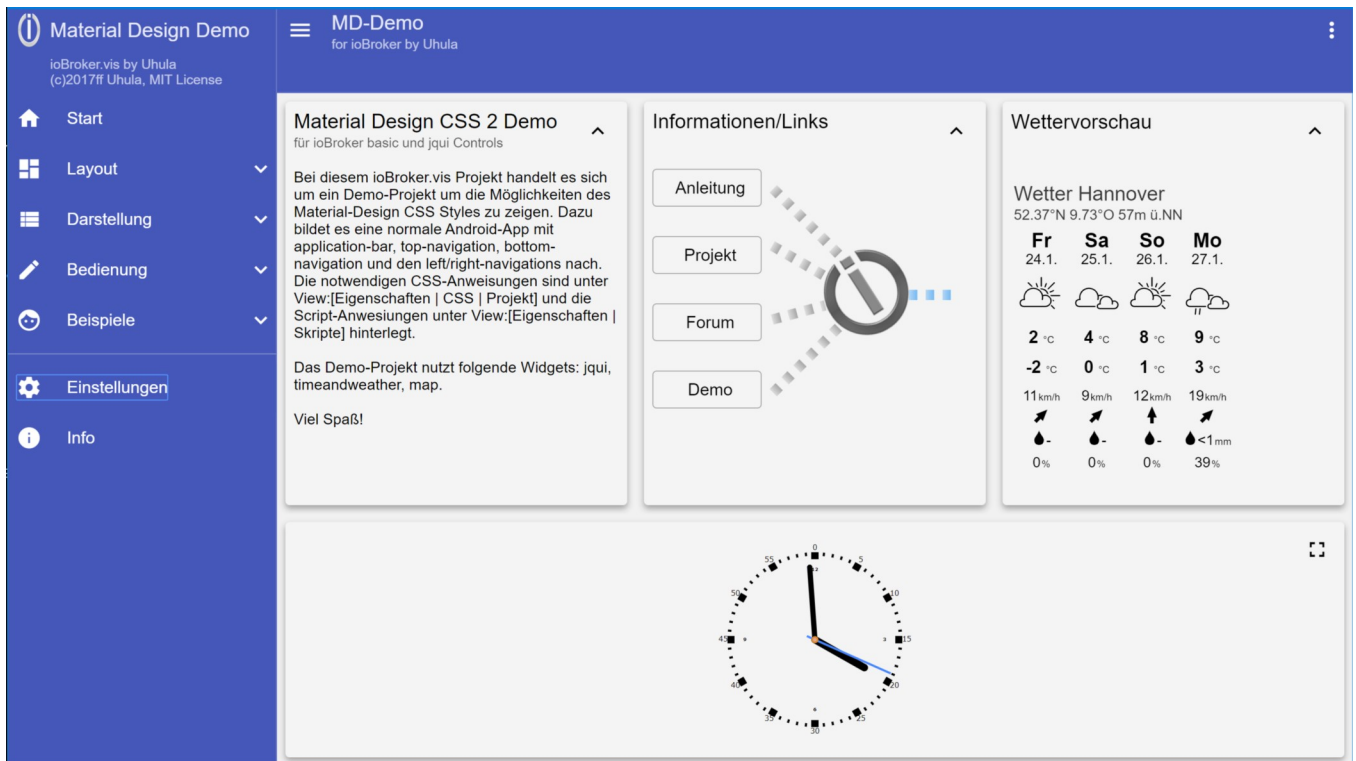
**Version MDCSS v2**

**© Uhula 2017, 2020, MIT License**

# Inhaltsverzeichnis

<b>1. Anleitung zur Material Design CSS.....</b>	<b>3</b>	<b>12. Buttons.....</b>	<b>26</b>
1.1. Installation.....	3	<b>13. Radio Buttons.....</b>	<b>27</b>
1.2. Konfiguration.....	4	<b>14. Chips.....</b>	<b>28</b>
<b>2. Farben.....</b>	<b>5</b>	<b>15. Inputs.....</b>	<b>29</b>
2.1. Farben variabel verwenden.....	6	<b>16. Selects.....</b>	<b>30</b>
2.2. Light-Theme und Dark-Theme.....	6	<b>17. Switches.....</b>	<b>31</b>
2.3. Light-/Dark-Theme zur Laufzeit (mdui-config).....	7	<b>18. Slider.....</b>	<b>32</b>
2.4. Light-/Dark-Theme Entwurfszeit (vis-Designer).....	8	18.1. Farbige Slider.....	32
<b>3. Icons, WebFont.....</b>	<b>10</b>	18.2. Transparente Slider.....	33
3.1. Material Icons von Google.....	10	<b>19. Baumstruktur-Menüs in der Inav, rnav.....</b>	<b>34</b>
3.2. Material Design Icons (Templarian).....	11	19.1. Menü-Eintrag 1.Ebene.....	34
<b>4. Layout.....</b>	<b>12</b>	19.2. Menü-Eintrag 2.Ebene.....	35
4.1. Raster- und Grid-Größen.....	12	<b>20. Bild skalieren, als Vollbild.....</b>	<b>36</b>
4.2. Layout-Zeichnung.....	13	20.1. Skalierungs-Schaltflächen.....	36
<b>5. Application bar.....</b>	<b>14</b>	20.2. Schaltflächen verstecken/anzeigen.....	36
5.1. abar-View.....	14	20.3. Vollbild-Modus.....	37
5.2. abar Container.....	14	<b>21. FLOT Diagramm Zeitspanne setzen, Vollbild...38</b>	
<b>6. Top-Navigation, Bottom-Navigation.....</b>	<b>15</b>	21.1. Zeitspannen-Schaltflächen.....	38
6.1. tnav-View und bnav-View.....	15	21.2. Schaltflächen verstecken/anzeigen.....	38
6.2. tnav-/bnav-Container.....	15	21.3. Vollbild-Modus.....	39
<b>7. Content.....</b>	<b>17</b>	<b>22. Glühen - Flashen – Blinken – Pulsieren.....</b>	<b>40</b>
7.1. content-View.....	17	<b>23. Bargraphanzeigen.....</b>	<b>41</b>
7.2. content-Container.....	18	23.1. Farben.....	42
7.3. Grid-Struktur, Responsive Design.....	18	<b>24. Tabellen.....</b>	<b>43</b>
<b>8. Linke Navigation, rechte Navigation.....</b>	<b>20</b>	24.1. Card/Tile Darstellung mit Label.....	45
8.1. Inav-Views, rnav-Views.....	20	24.2. Tabellen responsive gestalten.....	46
8.2. Inav-, rnav-Container.....	20	<b>25. Menüs, Popupmenüs.....</b>	<b>47</b>
<b>9. card-Views.....</b>	<b>22</b>	<b>26. Listen.....</b>	<b>48</b>
<b>10. Labels.....</b>	<b>23</b>	<b>27. Demo Projekte.....</b>	<b>49</b>
<b>11. States.....</b>	<b>24</b>	<b>28. mdui Übersicht.....</b>	<b>50</b>
		<b>29. Änderungen.....</b>	<b>51</b>
		<b>30. Lizenz.....</b>	<b>52</b>

# 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. Weiterhin ist es notwendig, den Javascript Code auf der Skripte-Registerseite einzufügen.

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
<b>Ja</b>	CSS Klasse	...	Der Wert MUSS gesetzt werden
	CSS Klasse	...	Optional

## 1.1. Installation

Da es sich nicht um einen ioBroker Adapter handelt, ist im ioBroker Admin auch nichts zu installieren. Die Installation erfolgt ausschließlich in der vis. Um das MDCSS dort nutzen zu können, kann entweder ein Beispielprojekt als Basis verwendet werden, oder es wird mit einem neuen vis-Projekt begonnen.

### Beispiel Projekte verwenden (empfohlen)

Unter <https://github.com/Uhula/ioBroker-Material-Design-Style/tree/master/ioBroker%20projects> befinden sich Beispielprojekte MD\_Demo und MD\_Simpel welche herunter geladen und als Projekt in der vis importiert werden können. Bitte die Projektnamen (vorerst) so belassen.

MD\_Demo ist ein umfangreiches Projekt in welchem fast alle MDCSS Funktionalitäten benutzt werden, es kann auch später als Copy&Paste für eigene vis-Projekte gute Dienste leisten.

MD\_Simpel ist ein fast leeres Projekt und bietet sich als Basis für eigene Projekte an.

### **Mit einem neuen vis-Projekt starten**

Wenn mit einem neuen vis-Projekt gestartet werden soll, muss dieses nach der Neuanlage noch erweitert werden:

- Download der Projekt-CSS von Github (<https://github.com/Uhula/ioBroker-Material-Design-Style/tree/master/source> ) und einfügen in der vis unter „CSS | Projekt“
- Download der Projekt-Javascript von Github (<https://github.com/Uhula/ioBroker-Material-Design-Style/blob/master/source/script.js> ) und einfügen in der vis unter „Skripte“
- Download der WebFont Dateien MaterialIcons-Regular.ttf, MaterialIcons-Regular.eot und MaterialIcons-Regular.woff von <https://github.com/google/material-design-icons/tree/master/iconfont> und hochladen der Dateien über die vis in den Projektordner.  
In den Projekt-CSS Anweisungen werden diese Dateien eingebunden. Je nach Projektnamen/Speicherordner müssen die Pfadangaben dort einmalig angepasst werden. Es ist auf Groß-/Kleinschreibung zu achten. Siehe auch Kapitel „Icons“.

## **1.2.Konfiguration**

### **Der Pfad zum Material Icons-WebFont**

In den Projekt-CSS Anweisungen werden diese Dateien eingebunden. Je nach Projektnamen/Speicherordner müssen die Pfadangaben dort einmalig angepasst werden. Es ist auf Groß-/Kleinschreibung zu achten.

Siehe auch Kapitel „3.1 Material Icons von Google“.

### **(Optional) Pfad zu den Material Design Icons-WebFont**

Sollen auch Icons aus dem "Material Design Icons" Projekt nutzbar sein, so sind die umfassenden Kommentarsymbole bei (2) zu entfernen.

Siehe auch Kapitel „3.2 Material Design Icons (Templarian)“.

### **(Optional) Nutzung des Dark-Themes zur Designzeit in der vis**

Das vis Design ist per Vorgabe auf "light"-Theme, also schwarze Schrift auf weißem Grund eingestellt, wer in der vis im "dark"-Theme arbeiten möchte, muss Einstellungen unter ":root / Design time" vornehmen

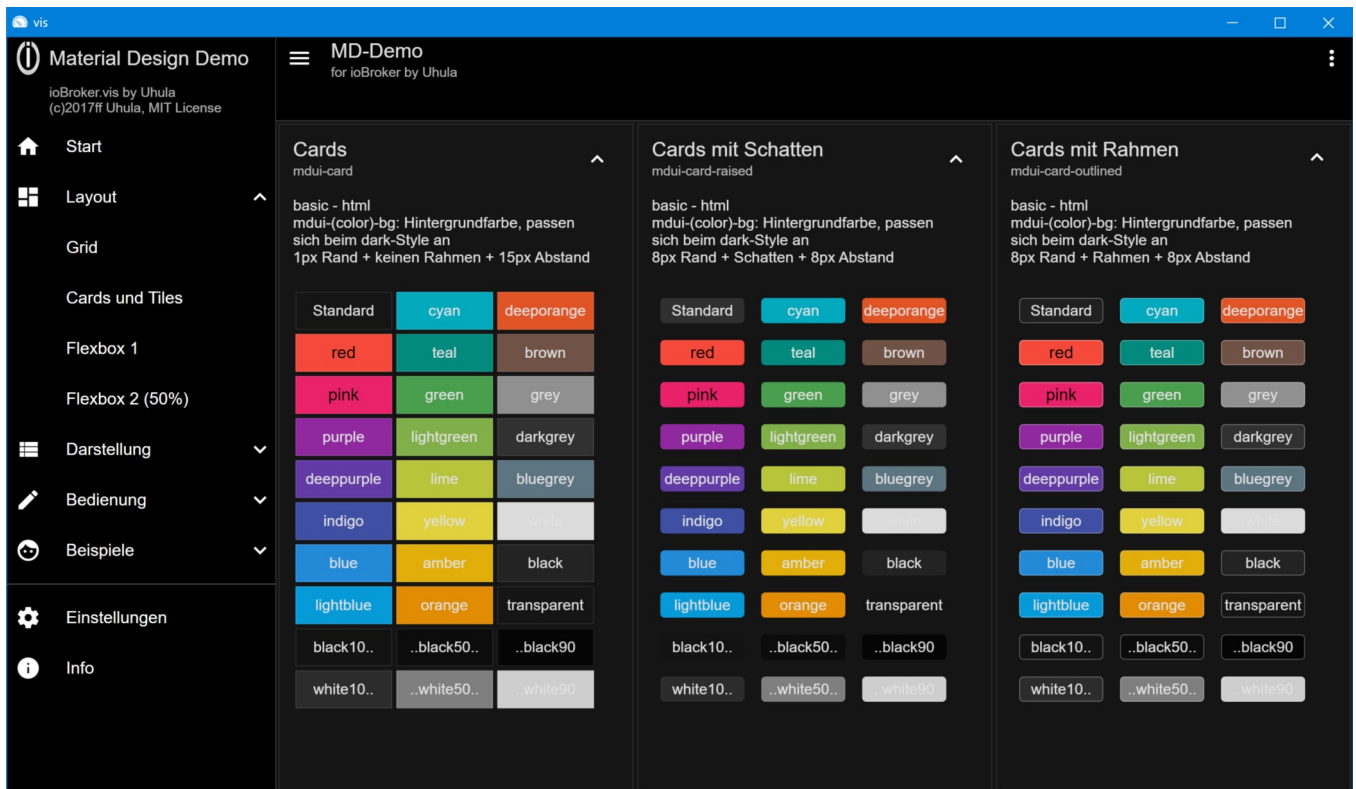
Siehe auch Kapitel „2.4 Light-/Dark-Theme Entwurfszeit (vis-Designer)“.

