CSS im VIS

Inhaltsverzeichnis

- I - Vorweg:
- II - Grundsätze
- III - Die Styles
- IV - Beispiel: mein VIS
  - IV.I - Hauptkasse
  - IV.II - Größe
  - IV.III - Farbe
  - IV.IV - 4te Klasse
- V - Der Einsatz
- VI - Besonderheiten
  - VI.I - TV-Spielfilm-Widget
  - VI.II - ical-Widget (Kalender)
- VII - Schlussbemerkung
  - VII.I - Weblinks

Hier möchte ich mal kurz an meinem VIS die Verwendung von CSS im VIS erläutern

Da ich öfter mal nach meinem Design gefragt werde, möchte ich hier mal grundsätzlich erklären, wie man im VIS mit CSS die meisten Widgets einfach und einheitlich stylen kann.

Einleitung

I - Vorweg:

Alle Widgets lassen sich direkt und einzeln über die CSS-Tabs in den Einstellungen anpassen

diese Felder machen nichts anderes, als die entsprechende CSS-Option, nur etwas einfacher dargestellt. Auch hier sind komplexere Eingaben möglich. z.B.

Grundsätzlich unterscheidet sich das Styling hier nicht von dem generellen CSS-Styling im "CSS"-Reiter. Es sieht einfach nur ein bischen übersichtlicher aus.

II - Grundsätze

CSS-Styles, die für das gesamte VIS gelten sollen, werden grundsätzlich im "CSS"-Reiter eingegeben,

Wichtig ist, dass hier "Projekt" anstatt "Global" eingestellt wird, ansonsten hätten die Einstellungen auch Auswirkungen auf die Editor-Oberfläche.

um einem Widget irgendeine CSS-Klasse zuzuweisen, muss diese direkt im Eigenschaften-Dialog des Widgets hinterlegt werden.

CSS

III - Die Styles

Im VIS können nahezu alle CSS-Eigenschaften, die es gibt, verwendet werden. Hier und da gibt es bei dem ein oder anderen Widget leider ein paar Einschränkungen oder irgendwelche Eigenschaften die nicht wie gewollt funktionieren. Da hilft dann leider nur probieren.

IV - Beispiel: mein VIS

Wer sich etwas mit CSS beschäftigt, wird schnell feststellen, das sich mehrere CSS-Klassen kombinieren lassen.

Dies ist grundsätzlich sinnvoll. Es erhöht die Übersichtlichkeit und vereinfacht notwenige Überarbeitungen.

Ich style meine Widgets mit 3 Klassen. Damit vermeide ich doppelte und redundante Einträge.

Meine Widgets haben also grundsätzlich 3 Klassen (wie schon in der Einleitung in den Screenshots zu sehen)

Damit fahre ich sehr gut und bin recht flexibel.

IV.I - Hauptkasse

meine "Grundklasse" heißt ".rod" (engl. Stäbchen)

Ihr könnt eure Kasse natürlich auch ".Paul" oder wie auch immer nennen.

Ich nutze diese "Grundklasse" um allen Widgets die Grundeinstellungen mitzugeben, die für alle Widgets gleich sein sollen, wie Textausrichtung und Ränder.

mein CSS sieht dann so aus.

Code

```
.rod{
    display:grid;
    align-items:center;
    justify-items:center;
    text-align:center;
    align-content:center;
    border-radius:4px;
    font-weight:bold;


    line-height:0.9;
    letter-spacing:-1.1px;
}
```

Alles anzeigen

Die ganzen Einträge mit "center" am Ende sind teilweise doppelt, aber da unterschiedliche Widgets teilweise auf andere Eigenschaften reagieren, ist dies leider nötig.

IV.II - Größe

In meinem VIS verwende ich Widgets unterschiedlicher Größen.

Meine Standardhöhe für einzeilige Grafiken ist 35 Pixel. Für z.B. Buttons verwende ich die doppelte Höhe also 70 Pixel.

Deswegen habe ich verschiede Styles für die verschiedenen Höhen. In diesen Styles selbst ist das "height"-Attribut gesetzt, das heißt, das Widget wird 'automatisch' in der richtigen Höhe angezeigt. Eine zusätzliche Höhenangabe in den Eigenschaften des Widgets ist damit nicht mehr notwenig.

Derzeit habe ich für die Größe folgende Styles definiert.

Code

```
.norm{
    height:35px;
    font-size:24px;
}
.big{
    height:70px;
    font-size:28px;
}
.triple{
    height:105px;
    font-size:24px;
}
.head{
    height:70px;
    font-size:40px;
    font-weight:900;
}
.small{
    height:35px;
    font-size:18px;
}
.wide{
    height:35px;
    font-size:24px;
}
.icon{
    height:70px;
    width:70px;
    font-size:18px;
    letter-spacing:-0.09em;
    line-height:13px;
    padding-top:20px;
}
.iconmini{
    height:35px;
    width:35px;
}
.button{
    height:70px;
    width:70px;
    font-size:18px;
}
.half{
    height:17.5px;
    font-size:14px;
}
.textbox{
    font-size:22px;
}
```

Alles anzeigen

Anmerkung:

Die 'Icons', die ich in meiner VIS als rechteckige Buttons nutze, haben nicht nur ein "height"-Attribut sondern auch ein "width"-Attribut, welches die Breite bestimmt.

Die zusätzlichen Angaben im ".icon"-Style sind notwendig, damit die eingefügte Grafik auch vernünftig angezeigt wird. (Ohne diese Angaben wären die Grafiken leider an die obere Kante gequetscht)

IV.III - Farbe

als drittes definiere ich die Farbe der Widgets.

