

Zeitberechnung in Adobe Acrobat Nachtrag vom 20. Januar 2018 (ab Seite 3)

Berlin am 31. Dezember 2017,

Kleines Skript für die Berechnung von Uhrzeit-Differenzen und deren Darstellung auch in Dezimalzeit,
aufgeschrieben von Frank Zeitz

Von	Bis	Stunden u. Minuten	Dezimalzeit
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Der Code

```
1 var dtStart = this.getField("von").value;
2 var dtEnde = this.getField("bis").value;
3 dtStart = dtStart.toString();
4 dtStart = dtStart.split(":");
5 var dtStartH = Number(dtStart[0]);
6 var dtStartM = Number(dtStart[1]);

7 dtEnde = dtEnde.toString();
8 var dtEnde = dtEnde.split(":");
9 var dtEndeH = Number(dtEnde[0]);
10 var dtEndeM = Number(dtEnde[1]);

11 var zeitStart = dtStartH * 60 + dtStartM;
12 var zeitEnde = dtEndeH * 60 + dtEndeM;

13 var zeitDiff = zeitEnde - zeitStart;

14 var minuten = zeitDiff % 60;
15 var stunden = (zeitDiff-minuten) / 60;

16 var minutenausdruck = minuten;

17 if (minuten < 10) {
18     minutenausdruck = "0"+minuten;
19 }

20 this.getField("dauer").value = stunden + ":" + minutenausdruck;

21 var dezimalzeit = Number(stunden) + Number(minuten/60);
22 dezimalzeit = dezimalzeit.toFixed(3);

23 this.getField("dezimaldauer").value = dezimalzeit;
```

Dieses Dokument ist interaktiv. Probieren Sie die Funktionalität einfach oben in den vier Feldern aus!

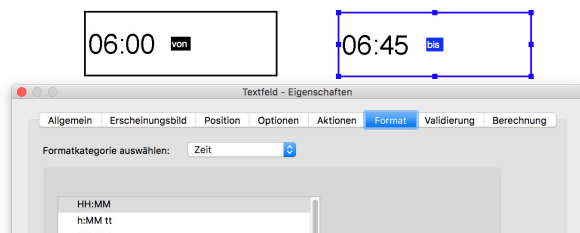
Erläuterung des Codes

Das Rechnen mit Zeit ist in Excel ja sehr einfach, wir hatten da einen kurzen Exkurs im Seminar.

In JavaScript ist es eigentlich auch nicht viel anders, nur evtl. etwas ungewohnt. Ich habe das Skript, das ich dafür geschrieben habe, deshalb sehr ausführlich, und damit hoffentlich auch nachvollziehbar gehalten.

Grundgedanke

Ich habe zwei Felder „von“ und „bis“, die ich so formatiere, daß nur Uhrzeiten eingegeben werden können.



Ich lese diese im Skript in zwei Variablen ein, und konvertiere die Variable dann in einen normalen „String“, also eine Zeichenkette [vgl. Seite 1, Zeilen 1 bis 3 und 7]:

```
var dtStart = this.getField("von").value;
var dtEnde = this.getField("bis").value;
dtStart = dtStart.toString();
```

Dann splitte ich beide Werte jeweils in einen Stunden und einen Minutenwert (die Trennmarke ist der Doppelpunkt der Uhrzeit) [vgl. Seite 1, Zeilen 4 bis 6 und 8 bis 10]:

```
dtStart = dtStart.split(":");
var dtStartH = Number(dtStart[0]);
var dtStartM = Number(dtStart[1]);
var dtEnde = dtEnde.split(":");
var dtEndeH = Number(dtEnde[0]);
var dtEndeM = Number(dtEnde[1]);
```

Diese vier Werte wandle ich dann in Zahlen um [vgl. Seite 1, Zeilen 11 bis 12]:

```
var zeitStart = dtStartH * 60 + dtStartM;
var zeitEnde = dtEndeH * 60 + dtEndeM;
```

So bekomme ich Gesamtminuten, die ich voneinander abziehen kann [vgl. Seite 1, Zeile 13]:

```
var zeitDiff = zeitEnde - zeitStart;
```

Diese rechne ich dann wieder in Stunden um. Indem ich die Zeitdifferenz *zeitDiff* modulo *60* rechne, erhalte ich die Restminuten. Das ist die Zahl, die ich rechts vom Doppelpunkt anzeigen möchte.