## 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), `mdui-(color)-ol` (Rahmenfarben) 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, pink, purple, deeppurple, indigo, blue, lightblue, cyan, teal, green, lightgreen, lime, yellow, amber, orange, deeporange, brown, grey, darkgrey, bluegrey, white, black**



Weiterhin stehen teilweise schwarze und weiße Teiltransparenzen zur Verfügung, z.B. wenn ein Bereich mal leicht abgedunkelt werden soll. Die Zahl entspricht der Deckkraft in %:

**black010, black020, ... black090, white010, white020. ... white090**

Auch stehen drei eigene individuelle Farbwerte zur Verfügung, welche über die Konfiguration mit eigenen Farben versehen werden können:

**color1, color2, color3**

### 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.

## 2.1. Farben variabel verwenden

Möchte man eine Hintergrund-/Schriftfarbe in Abhängigkeit von ioBroker.vis Variablen steuern, so bietet sich die Nutzung der ioBroker.vis Bindings-Funktionalität an.

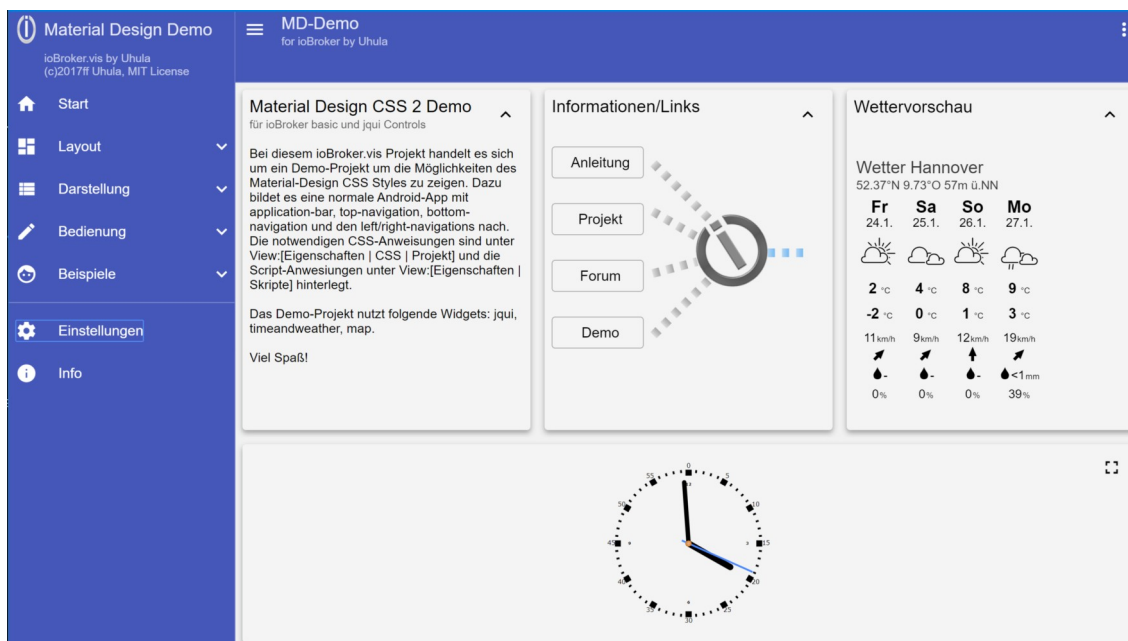
### BEISPIEL

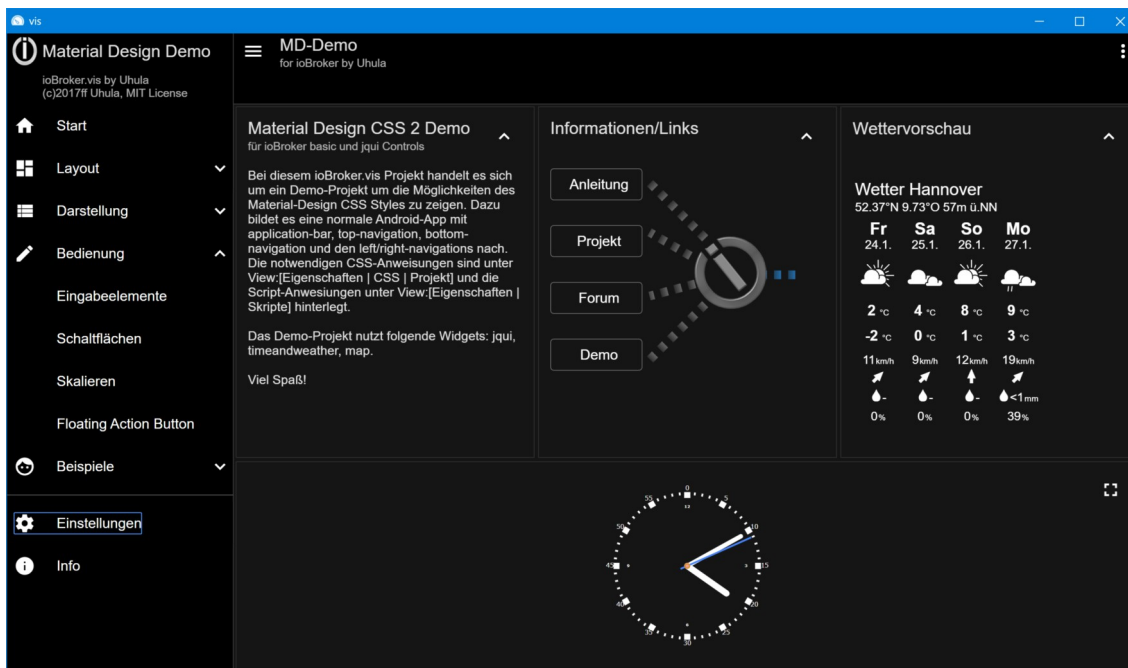
Morgens soll der Hintergrund eines Widgets „mdui-blue-bg“ sein, abends „mdui-brown-bg“

1. Anlage einer ioBroker.vis Variablen, z.B. `0_userdata.0.mdui.vis.mybackground`
2. Schreiben eines Scriptes, welches der Variablen die Werte „mdui-blue-bg“ bzw. „mdui-brown-bg“ zeitgesteuert zuweist
3. In den betreffenden Widgets in der CSS Eigenschaft "{0\_userdata.0.mdui.vis.mybackground}" eintragen  
ioBroker.vis ersetzt den {}-Inhalt beim Rendern durch die Variablenwerte und sorgt auch für Updates zur Anzeigzeit.

## 2.2. Light-Theme und Dark-Theme

Das MDCSS ist so entworfen, dass man in der Darstellung sowohl mit einem Light-Theme (helle Hintergründe) als auch mit einem Dark-Theme (dunkle Hintergründe) arbeiten kann. Es ist sogar möglich z.B. tagsüber das Light-Theme und nachts das Dark-Theme zu verwenden. (Fast) Alle anderen Farben, z.B. die der Schriften, passen sich automatisch an das gewählte Thema an.





## 2.3.Light-/Dark-Theme zur Laufzeit (mdui-config)

Ob es sich bei der Darstellung zur Laufzeit um Light- oder Dark-Theme handelt, wird automatisch anhand der gesetzten Hintergrundfarben bestimmt. Hierbei werden die einzelnen Bereiche Kopfzeile, obere Navigation, linke Navigation, rechte Navigation, untere Navigation und Inhalt separat betrachtet. Es sind somit auch Mischformen wie Kopfzeile und obere Navigation im Dark-Theme und Inhalt im Light-Theme möglich.

Welche Farben konkret verwendet werden, wird innerhalb des Projektes in einem HTML-Widget festgelegt, welches die CSS-Klasse `mdui-config` zugewiesen bekommt. Dieses befindet sich i.d.R. in der Kopfzeile (im abar-View), da diese auf jeder Seite eingebunden ist.

In der HTML-Eigenschaft des Widgets ist die Konfiguration in der JSON Notifikation zu erfassen:

```
"primary_color": "#000000",
"abar_color": "#121212",
"tnav_color": "#000000",
"bnav_color": "#000000",
"lnav_color": "#000000",
"rnav_color": "#000000",
"secondary_color": "amber",
"content_color": "#121212",
"lnav_fixed_width": "{demo_lnav_fixed_width}",
"lnav_fixed_open": "true"
"color1": "",
"color1_dark": "",
"color2": "",
"color2_dark": "",
"color3": "",
"color3_dark": ""
```

Als **Farbwert** kann eines der Farbwerte (rot, pink, ...), ein RGB-Farbcode in der Form `#rrggbb` oder ein leerer Wert gewählt werden. Letzterer entspricht dann dem im MDCSS genutzten Vorgabewert.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	primary_color	Farbwert	Diese Farbe wird für die Darstellung der

			Kopfzeile und der oberen/unteren Navigation verwendet wenn diese keine eigenen Angaben haben
	abar_color	Farbwert	Optional kann die <b>Kopfzeile</b> eine eigene Farbzuzuweisung erhalten
	tnav_color	Farbwert	Optional kann die <b>oberen Navigation</b> eine eigene Farbzuzuweisung erhalten
	bnav_color	Farbwert	Optional kann die <b>untere Navigation</b> eine eigene Farbzuzuweisung erhalten
	lnav_color	Farbwert	Optional kann die <b>linke Navigation</b> eine eigene Farbzuzuweisung erhalten. Wenn nicht, nutzt sie die content_color
	rnav_color	Farbwert	Optional kann die <b>rechte Navigation</b> eine eigene Farbzuzuweisung erhalten. Wenn nicht, nutzt sie die content_color
<b>Ja</b>	secondary_color	Farbwert	Hierbei handelt es sich um die Farbe, welche z.B. für Akzente genutzt wird.
<b>Ja</b>	content_color	Farbwert	Die Farbe für den Hintergrund des Inhaltsbereichs. Für ein dark-Theme bietet sich #121212 an, für das light-Theme #f8f8f8.
<b>Ja</b>	lnav_fixed_width	0 ... 9999	Angabe der Anzeigebreite in Pixel ab der die linke Navigation fest eingeblendet werden soll. Ein Wert von 0 verhindert dieses, 640 ist der Vorgabewert
	lnav_fixed_open	true   false	Bei true wird die linke Navigation beim Seitenwechsel automatisch geöffnet (wenn die Breite ausreichend ist)
	color1 color1_dark	Farbwert	Freie Farben für das light-Theme und dark-Theme für die Verwendung in mdui-color1, mdui-color1-bg, mdui-color1-ol
	color2 color2_dark	Farbwert	Freie Farben für das light-Theme und dark-Theme für die Verwendung in mdui-color2, mdui-color2-bg, mdui-color2-ol
	color3 color3_dark	Farbwert	Freie Farben für das light-Theme und dark-Theme für die Verwendung in mdui-color3, mdui-color3-bg, mdui-color3-ol

## 2.4.Light-/Dark-Theme Entwurfszeit (vis-Designer)

Grundsätzlich erfolgt das Designen in der vis im Light-Theme, also auf hellen Hintergründen. Es besteht jedoch auch die Möglichkeit auch im vis-Design ein dark-Theme einzustellen. Hierzu muss nach der Übernahme der CSS-Anweisungen in die Projekt-CSS dort eine Änderung durchgeführt werden:

Die beiden Zeile FÜR DAS DARK THEME LÖSCHEN müssen gelöscht werden.

```
/* (3) Umschalten vis-Edit light/dark Theme */
:root {
  /* Design-time light-Theme */
```



```
--content-background: #f8f8f8;
--design-font-color: #000000;
--design-hint-background: #0000ff;
--design-hint-color: #ffffff;
--design-grid: #000000;
--design-color-033:rgba(0,0,0,.33);

/* Design-time dark-Theme */
/* FÜR DAS DARK_THEME DIESE ZEILE LÖSCHEN
--content-background: #404040;
--design-font-color: #ffffff;
--design-hint-background: #ff8000;
--design-hint-color: #ffffff;
--design-grid: #ffffff;
--design-color-033:rgba(255,255,255,.33);
FÜR DAS DARK_THEME DIESE ZEILE LÖSCHEN */
}
```

### 3. Icons, WebFont

Natürlich kann man für Symbole auch Bilddateien wie PNG usw. verwenden, allerdings haben diese zwei gravierende Nachteile:

- a) sie skalieren nicht und sehen, wenn sie denn nicht in nativer Größe angezeigt werden, verschwommen aus
- b) sie passen ihre Farbe nicht an, was das dynamische Umschalten zwischen light/dark Themes fast unmöglich macht

Für beides gibt es eine Lösung: Die Verwendung von WebFonts für die Icon Darstellung.