Hier ist garnicht so viel zu zu sagen. Es werden jeweils nur der Hintergrund mit Fabverlauf sowie die Schriftfarbe definiert.

Code

```
.black{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #5b5b5b    0%,#0c0c0c    100%);

}
.yellow{
    color:black;
                background:    linear-gradient(to    bottom,    #fffaa8    0%,#f9d000    100%);

}
.red{
    color:black;
                background:    linear-gradient(to    bottom,    #ff7575    0%,#9b2b29    100%);

}
.lightblue{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #aad9ff    0%,#1b5989    100%);

}
.blue{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #4f9bd6    0%,#002e54    100%);

}
.grey{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #c9c9c9    0%,#424242    100%);
    border-color:#424242;
}
.green{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #aaff91    0%,#12b500    100%);
    border-color:#12b500;
}
.white{
    color:black;
                background:    linear-gradient(to    bottom,    #ffffff    0%,#bcbcbc    100%);
    border-color:#bcbcbc;
}
.darkyellow{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #ddac4f    0%,#7c5b01    100%);

}
.darkblue{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #406dad    0%,#001c38    100%);

}
.darkgreen{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #62e03c    0%,#043f00    100%);
    border-color:#043f00;
}
.darkred{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #cc4747    0%,#561111    100%);

}
.pink{
    color:#ececec;
                background:    linear-gradient(to    bottom,    #f2c8f4    0%,#7f0a7d    100%);

}
.orange{
    color:black;
                background:    linear-gradient(to    bottom,    #f7d7a3    0%,#e06800    100%);
    border-color:#e06800;
}
.glowred{
```

Alles anzeigen

Anmerkung: Die beiden letzten Klassen mit "glow" am Anfang, definieren nur einen leuchtenden Rand um die Widgets und sind keine 'eigentlichen' Farbklassen in meinem Sinn, sonder werden von mir nur zusätzlich zu den gewählten Farbklassen eingesetzt, um einem Widget zusätzlich zur Farbe, einen Leuchteffekt mitzugeben (siehe auch nächstes Kapitel)

IV.IV - 4te Klasse

Einige Widgets habe ich für spezielle Funktionen vorgesehen, deshalb haben diese eine 4te CSS-Klasse wie z.B .blink oder .marquee (oder auch die "glow"-Effelkte im vorherigen Kapitel).

Diese dienen zusätzlich zu den normalen Stylings dazu, das Widget blinken zu lassen oder den Text im Widget als Laufschrift zu gestalten.

Code

```
@keyframes                    marquee                    {
                                          0%
                                          100%          {          text-
}
.marquee                                                 {
    color:white;




                    animation:    marquee    6s    cubic-bezier(0.2,0.7,0.8,0.3)    infi
}
@keyframes                    blink                      {
                                               0%        {
                                               20%       {
                                               80%       {
                                               100%      {
}
.blink                                                   {
                                                   animation:
}
```

Alles anzeigen

Auf die Funktionen will ich nicht extra eingehen, da verweise ich die Weblinks unten.

V - Der Einsatz

Grundsätzlich lässt sich nahezu jede CSS-Eigenschaft benutzen, wichtig zu wissen ist dabei, das nicht jedes Widget mit jeder CSS-Eigenschaft gestylt werden kann.

Hier gilt leider einfach das Motto... Versuch macht kluch...

Weiterhin zu beachten ist, dass die CSS-Klasse im "CSS"-Reiter grundsätzlich mit einem Punkt voran geschrieben werden muss. (Wie es auch die CSS-Definition sagt)

Um dem Widget die entspechenden Klassen zuzuweisen, werden diese ohne Punkte in den Eigenschaften-Dialog geschrieben.

VI - Besonderheiten

Es gibt Widgets, die sich anders Verhalten und spezielle Styling-Optionen benötigen.

Als Beispiel hier kurz erwähnt:

VI.I - TV-Spielfilm-Widget

Da es sich hier um tabellarische Ansichten handelt, lassen sie diese nur durch die Verwendung von CSS-Styles für Tabellen bearbeiten.

Hier mal mein Code dazu:

Code

```
.tv                                                                    {

}
.tv                                        a                           {

    text-decoration:none;
}
.tv-tr:nth-child(odd)                                                  {

}
.tv-tr:nth-child(even)                                                 {

}
```

Alles anzeigen

VI.II - ical-Widget (Kalender)

Das ical-Widget ist ebenfalls eine Besonderheit, hier liegt ebenfalls eine Tabelle vor, welche von Hause aus mit CSS-Tags versehen ist und über spezielle Tags gestylt werden muss.

hier gibt es die Möglichkeit, heutige, morgige, übermorgige und sonstige Termine separat zu stylen.

Im einzelnen verweise ich hier mal auf die Doku vom ical-Adapter.

VII - Schlussbemerkung

Das oben Gezeigte sind natürlich alles nur Beispiele und sollen eine Grundidee erklären.

Jedes Layout, jeder CSS-Klasse und jeder Style können, mit den nötigen CSS-Kenntnissen, komplett individuell angepasst werden.

Mein Layout ist nicht als starre Vorgehensweise zu sehen, sonder eher als meine persönliche "Best-Practice".

Kommentare und Ergänzungen sind gern gesehen.

VII.I - Weblinks

- Alle wichtigen Informationen zu CSS
  https://wiki.selfhtml.org/wiki/CSS
- Tool für Farbverläufe mit CSS
  https://www.colorzilla.com/gradient-editor/