Die Restminuten ziehe ich dann von *zeitDiff* ab und teile diesen Wert durch *60*. Dadurch erhalte ich genau die Stunden, die ich links des Doppelpunkts anzeigen möchte [vgl. Seite 1, Zeilen 14 bis 15]:

```
var minuten = zeitDiff % 60;
var stunden = (zeitDiff - minuten) / 60;
```

Das ist alles, das benötigt wird für die Anzeige im dritten Feld, also im Feld „Stunden und Minuten“. Falls ich weniger als 10 Restminuten habe, benötige ich noch eine *0*, die ich diesen voranstelle, damit ich eine schöne Minutenanzeige erhalte. Das ist aber auch schon alles [vgl. Seite 1, Zeilen 16 bis 20]:

```
var minutenausdruck = minuten;
if (minuten < 10) {
    minutenausdruck = "0" + minuten;
}
this.getField("dauer").value = stunden + ":" + minutenausdruck;
```

Da man aber mit diesem Wert nicht rechnen kann, also jedenfalls keine sinnvolle Multiplikation mit Eurobeträgen, habe ich noch ein viertes Feld eingefügt, das den Zeitwert in sogenannte Dezimalzeit umwandelt.

Sowohl die stunden also auch die minuten Variable sind ja Strings. Deshalb müssen diese erst mithilfe des Befehls `Number()` in Zahlen zurückverwandelt werden. Der Wert für Minuten muß dabei natürlich durch 60 geteilt werden. Jetzt entsteht gegebenenfalls eine lange Gleitkommazahl. Ich habe deshalb die Anzahl der Nachkommestellen auf drei reduziert. Und anschließend den Wert in das vierte Feld „Dezimalzeit“ übertragen [vgl. Seite 1, Zeilen 20 bis 23]:

```
var dezimalzeit = Number(stunden) + Number(minuten/60);
dezimalzeit = dezimalzeit.toFixed(3);
this.getField(„dezimaldauer“).value = dezimalzeit;
```

Die Dezimalzeit kann dann für weitere Berechnungen, z.B. für Multiplikation mit einem Stundensatz, verwendet werden.

Sicherlich kann man das Skript sehr viel kürzer fassen, und damit sicher auch etwas effizienter, aber mein Ziel war hier, wirklich jeden Einzelschritt, der gedanklich notwendig ist, in eine eigene Programmierzeile zu packen.

Dieses kleine Skript ist auch überhaupt nicht vollständig. Wenn man zum Beispiel für die „Bis“ Zeit einen niedrigeren Wert eingibt als für die „Von“ Zeit, dann ist das Ergebnis, nun ja, mindestens etwas unbefriedigend.

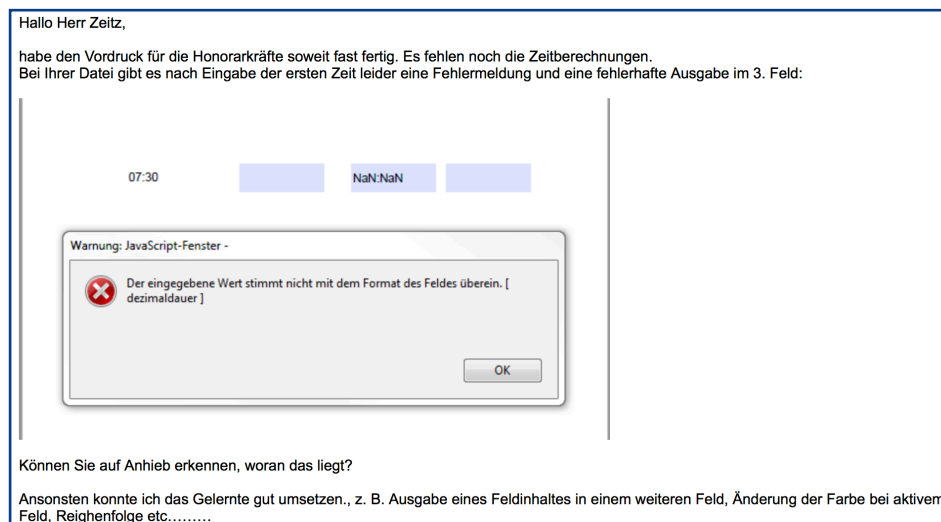
Hier könnte man sicherlich sehr leicht mithilfe einer Wenn-Dann-Bedingung [vgl. die Zeilen 17-19 auf Seite 1] dafür sorgen, daß immer ein positiver Wert für *zeitDiff* berechnet wird.

Ich hoffe, Sie können das gut nachvollziehen.

Falls an einer Stelle nicht: Einfach nachfragen!

Erweiterung des Codes

Am 16. Januar erhielt ich noch einmal diese Nachfrage.



Hallo Herr Zeitz,

habe den Vordruck für die Honorarkräfte soweit fast fertig. Es fehlen noch die Zeitberechnungen.
Bei Ihrer Datei gibt es nach Eingabe der ersten Zeit leider eine Fehlermeldung und eine fehlerhafte Ausgabe im 3. Feld:

07:30 NaN:NaN

Warnung: JavaScript-Fenster -
Der eingegebene Wert stimmt nicht mit dem Format des Feldes überein. [dezimaldauer]

Können Sie auf Anhieb erkennen, woran das liegt?