#### 3.1. Material Icons von Google



Google hat 900 Icons in einem WebFont untergebracht, welche im MDCSS direkt verwendet werden können. Eine Übersicht findet sich auf <https://material.io/resources/icons/?style=baseline> .

Damit dieses funktioniert, müssen entsprechende WebFont-Dateien mit in den Projektordner kopiert werden. In den Beispiel-Projekten MD\_Demo und MD\_Simpel sind diese im Projekt-Unterordner images vorhanden. Es handelt sich um die Dateien:

- MaterialIcons-Regular.ttf
- MaterialIcons-Regular.eot
- MaterialIcons-Regular.woff

In den Projekt-CSS Anweisungen werden diese Dateien eingebunden. Je nach Projektnamen/Speicherordner müssen die Pfadangaben dort einmalig angepasst werden. Es ist auf Groß-/Kleinschreibung zu achten.

```
/* (1) An Projektnamen anpassen (MD_Demo ersetzen) */
@font-face {
  font-family: 'Material Icons';
  font-style: normal;
  font-weight: 400;
  src: url(/vis.0/MD_Demo/images/MaterialIcons-Regular.eot); /* For IE6-8 */
  src: url(/vis.0/MD_Demo/images/MaterialIcons-Regular.woff) format('woff'),
       url(/vis.0/MD_Demo/images/MaterialIcons-Regular.ttf) format('truetype');
}
```

Für die Nutzung der Icons wird kein image-Widgets verwendet, sondern sie werden direkt in den HTML/Text-Eigenschaften der Widgets in der Form `<i class='mdui-icon'>(name)</i>` angegeben.



`<i class='mdui-icon'>home</i>`



<i class='mdui-icon'>alarm</i>

Die Icons skalieren zusammen mit der Schriftgröße und nehmen die Schriftfarbe an.

### 3.2. Material Design Icons (Templarian)

Wem die 900 Google Icons nicht ausreichen kann optional die Material Design Icons nutzen. Auch hierbei handelt es sich um einen WebFont. Diesmal mit rund 5000 (!) Icons.

Eine Übersicht findet sich auf <https://cdn.materialdesignicons.com/5.0.45/> .

Damit dieses funktioniert, müssen entsprechende WebFont-Dateien mit in den Projektordner kopiert werden, z.B. in den images Ordner. Die Dateien können hier herunter geladen werden:

<https://github.com/Templarian/MaterialDesign-Webfont/tree/master/fonts>

. Es handelt sich um die Dateien:

- materialdesignicons-webfont.eot
- materialdesignicons-webfont.ttf
- materialdesignicons-webfont.woff

In den Projekt-CSS Anweisungen werden diese Dateien eingebunden. Je nach Projektnamen/Speicherordner müssen die Pfadangaben dort einmalig angepasst werden. Es ist auf Groß-/Kleinschreibung zu achten. Für die Nutzung müssen in der Projekt-CSS die beiden Zeilen „FÜR DIE ...“ entfernt werden.

```
/* (2) Zur Nutzung der MaterialDesignIcons (https://github.com/Templarian/MaterialDesign-Webfont/tree/master/fonts) die den @font-face Block umfassenden Kommentare entfernen und die Projektnamen (MD_) durch die eigenen ersetzen. Aus dem angegebenen Link müssen die Font-Dateien via vis-Dateimanager in den (project)/images Ordner kopiert werden. In der vis Nutzung in der Form: <i class="mdui-mdi">&#xF02DC;</i> */
```

```
/* FÜR DIE zusätzlichen MaterialDesignIcons DIESE ZEILE LÖSCHEN
```

```
@font-face {
  font-family: "Material Design Icons";
  src: url("/vis.0/MD_Demo/images/fonts/materialdesignicons-webfont.eot?v=5.0.45");
  src: url("/vis.0/MD_Demo/images/materialdesignicons-webfont.woff?v=5.0.45")
  format("woff"),
  url("/vis.0/MD_Demo/images/materialdesignicons-webfont.ttf?v=5.0.45")
  format("truetype");
  font-weight: normal;
  font-style: normal;
}
```

```
FÜR DIE zusätzlichen MaterialDesignIcons DIESE ZEILE LÖSCHEN    */
```

Entgegen dem Google-Font, können bei der Nutzung keine Namen genutzt werden, sondern es muss mit den Char-Codes, welche man auf der WebFont Übersichtsseite sieht, genutzt werden.



<i class='mdui-icon'>&#xF0EF4;</i> (mdi-fishbowl-outline)

Die Icons skalieren zusammen mit der Schriftgröße und nehmen die Schriftfarbe an.

## 4. Layout

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

Aufbau einer Seite:

Left Navigation (Side-Panel)	Application bar	Right Navigation (Side-Panel)
	Top-Navigation (Tabs)	
	Content	
	Bottom-Navigation (optional)	

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

Im **Content** wird später die View mit dem eigentlichen Inhalt der Seite eingeblendet. Views, welche als content genutzt werden, sollten mit „cont“ beginnen.

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

Die **linke 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 linke Navigation genutzt werden, sollten mit "lnav" beginnen, i.d.R. wird es nur genau eine geben. Über die Konfiguration kann eingestellt werden, dass die linke Navigation fixed dargestellt werden kann.

Die **rechte 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 rechte Navigation genutzt werden, sollten mit "rnav" beginnen, z.B. "rnavMain". Auf unterschiedlichen Seiten können verschiedene Navigationen 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.

### 4.1. Raster- und Grid-Größen

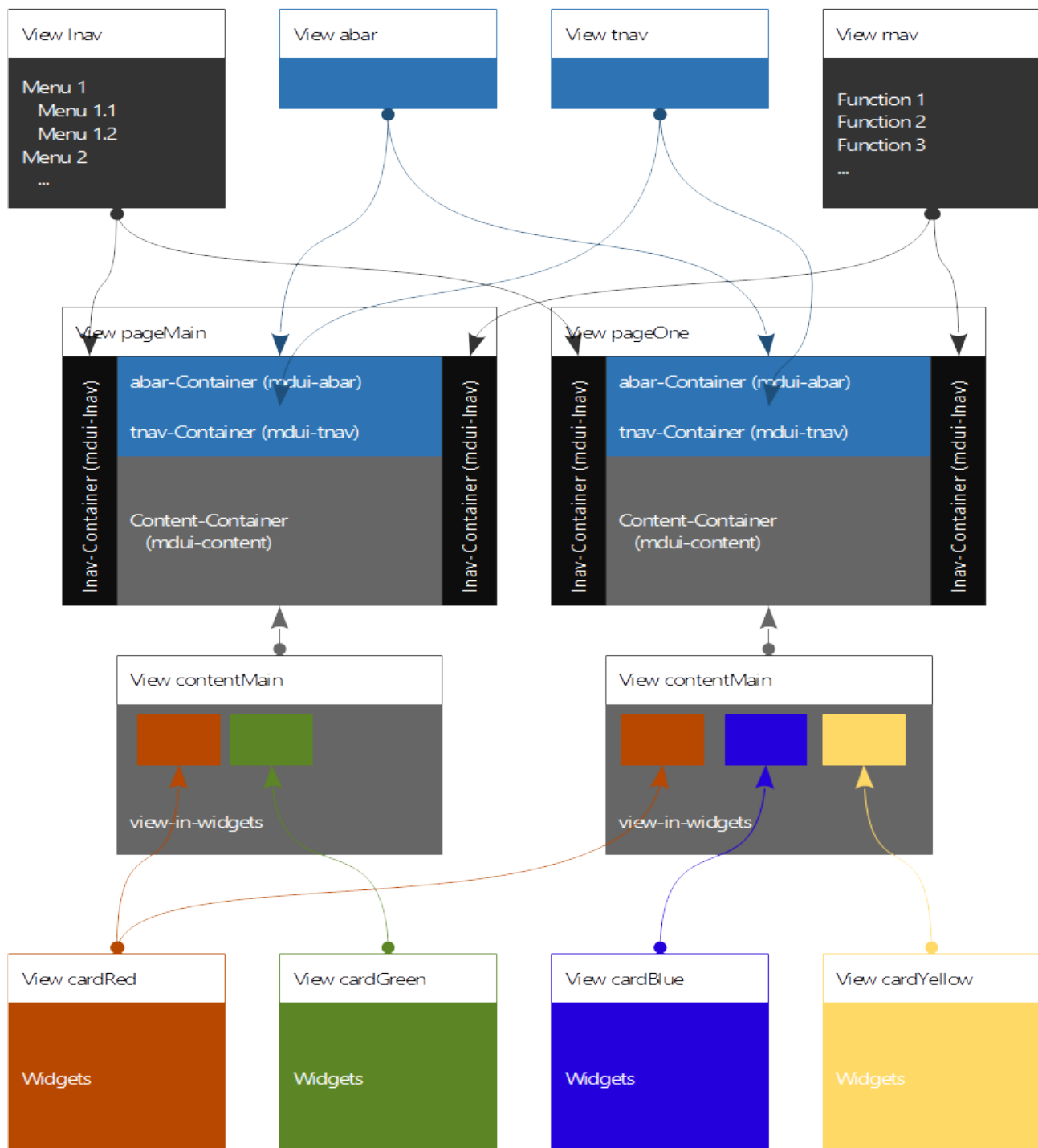
Zwar kann man das Material Design mit allen Rastergrößen verwenden, jedoch basieren die Beispiele auf einem 8 Pixel Raster und einem 80 Pixel Grid. Wenn mit den **mdui-cols-X** gearbeitet wird, sind dieses Vielfache von 80 Pixel. Damit ist leicht ein Design zu realisieren, welches auch auf Handys noch gut zu bedienen ist (die haben häufig eine physikalische Auflösung von 1280x720 Pixeln, was bei double density dann zu 640x360 bzw 360x640 Pixel führt).

Empfohlenen Größen in den page-Views ohne bnav:

abar Höhe:40px, tnav Höhe:40px, content Height:calc(100% - 80px)

Empfohlenen Größen in den page-Views mit bnav:  
abar Höhe:40px, tnav Höhe:40px, bnav Höhe:40px, content Height:calc(100% - 120px)

## 4.2.Layout-Zeichnung



## 5.Application bar

### 5.1.abar-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. Die Höhe sollte 40px 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
<b>Ja</b>	CSS Klasse	<code>mdui-lnavbutton</code>	

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-rnavbutton</code>	

### 5.2.abar Container

Auf jeder Seite (page-View), 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 40px haben.

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	Width Height	<code>100%</code> <code>40px</code>	Kann auch ein fester Wert sein, wenn man fix für eine Ausgabebreite designed, z.B. 1280px
<b>Ja</b>	CSS Klasse	<code>mdui-abar</code>	

## 6.Top-Navigation, Bottom-Navigation

### 6.1.tnav-View und bnav-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 40px betragen.

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

Muss?	Eigenschaft	Werte	Beschreibung
	CSS Klasse	<code>mdui-navbutton</code>	Navigationsschaltfläche
	CSS Klasse	<code>mdui-order-(n)</code>	Hierüber kann die Reihenfolge festgelegt werden, wenn in dem <b>basic-view in widget Container</b> Widget, wo diese View eingebunden wird, <code>mdui-flex</code> für das automatische links Anordnen gesetzt wird. n:1..99
	Text	„ <code>Buttontext</code> “	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 <code>mdui-show480</code> und <code>mdui-hide480</code> zu nutzen. Bsp: <code>&lt;span class="mdui-show480"&gt;EG&lt;/span&gt; &lt;span class="mdui-hide480"&gt;Erdgeschoß&lt;/span&gt;</code>
	CSS Klasse	<code>mdui-noflex</code>	So müssen Widgets gekennzeichnet werden, wenn diese aus dem <code>mdui-flex</code> Anordnung heraus genommen werden sollen. ZB solche, die über die Eigenschaft <code>Links:calc(100% - 80px)</code> fest rechts angeordnet werden

#### 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.tnav-/bnav-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 40px haben.

#### Top-Navigation:

Muss?	Eigenschaft	Werte	Beschreibung
-------	-------------	-------	--------------

	Width Height Top Left	100% 40px 40px 0px	
<b>Ja</b>	CSS Klasse	<code>mdui-tnav</code>	Top-Navigation
		<code>mdui-flex</code>	Führt dazu, dass die Widgets des eingebundenen Views automatisch links angeordnet werden

#### Bottom-Navigation:

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left	100% 40px <code>calc(100% - 40px)</code> 0px	
<b>Ja</b>	CSS Klasse	<code>mdui-bnav</code>	Top-Navigation
		<code>mdui-flex</code>	Führt dazu, dass die Widgets des eingebundenen Views automatisch links angeordnet werden



## 7.Content

### 7.1.content-View

Hierbei handelt es sich Views, der später in den content Containern verwendet werden und den echten Inhalt der Seiten bekommen. Zu jedem page-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 Anordnen des Inhalts möglich ist.

Die in diesem View eingefügten **basic-view in widget Container** (die jeweils selbst ein card-View enthalten) können zugewiesen werden:

Muss?	Eigenschaft	Werte	Beschreibung
(ja)	CSS Klasse	<b>mdui-card</b> <b>mdui-card-outlined</b> <b>mdui-card-raised</b>	Darstellung des Containers als „card“, ohne weitere Angabe flach, ohne Rahmen  -outlined : Mit einem Rahmen versehen -raised: Mit einem Schatten versehen
	CSS Klasse	<b>mdui-order-(n)</b>	Hierüber kann die Reihenfolge festgelegt werden, wenn in dem content-Container, wo diese View eingebunden wird, <b>mdui-flex</b> für das automatische links Anordnen gesetzt wird. n: 1..99
	CSS Klasse	<b>mdui-(color)-bg</b>	Kann gesetzt werden. Allerdings sollte beachtet werden, dass die Schriftenfarbenautomatik hierauf keine Rücksicht nimmt.
	CSS Klasse	<b>mdui-cols-(n)</b> <b>mdui-cols-(n)-toc-(m)</b>	Alternative Angabe der Breite unter Nutzung des Grid-Systems. n=1 → 80 Pixel, n=2 → 160 Pixel usw. n im Bereich 1 bis 24. Wird auch -toc-(m) mit angegeben, so ist die Breite nicht fixiert, sondern wird je nach Platz zwischen n*80 und m*80 Pixel variieren. Bsp: mdui-cols-4-toc-8 → 320 – 640 Px
	CSS Klasse	<b>mdui-rows-(n)</b> <b>mdui-rows-(n)-tor-(m)</b>	Alternative Angabe der Höhe unter Nutzung des Grid-Systems. n=1 → 80 Pixel, n=2 → 160 Pixel usw. n im Bereich 1 bis 24. Wird auch -tor-(m) mit angegeben, so ist die Höhe nicht fixiert, sondern wird je nach Platz zwischen n*80 und m*80 Pixel variieren. Bsp: mdui-rows-4-tor-8 → 320 – 640 Px Hinweis: Damit die Höhe variabel berechnet wird, muss im content-Container <b>mdui-flex-stretch</b> gesetzt werden.
	CSS Klasse	<b>mdui-noflex</b>	So müssen Widgets gekennzeichnet werden, wenn diese aus dem <b>mdui-flex</b> Anordnung heraus genommen werden sollen.
	CSS Klasse	<b>mdui-title</b>	Die Darstellung erfolgt mit einer abgedunkelten Hervorhebung des Titelbereiches (oberen 64

			Pixel)
	CSS Klasse	<code>mdui-title-(color)-bg</code>	Farbangabe für die Einfärbung des Titelbereiches

## 7.2. content-Container

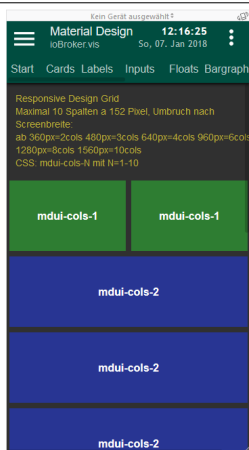

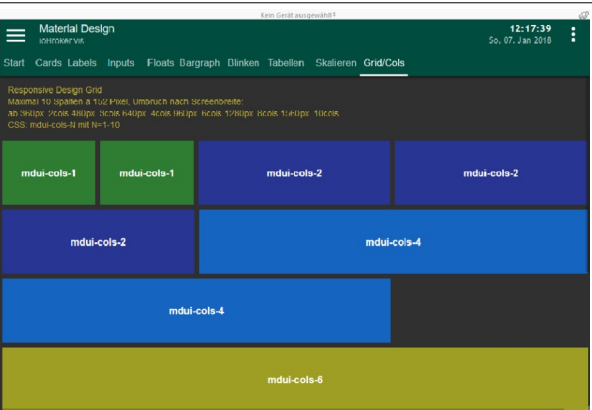
Auf jeder Seite gibt es genau ein Widget ***basic-view in widget Container*** welches den Content View darstellen wird.

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left	100% <code>calc(100% - 80px)</code> 80px 0px	Wenn mit Bottom-Navigation, dann ist Height: <code>calc(100% - 120px)</code>
<b>Ja</b>	CSS Klasse	<code>mdui-content</code>	
	CSS Klasse	<code>mdui-flex</code>	Die im eingebundenen View liegenden Widgets werden im flex-Verfahren zeilenweise automatisch links angeordnet. Den Widgets zugewiesenen <code>mdui-cols-(n)-toc-(m)</code> und <code>mdui-order-(n)</code> Werte werden berücksichtigt.
	CSS Klasse	<code>mdui-flex-stretch</code>	Die im eingebundenen View liegenden Widgets werden zugewiesenen <code>mdui-rows-(n)-tor-(m)</code> Werte werden berücksichtigt.

## 7.3. Grid-Struktur, Responsive Design

Möchte man ein echtes responsive Design erreichen, so kann dieses über die Nutzung der Grid-Struktur erreicht werden. Hierbei wird den Widget ***basic-view in widget Containern*** in den contViews keine Width direkt zugeordnet, sondern nur eine Zahl, wie viel Grid-Spalten das Widget breit sein soll.

Jede Grid-Spalte ist dabei 80px breit, für die Spaltenanzahl kann ein von und ein bis-Wert angegeben werden. Soweit möglich werden die Zeilen immer aufgefüllt. Widgets, mit denen die Spaltenanzahl überschritten wird, werden in die nächste Zeile umgebrochen.

 <p>360px → 2 Spalten</p>	 <p>640px → 4 Spalten</p>	 <p>960px → 6 Spalten</p>

Den Widget **basic-view in widget Container** in den cont-Views können zugewiesen werden:

Muss?	Eigenschaft	Werte	Beschreibung
	CSS Klasse	<code>mdui-cols-(n)</code> <code>mdui-cols-(n)-toc-(m)</code>	Alternative Angabe der Breite unter Nutzung des Grid-Systems. $n=1 \rightarrow 80$ Pixel, $n=2 \rightarrow 160$ Pixel usw. $n$ im Bereich 1 bis 24. Wird auch <code>-toc-(m)</code> mit angegeben, so ist die Breite nicht fixiert, sondern wird je nach Platz zwischen $n*80$ und $m*80$ Pixel variieren. Bsp: <code>mdui-cols-4-toc-8</code> $\rightarrow 320 - 640$ Px

**TIPP**

Um eine sinnvolle Darstellung auch auf Handys zu erreichen (Hochkant i.d.R. 360px Breite), sollte `mdui-cols-4`, was 320px entspricht, verwendet werden.

## 8. Linke Navigation, rechte Navigation

### 8.1. Inav-Views, rnav-Views

Hierbei handelt es sich um Views, die später in den Left/Right-Navigation Containern auf den page-Views verwendet werden. Normalerweise werden hier Navigationsschaltflächen untergebracht, welcher der internen und externen Navigation dienen. Views, die als Left/Right-Navigation genutzt werden, sollten mit "Inav" bzw. „rnav“ beginnen, z.B. "InavMain". Als Widgets bieten sich an: basic-HTML, jquery-navigation-Icon, ...

Den eingefügten Widgets können die folgenden Werte zugewiesen werden:

Muss?	Eigenschaft	Werte	Beschreibung
	CSS Klasse	<code>mdui-navitem</code>	Weist das Widget als Navigationsitem aus.
	CSS Klasse	<code>mdui-order-(n)</code>	Hierüber kann die Reihenfolge festgelegt werden, wenn in dem Inav-Container (in den page-Views), wo diese View eingebunden wird, <code>mdui-flex</code> für das automatische oben Anordnen gesetzt wird. (n): 1..99
	CSS Klasse	<code>mdui-noflex</code>	So müssen Widgets gekennzeichnet werden, wenn diese aus dem <code>mdui-flex</code> Anordnung heraus genommen werden sollen. ZB wenn sie fest rechts oben angeordnet werden sollen.
	Width	(n)%	Normalerweise wird n=100, also 100% angegeben damit jedes Widget eine Zeile belegt. Möchte man zB zwei Widgets in einer Zeile nebeneinander haben, wird n=50, also 50% angegeben.

#### 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 „19 Baumstruktur-Menüs in der Inav, rnav“

### 8.2. Inav-, rnav-Container

Auf jeder Seite (page-View), 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 Height Top Left		Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am linken Rand mit 250 Px Breite angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden.

<b>Ja</b>	CSS Klasse	<code>mdui-lnav</code>	Linke Navigation
	CSS Klasse	<code>mdui-flex</code>	Die im eingebundenen View liegenden Widgets werden im flex-Verfahren automatisch oben angeordnet. Den Widgets zugewiesenen <code>mdui-order-(n)</code> Werte werden berücksichtigt.

### Right-Navigation Container

Muss?	Eigenschaft	Werte	Beschreibung
	Width Height Top Left		Egal, da diese Werte nur für den Designer gelten, zur Laufzeit werden die Container automatisch am rechten Rand mit 250 Px Breite angezeigt. Man kann also auch eine (nicht störende) Größe von 48x48px verwenden.
<b>Ja</b>	CSS Klasse	<code>mdui-rnav</code>	Rechte Navigation
	CSS Klasse	<code>mdui-flex</code>	Die im eingebundenen View liegenden Widgets werden im flex-Verfahren automatisch oben angeordnet. Den Widgets zugewiesenen <code>mdui-order-(n)</code> Werte werden berücksichtigt.

## 9.card-Views

Cards-Views dienen dem thematischen Gruppieren von Widgets. Sie werden anschließend im content-View in **basic-view in widget Container** Widgets abgerufen und dort mit CSS Klassen versehen um ihr Aussehen fest zu legen (siehe im Kapitel content-View).

### Titel

Sollen Cards Titel und Untertitel haben, so können hierfür die unter „Labels“ beschriebenen CSS-Styles **mdui-title** und **mdui-subtitle** verwendet werden.

### Collapse / Expand

Nutzt man eine Schaltfläche mit der Zuordnung **mdui-expand** so agiert diese zum „Kollabieren“ (collapse) bzw. „Expandieren“ des card-Views zur Laufzeit. Im kollabierten Zustand haben die card.Views nur noch eine Höhe von 72px. Als Symbol sollte `<i class="material-icons">expand_less</i>` verwendet werden.

#### Hinweis

Dieses funktioniert nur, wenn die card-View selbst in einem „view in widget“ Widget verwendet wird und dieses die CSS Klasse **mdui-card** bzw. **mdui-card-outlined** bzw. **mdui-card-raised** zugewiesen bekommen hat.