Ansonsten konnte ich das Gelernte gut umsetzen., z. B. Ausgabe eines Feldinhaltes in einem weiteren Feld, Änderung der Farbe bei aktivem Feld, Reihenfolge etc.....

Deshalb erweitere ich den Code nun an drei Stellen ein wenig. Im dritten Feld steht hier sowohl für den Stunden- als auch für den Minutenwert „NaN“, was für „not a number“ steht. Das ist auch nachvollziehbar, denn das zweite Feld (Endzeit) ist ja leer. Die Zeit von Anfangszeit bis zu leerer Endzeit kann natürlich nicht berechnet werden. Das gleiche Problem ist gegeben, wenn die Anfangszeit leer wäre. Ein weiteres Problem ist, wenn die Endzeit vor der Anfangszeit läge.

Den Code habe ich nun so erweitert, daß es eine Fehlermeldung gibt, wenn die *Anfangszeit* leer ist. Das wird aber nicht nur mit einer Fehlermeldung abgefangen, sondern das dritte Feld (Dauer) und das vierte Feld (Dauer als Dezimalzeit) werden dann auf leer gesetzt. Die Zeile mit der Fehlermeldung selbst (über den Befehl `app.alert`) könnte man hier auch gern herausnehmen, denn es ist ja normal, daß anfangs nicht alle Felder ausgefüllt sind.

Die genau entsprechende Erweiterung habe ich auch vorgenommen, falls die *Endzeit* noch leer ist. Auch hier gibt es eine Fehlermeldung per `app.alert`, die gerne entfernt werden kann.

Wenn die Endzeit früher ist als die Anfangszeit, gibt es mehrere Möglichkeiten, dies zu behandeln. Ich habe mich dafür entschieden, es hier als Fehler zu betrachten, der eine Meldung nach sich zieht. Eine andere Möglichkeit wäre, den negativen Ergebniswert zunächst zuzulassen, dann aber 24 Stunden hinzuzuaddieren. Dies hätte den Effekt, daß die „frühere“ Uhrzeit als Uhrzeit am nächsten Tag bewertet würde. Natürlich könnte man die ganze Sache so erweitern, daß sowohl bei Anfangszeit als auch bei Endzeit auch noch ein jeweils zugehöriges Anfangs- und Enddatum eingetragen werden müßte. Aber ich wollte hier ja nicht eine Lösung anbieten, die alle denkbaren Anforderungen erfüllt, sondern die grundlegend zeigt, das Zeitberechnungen in Acrobat-Formularen möglich sind.

Hier nun der Code:

```
var dtStart = this.getField("von").value;
var dtEnde = this.getField("bis").value;
if (dtStart == "") {
    app.alert("Das Feld Anfangszeit darf nicht leer sein, sondern muß so ausgefüllt sein: (H)H:MM");
    this.getField("dauer").value = "";
    this.getField("dezimaldauer").value = "";
} else {
    if (dtEnde == "") {
        app.alert("Das Feld Endzeit darf nicht leer sein, sondern muß so ausgefüllt sein: (H)H:MM");
        this.getField("dauer").value = "";
        this.getField("dezimaldauer").value = "";
    } else {
        dtStart = dtStart.toString();
        dtStart = dtStart.split(":");
        var dtStartH = Number(dtStart[0]);
        var dtStartM = Number(dtStart[1]);

        dtEnde = dtEnde.toString();
        var dtEnde = dtEnde.split(":");
        var dtEndeH = Number(dtEnde[0]);
        var dtEndeM = Number(dtEnde[1]);

        var zeitStart = dtStartH * 60 + dtStartM;
        var zeitEnde = dtEndeH * 60 + dtEndeM;

        var zeitDiff = zeitEnde - zeitStart;
        if (zeitDiff < 0) {
            app.alert("Die Anfangszeit darf nicht später als die Endzeit sein..");
        }
    }
}
```

```
var minuten = zeitDiff % 60;

var stunden = (zeitDiff-minuten) / 60;

var minutenausdruck = minuten;

if (minuten < 10) {
    minutenausdruck = „0“+minuten;
}

if (zeitDiff < 0) {
    app.alert(„Die Anfangszeit darf nicht später als die Endzeit sein.“);
    this.getField(„dauer“).value = „“;
    this.getField(„dezimaldauer“).value = „“;
} else {
    this.getField(„dauer“).value = stunden + „:“ + minutenausdruck;

    var dezimalzeit = Number(stunden) + Number(minuten/60);
    dezimalzeit = dezimalzeit.toFixed(3);
    //app.alert(dezimalzeit);
    this.getField(„dezimaldauer“).value = dezimalzeit;
}
}
}
```

Den Code können Sie – falls Sie über Adobe Acrobat Pro verfügen – auch ganz einfach aus dem Skript im dritten Feld (Dauer) auf Seite 1 herauskopieren. Sie müssen dafür natürlich diese Datei mit Acrobat öffnen und in den Formularbearbeitungs-Modus gehen.