### Fullscreen

Nutzt man eine Schaltfläche mit der Zuordnung **mdui-fullscreen** so agiert diese zum „Vollbild“ bzw. „Normalbild“ des card-Views zur Laufzeit. Im Vollbild nimmt die card-View dann die komplette Browserfläche ein. Sinnvoll z.B. wenn im card-View Bilder oder Tabellen angezeigt werden. Als Symbol sollte `<i class="material-icons">fullscreen</i>` verwendet werden.

#### Hinweis

Dieses funktioniert nur, wenn die card-View selbst in einem „view in widget“ Widget verwendet wird und dieses die CSS Klasse **mdui-card** bzw. **mdui-card-outlined** bzw. **mdui-card-raised** zugewiesen bekommen hat.



## 10.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.

### **mdui-title**

Leicht vergrößerte Schrift, für Überschriften

### **mdui-subtitle**

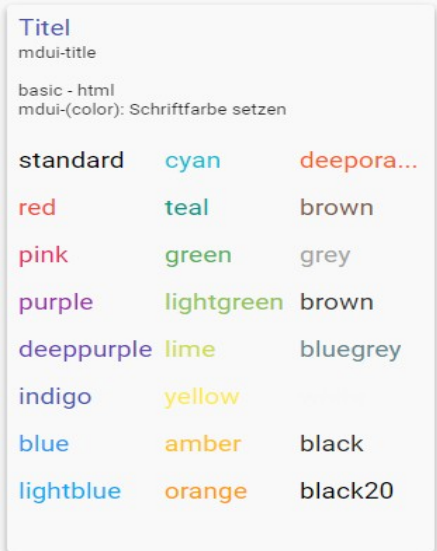
Verkleinerte Schrift, für Unter-Überschriften und Erläuterungen

### **mdui-label**

Normale Schrift

### **mdui-value**

Nomale, fette Schrift, für die Darstellung von Werten



Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	mdui-label oder mdui-value oder mdui-title oder mdui-subtitle	Label-Text Wert Titel, Überschrift 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 zugegriffen 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}**

## 11.States

Für die Darstellung von States wird das **basic-bool HTML** und das **basic-ValueList HTML** Widget verwendet. Es kann sowohl für die Darstellung von Status-Texten als auch von Status-Symbolen verwendet werden. Bei der Textdarstellung kann entweder der Text selbst oder der Hintergrund in verschiedenen Farben gefärbt werden. Weiterhin kann die Aufmerksamkeit durch die Verwendung von blink/flash/pulse Effekten erhöht werden.

Bisher wurden die CSS Klassen immer nur dem Widget selbst über die Eigenschaft „CSS“ zugewiesen. Dieses Mal nicht nur, da ja in Abhängigkeit eines Wertes eine unterschiedliche Widget-Darstellung erreicht werden soll. Die CSS Klassen werden also auch direkt in der Widget Eigenschaft „Werteliste“ verwendet.

In der Eigenschaft „Werteliste“ wird für jeden möglichen Wert eine HTML-Anweisung für die Darstellung angegeben. Als Trennzeichen für die Anweisungen wird das „;“ verwendet, die erste Anweisung wird für den Wert 0, die zweite für den Wert 1 usw. genutzt.

### Statustexte

mdui-state

basic ValueList, basic bool HTML

Simuliere Status: 0 1 2













	Farbangaben in CSS Klasse (nicht state-abhängig)	Farbangaben in Werteliste (state-abhängig)
ohne	 geschl.	 geschl.
Font mdui-(color)	 geschl.	 geschl.
Background mdui-(color)-bg	 geschl.	 geschl.

### Statustexte (outlined)

mdui-state-outlined

basic ValueList, basic bool HTML

Simuliere Status: 0 1 2

	Farbangaben in CSS Klasse (nicht state-abhängig)	Farbangaben in Werteliste (state-abhängig)
ohne	 geöffnet	 geöffnet
Font mdui-(color)	 geöffnet	 geöffnet
Outline mdui-(color)-ol	 geöffnet	 geöffnet
Font und -ol	 geöffnet	 geöffnet
Background mdui-(color)-bg	 geöffnet	 geöffnet
-bg und -ol	 geöffnet	 geöffnet

Zum besseren Verständnis folgen nun Beispiele, welche alle die States (geschlossen, gekippt, geöffnet) eines Fenstersensors:

### BEISPIEL 1

Darstellung mit farbigem Hintergrund:



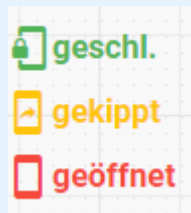


Eigenschaft „Werteliste“:

```
<div class="mdui-green-bg"><i class="mdui-icon">phonelink_lock</i>geschl.</div>
<div class="mdui-amber-bg"><i class="mdui-icon">mobile_screen_share</i>gekippt</div>
<div class="mdui-red-bg"><i class="mdui-icon">smartphone</i>geöffnet</div>
```

## BEISPIEL 2

Darstellung mit Symbolen und farbiger Schrift:



Eigenschaft „Werteliste“:

```
<div class="mdui-green"><i class="mdui-icon">phonelink_lock</i>geschl.</div>
<div class="mdui-amber"><i class="mdui-icon">mobile_screen_share</i>gekippt</div>
<div class="mdui-red"><i class="mdui-icon">smartphone</i>geöffnet</div>
```

Muss?	Eigenschaft	Werte	Beschreibung
ja	CSS Klasse	mdui-state mdui-state-oulined	Darstellung als Statuswert
	Werteliste		Siehe oben stehende Beschreibung und Beispiele

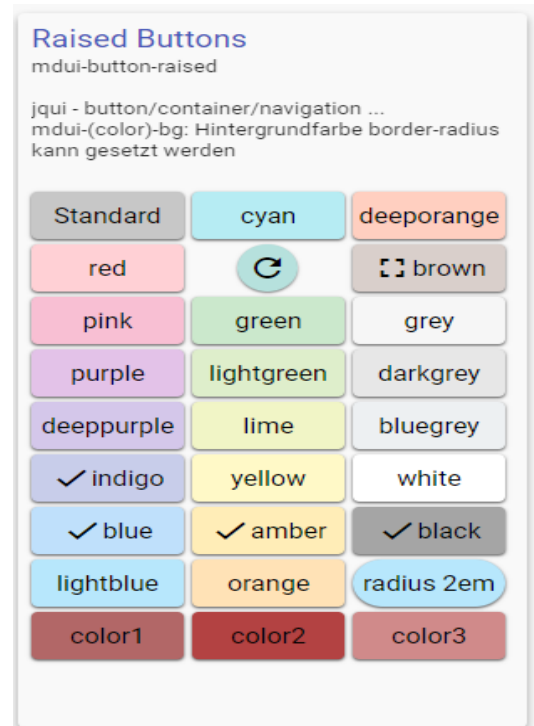
## 12.Buttons

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.

Buttons können aus Icons, Texten und Icons+Texten bestehen. Wenn Icons verwendet werden sollen, sind diese leicht durch den WebFont anzugeben.

Bsp: `<i class="mdui-icon">check</i>OK`

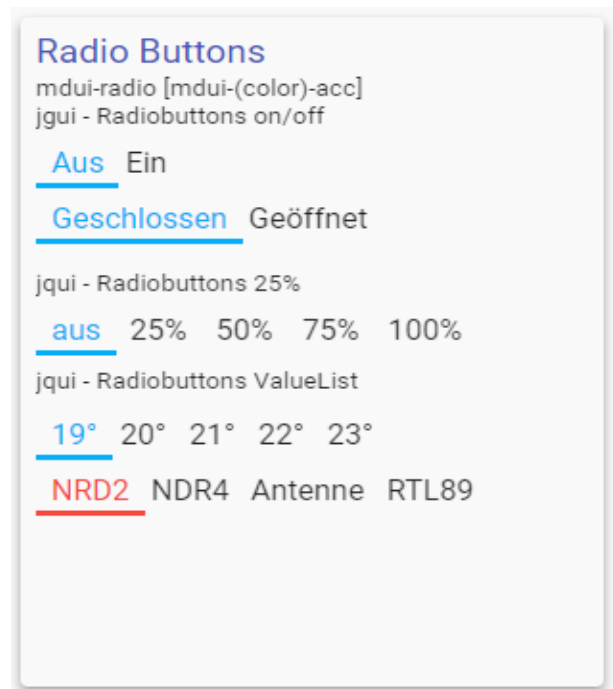
Weiterhin kann man über die Widget Eigenschaften „border-radius“ auch abgerundete oder runde Buttons (z.B. mit 1em) erzeugen.



Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-button mdui-button-raised mdui-button-outlined	
	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, fatter Text. Die jeweils aktive Option wird in der Akzentfarbe dargestellt und unterstrichen.

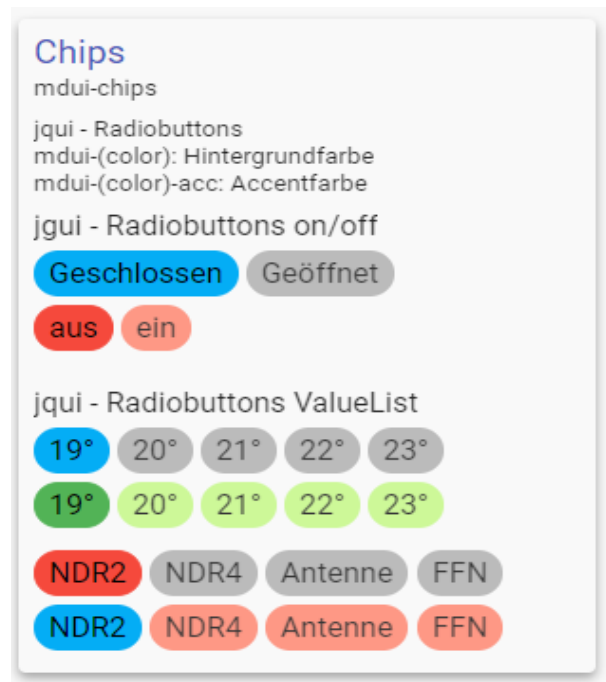


Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	<code>mdui-radio</code>	
	CSS Klasse	<code>mdui-(color)</code>	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	<code>mdui-(color)-bg</code>	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein,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

## 14.Chips

Wie für die Radio-Eingaben werden auch für die Chips eines der **jqui-Radio...** Widgets verwendet. Die Darstellung der wählbaren Optionen erfolgt als abgerundete Schaltflächen. Die jeweils aktive Option wird in der Akzentfarbe dargestellt und unterstrichen.

Neben den gefüllten Chips stehen auch solche mit Rahmenlinie zur Verfügung.



Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	mdui-chips mdui-chips-outlined	
	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“.

## 15.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.

### Input-Typ

Die vis kennt beim jgui-input eigentlich nur den Typ **string** als Texteingabe. Über eine MDCSS Klasse können aber auch andere Typen gesetzt werden, so dass dann abhängig vom Betriebssystem andere Eingabeoptionen geöffnet werden. Wie z.B. eine Farbauswahl oder eine nur Zifferneingabe. Als Rückgabe wird aber immer ein string gegeben. So liefern z.B. Farbauswahlen die Auswahl in der Form #rrggbb zurück.

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-input</code>	
	CSS Klasse	<code>mdui-(color)</code>	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	<code>mdui-(color)-bg</code>	Hintergrundfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "transparent".
	CSS Klasse	<code>mdui-input-number</code>	Die Eingabe beschränkt sich auf die Erfassung von Ziffern. Unter Android öffnet sich z.B. ein anderes Tastaturfeld.
	CSS Klasse	<code>mdui-input-time</code>	Die Eingabe beschränkt sich auf die Erfassung von Zeiten. Unter Android öffnet sich z.B. ein Zeitfeld.
	CSS Klasse	<code>mdui-input-date</code>	Die Eingabe beschränkt sich auf die Erfassung eines Datums. Unter Android öffnet sich z.B. ein Kalenderfeld.
	CSS Klasse	<code>mdui-input-color</code>	Es kann eine Farbe ausgewählt werden. Das Ergebnis wird als string der Form #rrggbb zurück gegeben.

## 16.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.



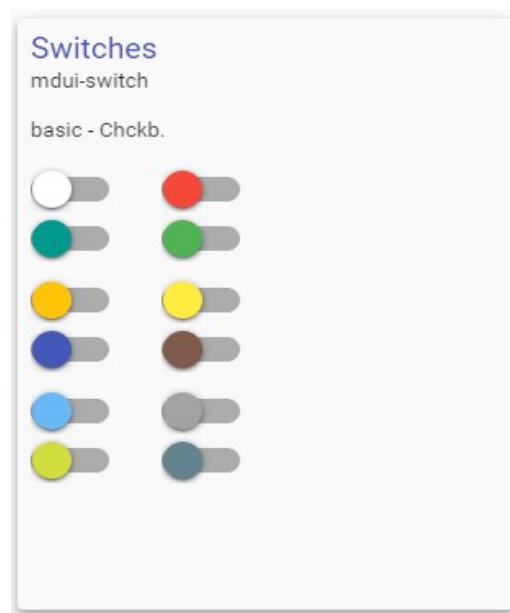
Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-select</code>	
	CSS Klasse	<code>mdui-(color)</code>	Schriftfarbe. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	<code>mdui-(color)-bg</code>	Hintergrundfarbe. (color) kann eine der Farben (siehe dort)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

### WICHTIG

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

## 17.Switches

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



Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	mdui-switch	
<b>Ja</b>	HTML anhängen	<label for="{wid}_checkbox"> <label>	Siehe <b>Wichtig!</b>
	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

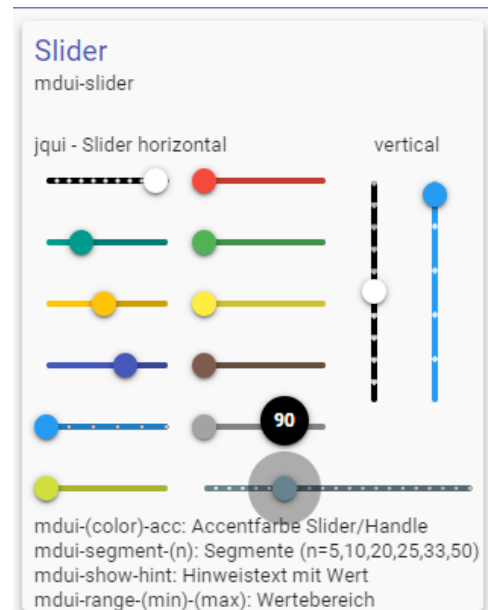
Damit der Switch funktioniert, muss in der Widget-Eigenschaft **HTML anhängen** der Wert **<label for="{wid}\_checkbox"> <label>** eingegeben werden.

## 18.Slider

### 18.1.Farbige Slider

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

Slider erhalten beim Fokus und bei ihrer Bedienung einen abgedunkelten Touchbereich. Außerdem kann ein Hinweiswert definiert werden, welche während des „slidens“ eingeblendet wird.



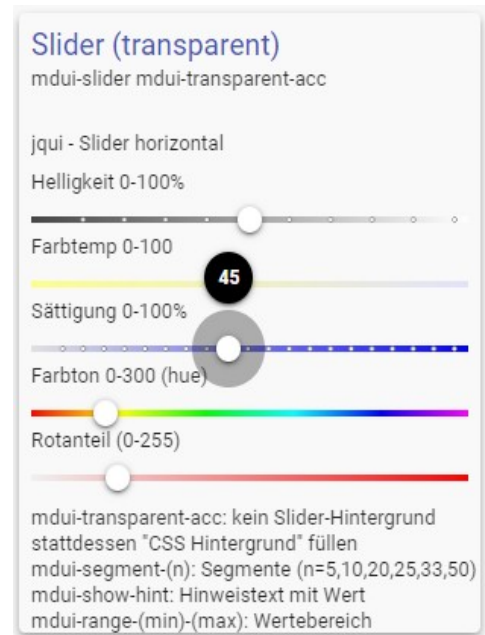
Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-slider	
	CSS Klasse	mdui-(color)-acc	Hiermit kann festgelegt werden, welche Farbe für den Button und den aktiven Bereich verwendet werden soll. (color) kann eine der Farben (siehe dort) sein, Vorgabe ist "weiß".
	CSS Klasse	mdui-segment-(n)	Hierüber kann der slider segmentiert dargestellt werden. Während des „slidens“ rastet er an diesen Werten ein. n gibt die Anzahl der Segmente an. N=5,10,20,25,33,50
	CSS Klasse	mdui-showhint	Während des Slidens wird ein Hinweistext mit dem aktuellen Wert angezeigt. Normalerweise 0..100, über <b>mdui-range-(min)-(max)</b> können andere Werte gesetzt werden.
	CSS Klasse	mdui-range-(min)-(max)	Hierüber kann ein Wertebereich für die Anzeige des Hinweistextes gesetzt werden. Min und max sind durch die Werte zu ersetzen. Diese Angabe muss zusätzlich zu den min/max Werten des Widgets erfolgen, da diese nicht ausgelesen werden können. Bsp: mdui-range-0-360



## 18.2.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)`

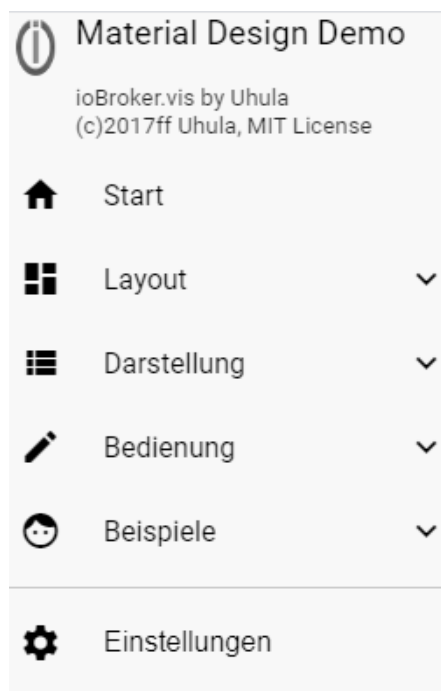
Farbton: `linear-gradient(to right, #ff0000, #ffff00, #00ff00, #00ffff, #0000ff, #ff00ff)`

Muss?	Eigenschaft	Werte	Beschreibung
...	...	...	Wie beim normalen Slider
<b>Ja</b>	CSS Klasse	<code>mdui-transparent-acc</code>	Transparente Darstellung des Silders-Weges, nur das Slider-Handle wird dargestellt. Hierdurch ist es möglich einen beliebigen Hintergrund zu wählen.

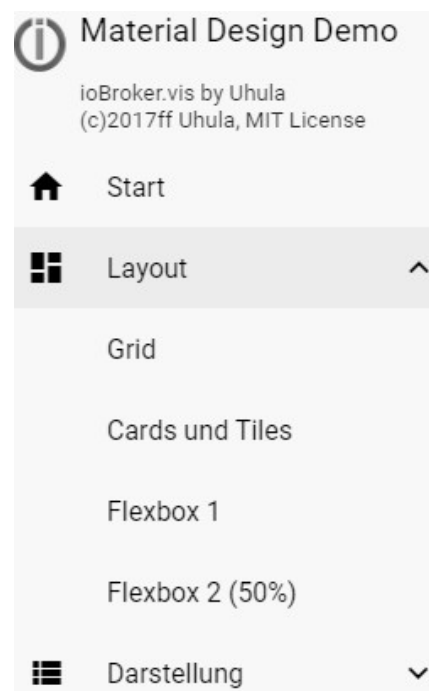
## 19. Baumstruktur-Menüs in der Inav, rnav

Für das Grundprinzip der Inav und rnav siehe Kapitel 8 Linke Navigation, rechte Navigation.

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.



Mit zwei Baumstruktur-Einträgen, alle geschlossen



Nach [Klick] auf „Layout“

Damit sich die Menü-Einträge so verhalten, müssen ihnen spezielle CSS-Klassen zugeordnet werden.

### 19.1. Menü-Eintrag 1.Ebene

Als Widget für einen Menüeintrag der 1.Ebene, der eine 2.Ebene öffnen/schließen wird ein normales **basic-HTML** Widgets verwendet und der Inhalt wird als HTML angegeben. Entweder nur als Text „Menübezeichnung“, oder kombiniert mit einem Icon.

```
<i class="mdui-icon_">home</i>Start
```

Weiterhin sind zusätzlich neben den in Kapitel 8 Linke Navigation, rechte Navigation genannten folgende CSS Klassen zuzuweisen:

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<b>mdui-toggle</b>	Damit wird das Widget als toggle-Widget gekennzeichnet, welches andere Widgets anzeigen/verstecken kann
<b>Ja</b>	CSS Klasse	<b>mdui-group-(name)</b>	Angabe des Gruppennamens, welchen auch die anzuzeigenden/zur 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.

## 19.2.Menü-Eintrag 2.Ebene

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

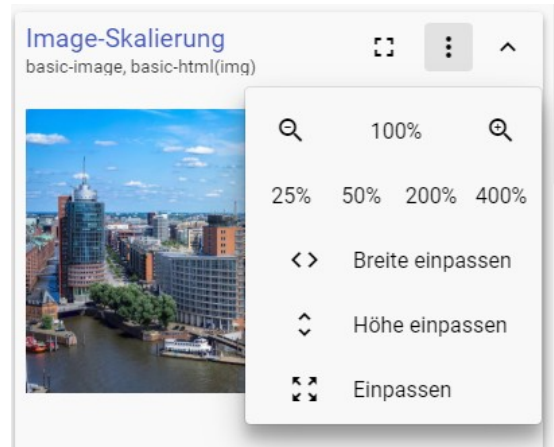
Weiterhin sind zusätzlich neben den in Kapitel 8 Linke Navigation, rechte Navigation genannten folgende CSS Klassen zuzuweisen:

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-group-(name)</code>	Angabe des Gruppennamens, wie im zugehörigen Menü-Eintrag der 1.Ebene
	CSS Klasse	<code>mdui-hide</code>	Kann/Sollte angegeben werden, wenn der Menüeintrag als Vorgabe versteckt sein soll. Wirkt sich nur zur Laufzeit aus, nicht im Designer.

## 20. 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** Widgets 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](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.

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

### 20.1. Skalierungs-Schaltflächen

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	<b>mdui-target-(widgetID)</b>	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	<b>mdui-scale-(type)</b>	Angabe der Skalierung, die angewendet werden soll. (type)= <i>fit</i> =Das <img> wird eingepasst <i>hfit</i> = Das <img> wird horizontal eingepasst <i>vfit</i> = Das <img> wird vertikal eingepasst <i>in</i> = Es wird um den Faktor 1,41 hineingezoomt <i>out</i> = Es wird um den Faktor 1,41 herausgezoomt <i>xxx</i> = Es wird auf xxx% gezoomt Bsp: mdui-scale-100
	CSS Klasse	<b>mdui-button</b>	
	CSS Klasse	<b>mdui-hide</b>	Kann/Sollte angegeben werden, wenn die Schaltfläche als Vorgabe versteckt sein soll. Wirkt sich nur zur Laufzeit aus, nicht im Designer.

### 20.2. 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.

Es bietet sich an hier das bekannte Menü-More-Icon zu verwenden:

```
<i class="mdui-icon">more_vert</i>
```

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

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

### 20.3.Vollbild-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 diese 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.

Sinnvollerweise besteht die Schaltfläche nur aus einem Icon.

Es bietet sich an hier das bekannte Fullscreen-Icon zu verwenden:

```
<i class="mdui-icon">fullscreen</i>
```

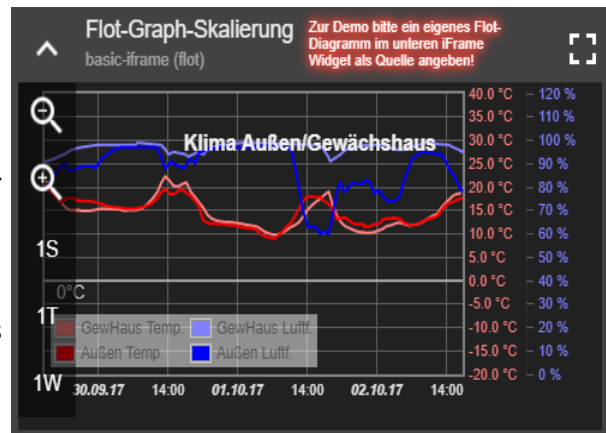
Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-fullscreen</code>	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	<code>mdui-button</code>	

## 21.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** Widgets verwendet. Über die Zuweisung von CSS-Klassen wird ihnen ihre Bedeutung zugewiesen.



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.

### 21.1.Zeitspannen-Schaltflächen

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

### 21.2.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.

Es bietet sich an hier das bekannte Menü-More-Icon zu verwenden:

```
<i class="mdui-icon">more_vert</i>
```

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

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

### 21.3.Vollbild-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.

Sinnvollerweise besteht die Schaltfläche nur aus einem Icon.

Es bietet sich an hier das bekannte Fullscreen-Icon zu verwenden:

```
<i class="mdui-icon">fullscreen</i>
```

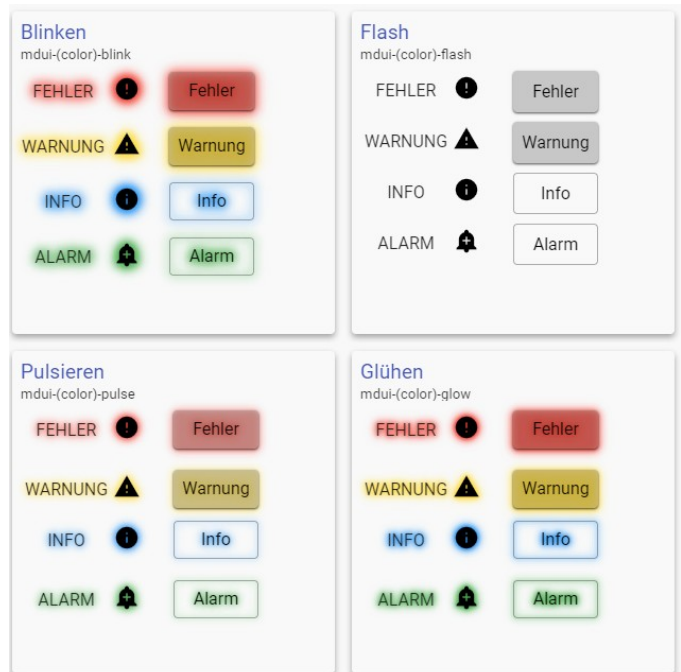
Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion.

Muss?	Eigenschaft	Werte	Beschreibung
<b>Ja</b>	CSS Klasse	<code>mdui-fullscreen</code>	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	<code>mdui-button</code>	

## 22. Glühen - Flashen – Blinken – Pulsieren

Manchmal ist es notwendig, dass man Texte / Symbole / Stati optisch auffälliger gestalten möchte. Hierzu stehen vier verschiedene Funktionen mit jeweils vier verschiedenen Farben zur Verfügung.

Die CSS-Klassen können im Prinzip jedem Widget zugewiesen werden und versehen dieses mit dem gewünschten Rahmen.



Über die Zuweisung von CSS-Klassen erhält dieses Widget seine Funktion. Es kann immer nur genau eine der aufgeführten CSS-Klasse zugewiesen werden.

Als (color) sind blue, red, green und yellow möglich.

Muss?	Eigenschaft	Werte	Beschreibung
	CSS Klasse	mdui-(color)-glow	Das Widget wird „glühend“ dargestellt. D.h. es erhält einen starren, nicht animierten farbigen Rahmen
	CSS Klasse	mdui-(color)-blink	Das Widget wird „blinkend“ dargestellt. D.h. es erhält einen farbigen Rahmen, welcher im 1 Sek Rhythmus blinkt
	CSS Klasse	mdui-(color)-flash	Das Widget wird „blitzend“ dargestellt. D.h. es erhält einen farbigen Rahmen, welcher im 1 Sek Rhythmus blitzt
	CSS Klasse	mdui-(color)-pulse	Das Widget wird „pulsierend“ dargestellt. D.h. es erhält einen farbigen Rahmen, welcher im 4 Sek Rhythmus pulsiert

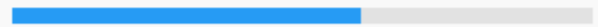


## 23. Bargraphanzeigen

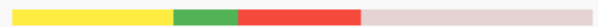
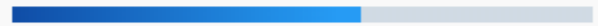
Die Anzeige von Bargraphen kann dazu verwendet werden um Stati / Werte in Balkenform anzuzeigen. Dabei besteht die Möglichkeit die Anzeige sowohl farblich als auch in der Form zu manipulieren. Für die Darstellung werden die **basic-bar** Widgets verwendet.

### Balkenfarben

durch Balken-color feste Balkenfarbe



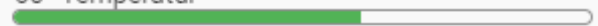
durch background Balkenfarbe nimmt Hintergrundfarbe an



durch Balken-color Balkenfarbe ändert sich je nach Value



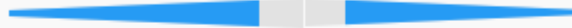
60° Temperatur



### Horizontal

mdui-h-bargraph

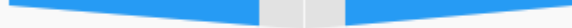
+ mdui-triangle ++ mdui-h-flip



+ mdui-ramp



+ mdui-triangle mdui-v-flip



+ mdui-segment-10



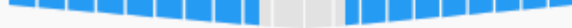
+ mdui-triangle mdui-segment-10



+ mdui-ramp mdui-segment-10

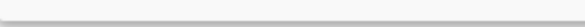
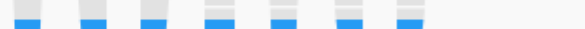
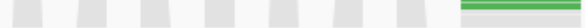
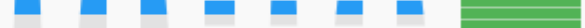
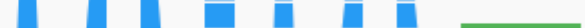


+ mdui-triangle mdui-v-flip mdui-segment-10



### Vertikal

mdui-v-bargraph



Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	mdui-h-bargraph mdui-v-bargraph	h = horizontal v = vertikal
	Farbe	#rrggbb	Eine direkte Farbangabe #rrggbb

	background	...	Siehe Erläuterung: Farben
	CSS Klasse	<code>mdui-(color)-pulse</code>	Das Widget wird „pulsierend“ dargestellt. D.h. es erhält einen farbigen Rahmen, welcher im 4 Sek Rhythmus pulsiert
	CSS Klasse	<code>mdui-triangle</code>	Darstellung in Dreiecksform
	CSS Klasse	<code>mdui-ramp</code>	Darstellung in Rampenform
	CSS Klasse	<code>mdui-segment-10</code>	Darstellung segmentiert, 10 Segmente
	CSS Klasse	<code>mdui-h-flip</code>	Darstellung wird horizontal gespiegelt, baut sich also von rechts nach links auf
	CSS Klasse	<code>mdui-v-flip</code>	Darstellung wird vertikal gespiegelt, baut sich also von unten nach oben auf

## 23.1.Farben

### Eigenschaft: CSS Hintergrund | background

Die Widget „CSS Hintergrund | background“ Eigenschaft kann genutzt werden um einen festen Farbverlauf, eine Farbabstufung oder eine Farbabstufung mit Verlauf festzulegen.

#### Farbverläufe

Blauer Verlauf: `linear-gradient(to right, #0D47A1 0px, #2196F3 160px )`

Gelb-Grün-Rot : `linear-gradient(to right, #FFEB3B 0px, #FFEB3B 80px, #4CAF50 80px, #4CAF50 112px, #F44336 112px )`

Gelb-Grün-Rot Verlauf: `linear-gradient(to right, #FFEB3B 0px, #4CAF50 80px, #F44336 160px )`

### Eigenschaft: Allgemein | Farbe

Wird hier die Farbe statt im background angegeben, kann darüber festgelegt werden, dass der Balken je nach Wert seine Farbe ändert. Hierzu wird die binding-Fähigkeit der vis genutzt um auf den darzustellenden Wert zuzugreifen und zu interpretieren.

Gelb bis 49, Grün 50-69, Rot ab 70: `{v:0_userdata.0.myvalue;v < 50 ? "#FFEB3B" :: v < 70 ? "#4CAF50" :: "#F44336"}`

## 24.Tabellen

HTML-Tabellen werden zur Darstellung von Terminen, Benachrichtigungen, Logbucheinträgen usw. verwendet. Damit die Tabellendarstellung dem Material Design entspricht, sollte der Tabelle/dem Widget die CSS Klasse `mdui-table` zugewiesen werden.

Dieses kann entweder dem Widget, in welchem sich die Tabelle befindet, oder, wenn die HTML-Tabelle mit `<table> </table>` selbst aufgebaut wird, mit `<table class="mdui-table">` dem table-Element zugewiesen werden. Die HTML-Tabellen sollten einfach strukturiert sein und keine oder nur wenige eigene style-Attribute in ihren Elementen haben.

Sollen die Tabellenzeilen nicht tabellarisch, sondern in card- oder Listform angezeigt werden, so ist die entsprechende CSS Klasse zu setzen. In der card-Form ordnen sich alle Zeilen als card linksbündig an und erhalten – damit es besser aussieht – alle die gleiche Höhe. Die Listform wird immer als eine Spalte dargestellt.

### Darstellung der normalen Tabelle:

Termin	Kalender	Betreff
22.09.2017 15:00	Privat	Kaffeetrinken mit Adele, Beta und Christine
22.09.2017 16:30	Beruf	Abgabe Planung 13
22.09.2017 18:00	Privat	Elternratsversammlun Am Schulweg 4, Dieter mitnehmen
23.09.2017 20:00	Müll	Restmüll
23.09.2017 20:00	Müll	Bio-Tonne
24.09.2017 8:00	Beruf	Vorbesprechung Carportbau, Schönefeld

### Darstellung als List:

22.09.2017 15:00 Kalender Privat Betreff Kaffeetrinken mit Adele, Beta und Christine
22.09.2017 16:30 Kalender Beruf Betreff Abgabe Planung 13
22.09.2017 18:00 Kalender Privat Betreff Elternratsversammlung, Am Schulweg 4, Dieter
23.09.2017 20:00 Kalender Müll

### Darstellung als Cards - raised

22.09.2017 15:00 Privat Kaffeetrinken mit Adele, Beta und Christine	22.09.2017 16:30 Beruf Abgabe Planung 13
22.09.2017 18:00 Privat Elternratsversammlun Am Schulweg 4, Dieter mitnehmen	23.09.2017 20:00 Müll Restmüll
23.09.2017 20:00 Müll Bio-Tonne	24.09.2017 8:00 Beruf Vorbesprechung Carportbau, Schönefeld

### Darstellung als Cards – outlined

### Tabelle (outlined)

mdi-table-outlined

22.09.2017 15:00

Privat

Kaffeetrinken mit Adele, Beta und Christine

22.09.2017 16:30

Beruf

Abgabe Planung 13

22.09.2017 18:00

Privat

Elternratsversammlung, Am Schulweg 4, Dieter

23.09.2017 20:00

Müll

Restmüll

23.09.2017 20:00

Müll

Bio-Tonne

24.09.2017 8:00

## HINWEIS

Da die „Umformung“ über Javascript-Ereignisse erfolgt, steht die Anzeige erst zur Laufzeit und nicht bereits im Editor zur Verfügung.

Bei den Darstellungen als Card und Tile können über die CSS Klassen weitere Optionen, wie z.B. die Breite der einzelnen Cards/Tiles gesetzt werden.

Muss?	Eigenschaft	Werte	Beschreibung
Ja	CSS Klasse	<code>mdui-table</code>	Sorgt für die Darstellung der HTML-Tabelle im Material Design Stil.
	CSS Klasse	<code>mdui-table-bordered</code>	Zwischen den Tabellenzeilen wird ein Trennstrich gezeichnet
	CSS Klasse	<code>mdui-table-striped</code>	Jeder 2. Tabellenzeile wird leicht aufgehellt dargestellt. Diese Funktionalität bleibt auch erhalten, wenn die Tabelle in Card- bzw. Tile-Form angezeigt wird.
	CSS Klasse	<code>mdui-table-outlined</code>	Die Tabellenzeilen werden als Cards mit Rahmen dargestellt. Erlaubte Optionen: <code>-r -w -c -l</code> (siehe unten) Bsp: <code>mdui-table-astile-c3-l</code>
	CSS Klasse	<code>mdui-table-raised</code>	Die Tabellenzeilen werden als Cards mit Schatten dargestellt. Optionen: Erlaubte Optionen: <code>-r -w -c -l</code> (siehe unten) Bsp: <code>mdui-table-ascard-w200</code>
	CSS Klasse	<code>mdui-table-list</code>	Die Tabellenzeilen werden als Liste dargestellt, die Spalten der Zeile werden untereinander aufgeführt. Erlaubte Optionen: <code>l</code> (siehe unten) Bsp: <code>mdui-table-aslist-l</code>
	CSS Klasse	<code>mdui-table-opt-...</code>	Hierüber können weitere Optionen angegeben werden, siehe Erläuterung der Optionen

## Erläuterung der Optionen

Die Optionen werden der CSS Klasse „mdui-table-opt-“ einfach angehängt. Bsp: „mdui-table-opt-r800-w180“

Option	Beschreibung
<b>-rNNNN</b>	Angabe einer Responsive-Breite in Pixel, bei deren Unterschreitung die CSS Klasse aktiviert werden soll. Siehe auch nächstes Kapitel. Bsp: <code>-r800</code> Erst wenn weniger als 800 Pixel Breite zur Anzeige zur Verfügung stehen, wird die CSS Klasse angewendet
<b>-wNNNN</b>	Angabe der festen Breite einer Card/eines Tiles in Pixel. Die Cards/Tiles werden immer mit dieser Breite angezeigt. Reicht der horizontale Platz nicht aus, erfolgt ein Umbruch in die nächste Zeile. Alternativ zur c-Option. Wird weder eine w- noch ein c-Option angegeben, erhalten die Cards/Tiles die Breite durch ihren Inhalt.
<b>-cNN</b>	Angabe der Spaltenanzahl, in denen die Cards/Tiles angezeigt werden. Die Breite

	der Cards/Tiles wird hierbei automatisch aus der zur Verfügung stehenden Breite dividiert durch die Spaltenanzahl berechnet. Alternativ zur w-Option. Wird weder eine w- noch ein c-Option angegeben, erhalten die Cards/Tiles die Breite durch ihren Inhalt.
<b>-l</b>	Übernahme der Spaltenüberschrifttexte aus den zugehörigen <th> Zellen als Label vor die Wertangaben. Dieses funktioniert nur korrekt, wenn die Tabelle zu jeder Wertpaltenzelle (<td>) auch eine Überschriftenzelle <th> hat.

### WICHTIG

Der Aufbau der HTML-Tabelle muss nach folgendem Schema erfolgen, da die CSS Klassen die einzelnen Elemente als Selektoren benötigen:

```
<table>
  <thead>
    <tr>
      <th>Überschrift Spalte1</th>
      <th>Überschrift Spalte2</th>
      <th>Überschrift Spalte3</th>
      ... usw ...
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Wert Zeile 1 Spalte1</td>
      <td>Wert Zeile 1 Spalte2</td>
      <td>Wert Zeile 1 Spalte3</td>
    </tr>
    <tr>
      <td>Wert Zeile 2 Spalte1</td>
      <td>Wert Zeile 2 Spalte2</td>
      <td>Wert Zeile 2 Spalte3</td>
    </tr>
    ... usw ...
  </tbody>
</table>
```

## 24.1.Card/Tile Darstellung mit Label

In der Card/Tile-Darstellung wird die Überschriftzeile ausgeblendet und die Datenspalten (Zellen) werden untereinander dargestellt. In einigen Fällen mag es aber sinnvoll sein, diesen Daten dann Labels (Texte) voranstellen zu können um eine bessere Lesbarkeit der Daten zu erhalten.

Hier gibt es zwei Möglichkeiten:

### (a) Übernahme der Überschriftentexte als Labels

Dieses ist, wie unter Optionen beschrieben, einfach durch das Setzen der Option -l zu erreichen: Übernahme der Spaltenüberschrifttexte aus den zugehörigen <th> Zellen als Label vor die Wertangaben. Dieses funktioniert nur korrekt, wenn die Tabelle zu jeder Wertpaltenzelle (<td>) auch eine Überschriftenzelle <th> hat.

### (b) Setzen eigener Labels in die Datenzellen

Alternativ, oder sollen nur einzelne Daten ein Label erhalten, kann dieses direkt in der <td> Anweisung der HTML-Tabelle mit angegeben werden. Dieses muss jedoch in jeder Zeile wiederholt werden.

```

<tbody>
  <tr>
    <td label="Mein Labeltext">Wert Zeile 1 Spalte1</td>
    <td>Wert Zeile 1 Spalte2</td>
    <td>Wert Zeile 1 Spalte3</td>
  <tr>
  <tr>
    <td label="Mein Labeltext">Wert Zeile 2 Spalte1</td>
    <td>Wert Zeile 2 Spalte2</td>
    <td>Wert Zeile 2 Spalte3</td>
  <tr>
  ... usw ...
</tbody>

```

## 24.2. Tabellen responsive gestalten

Soll eine HTML-Tabelle je nach zur Verfügung stehender Breite unterschiedlich dargestellt werden, so kann dieses ebenfalls durch die Angabe von CSS Klassen erreicht werden. Bei jeder `mdui-table-xxxx`-Angabe kann eine Response-Breite mit angegeben werden, bei deren Unterschreitung diese angewendet wird. Über mehrere solcher Angaben kann sogar eine mehrfache Darstellungsänderung erreicht werden.

### BEISPIEL 1

Die Tabellenzeilen sollen

- als Card mit 3 Spalten dargestellt werden, wenn die Breite kleiner als 1024 Pixel ist
- als Card mit 2 Spalten dargestellt werden, wenn die Breite kleiner als 600 Pixel ist

`mdui-table-ascard-r1024-c3 mdui-table-ascard-r600-c2`

### BEISPIEL 2

Die Tabellenzeilen sollen

- als Card mit je 180 Px Breite dargestellt werden, wenn die Breite kleiner als 1024 Pixel ist
- als Tile mit je 180 Px Breite dargestellt werden, wenn die Breite kleiner als 600 Pixel ist
- als List dargestellt werden, wenn die Breite kleiner als 360 Pixel ist

`mdui-table-ascard-r1024-w180 mdui-table-astile-r600-w180 mdui-table-aslist-r360`

### BEISPIEL 3

Die Tabellenzeilen sollen

- generell als Tile mit 240 Px Breite dargestellt werden
- als List mit Labels aus den Überschriften dargestellt werden, wenn die Breite kleiner als 360 Pixel ist

`mdui-table-astile-w240 mdui-table-aslist-r360-l`


## 25.Menüs, Popupmenüs

folgt

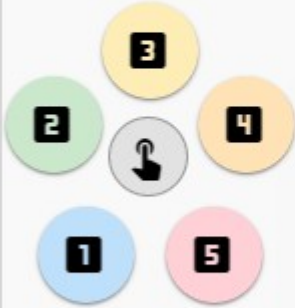
### Menu, Popupmenu

mdui-menu, mdui-menuitem

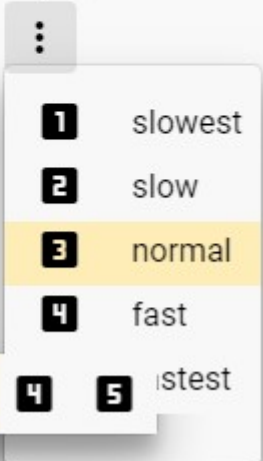
jq - button/container/navigation ...  
mdui-(color): Schriftfarbe  
mdui-(color)-bg: Hintergrundf.




#### Circle-Menu



#### Popup-Menu



#### Line-Menu



## 26.Listen







folgt

### List HTML (show only)

mdui-list, mdui-listitem

basic - html

Diese show-only-List kann dynamisch, ähnlich einer mdui-table, via Javascript erzeugt werden.

	Sonnenaufgang
	14.02.2020 07:48
	Briefkasten
	14.02.2020 08:12
	Küchenfenster geöffnet
	14.02.2020 09:00
	Küchenfenster schließen
	14.02.2020 11:00
	TempSensor Bad:Keine Verbindung
	14.02.2020 11:00
	Internet:Keine Verbindung



## 27.Demo Projekte

Mit auf GitHub ( <https://github.com/Uhula/ioBroker-Material-Design-Style/tree/master/ioBroker%20projects> ) befinden sich Demo-Projekte, welche eine ioBroker.vis Visualisierung ähnlich einer Android-App mit Application-bar, Tab-Navigation und Bottom-Navigation bieten. Die Widgets dieser Demo-Projekte sind nicht mit echten Instanzen verbunden.

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

### **MD\_Demo.zip**

Ein ioBroker Projekt in welchem fast alle Möglichkeiten des Material Design CSS 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.

## 28.mdui Übersicht

Werte	Beschreibung
-------	--------------

### Seitenlayout

mdui-abar	Application bar
mdui-tnav	Top Navigation (Register)
mdui-lnav	Linke Navigation (Hauptmenu, Sidepanel)
mdui-rnav	Rechte Navigation (Funktionen, Sidepanel)
mdui-content	Inhaltsbereich einer Seite

### Farben

mdui-(color)	Schriftfarbe
mdui-(color)-bg	Hintergrundfarbe
mdui-(color)-ol	Rahmenfarbe
mdui-(color)-acc	Akzentfarbe
(color) = red, pink, purple, deeppurple, indigo, blue, lightblue, cyan, teal, green, lightgreen, lime, yellow, amber, orange, deeporange, brown, grey, darkgrey, bluegrey, white, black, color1, color2, color3, white010..white-090, black010..black090	

### Hervorhebung

mdui-(color)-blink	Blinkender Schatten
mdui-(color)-flash	Blitzender Schatten
mdui-(color)-pulse	Pulsierender Schatten
mdui-(color)-glow	Glüheffekt

### Struktur-Widgets

mdui-card	Card-Darstellung
mdui-card-raised	Card+Schatten
mdui-card-outlined	Card+Rahmen

### Anordnung, Größe

mdui-flex	Widget anordnen
mdui-flex-stretch	Widget-height anpassen
mdui-cols-(n)-toc-(m)	Breite in Grid (a 80Px)
mdui-rows-(n)-tor-(m)	Höhe in Grid (a 80Px)

### Text-Widgets

mdui-label	Label Text
mdui-value	Wert
mdui-title	Titel, Überschrift
mdui-subtitle	Untertitel
mdui-state	State-Werte anzeigen

### Eingabe-Widgets

mdui-radio	Radio-Button
mdui-input	Eingabefelder

Werte	Beschreibung
mdui-select	Auswahlliste
mdui-switch	Ein/Aus Schalter
mdui-slider	Slider
mdui-chips	Chips für Radio button
<b>Gruppen (Zoom)</b>	
mdui-group-(name)	Dient der Gruppierung von Widgets (name) ist frei eingebbar
mdui-toggle	Widget togglet (sichbar ja/nein) andere Widgets
mdui-hide	Das Widget wird zur Laufzeit per Vorgabe versteckt
mdui-target-(widgetID)	Angabe einer (Ziel-)Widget-ID, welches verändert werden soll
mdui-scale-(type)	Ziel-Widget skalieren. (type)= fit   hfit   vfit   in   out   (nnn)
mdui-fullscreen	(nnn)= %-Angabe Vollbildzoom (Parent-View)
mdui-timespan-(time)	Zeitbereich FLOT-Diagramm setzen (time)= inc   dec   (nnnnn)
	(nnnnn)= Zeirspanne in [min]
<b>Tabellen</b>	
mdui-table	Tabelle
mdui-table-bordered	Trennlinien zwischen Zeilen
mdui-table-striped	Jede 2.Zeile aufhellen
mdui-table-raised	als Card+Schatten
mdui-table-outlined	als Card+Rahmen
mdui-table-list	Zeilen als List darstellen
Mdui-table-opt-...	(...): -rNNNN    -wNNNN    -cNN    -l
<b>Sonstiges</b>	
mdui-icon	Icon: <i class='mdui-icon'>home</i>
mdui-rotateZ-cw[w]	Um die Z,Y,X Achse im (cw) oder gegen (ccw) Uhrzeigersinn
mdui-rotateY-cw[w]	rotieren lassen
mdui-rotateX-cw[w]	Widget um nnn° drehen
mdui-rotate-(nnn)	Mit flow-Animation
mdui-flow	versehen
mdui-slowest	Geschwindigkeitsangabe, zB für mdui-
mdui-slower	

Werte	Beschreibung
mdui-slow	rotateZYX und mdui-flow
mdui-normal	
mdui-fast	
mdui-faster	
mdui-fastest	
<b>Formen</b>	
mdui-shape-arrow08	
mdui-shape-arrow16	
mdui-shape-trangle	
mdui-shape-corner	
mdui-shape-semicircle	
mdui-shape-circle	
mdui-shape-quadrant	
mdui-h-flip	Horizontal spiegeln
mdui-v-flip	Vertikal spiegeln

## 29.Änderungen

24.09.2017	UH	Kapitel „Select“ hinzugefügt Kapitel „Transparente Slider“ hinzugefügt
02.10.2017	UH	Kapitel „FLOT Diagramme“ hinzugefügt
11.10.2017	UH	Optische Überarbeitung Kapitel „Tabellen“ hinzugefügt
18.10.2017	UH	Übersicht mdui CSS Klassen hinzugefügt
05.01.2018	UH	Ergänzt um mdui-(color)-glow, mdui-state, mdui-cols-X
23.03.2020	UH	Überarbeitung zu v2

## 30.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 JEDLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL SIND DIE AUTOREN ODER COPYRIGHTINHABER FÜR JEDLICHEN 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.