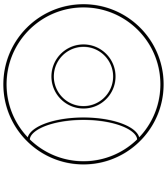
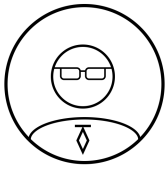
Entwicklung und Dokumentation einer Anwendung zur Darstellung von Zahlen in verschiedenen Zahlensystemen

Informatik

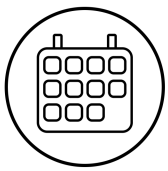


Fabian Kachlock 10b

Sportoberschule Chemnitz
Schuljahr 2019/2020



Herr Schenkel



Abgabetermin: 17.01.2020

Inhaltsverzeichnis

1 Einleitung

Ich war schon länger sehr interessiert an verschiedenen Zahlensystemen und dem Umrechnen zwischen ihnen. Ich hatte schon einmal ein Programm versucht zu schreiben, mit welchem man zwischen dem Dezimal- und Binärsystem umrechnen kann, leider erfolglos. Deshalb hatte ich die Idee, dieses Thema erneut aufzufassen und zu verallgemeinern. Somit habe ich mir als Ziel gesetzt ein Programm zu schreiben, welches von jedem Gerät aus nutzbar ist und in welchem man sich nicht an feste vorgegebene Positionssysteme halten muss.

Um vorher noch einige Dinge zu erklären: Die Entwicklungsgeschichte der Zahlensysteme ist auf Grund von hoher Inhaltlicher Komplexität auf Fakten beschränkt, welche für unser heutiges, weit verbreitetes Dezimalsystem zutreffend sind. Außerdem möchte ich im Voraus einmal die Arten von Zahlensystemen nennen und unterscheiden, da dieses Wissen nützlich ist, um die Entwicklungsgeschichte besser zu verstehen. Die Entwicklungsgeschichte basiert größtenteils auf Additions- und Stellenwert-/Positionssystemen. Es gibt noch eine dritte Art, die Hybridsysteme, diese sind aber sehr selten und komplex und daher in meiner Zusammenfassung nicht inbegriffen.

Bei Additionssystemen sind die Zeichen der Zahlenwerte einfach nur nebeneinander angeordnet. In einem Dezimalen Additionsystem würden also für die Zahl 28, 2mal das Zeichen für die Zehn und 8mal das Zeichen für die Eins nebeneinander stehen. Teilweise wurden auch Zwischenstufen, welche nicht einer Potenz der Basis entsprachen eingebaut (zum Beispiel: Bei einem dezimalen System ein Zeichen für die 5).

In Positions- oder Stellenwertssystemen hängt der Wert eines Zeichen oder Symbols hingegen von der Stelle in der Zeichenkette ab. Der Zahlwert ergibt sich aus der Summe von allen Zeichen, multipliziert mit der Potenz der Basis des Zahlensystems an entsprechender Position (zum Beispiel: Dezimalsystem: 28 = (Wert vom Symbol 2) * 10^1 + (Wert vom Symbol 8) * 10^0). Dies kann man auch in einer Summe ausdrücken:

$$\sum_{i=0}^{n} a_i \cdot g^i = a_0 + a_1 \cdot g + a_2 \cdot g^2 + \ldots + a_n \cdot g^n$$

Hierbei steht g für die Basis des Stellenwertsystems und a für den Wert des Symbols an dieser Stelle.

2 Entwicklungsgeschichte der Zahlensysteme

Bis es zu unserem heutigem, leicht verständlichem Dezimalsystem gekommen ist, ist sehr viel Zeit vergangen. Es sind viele verschiedene Ideen aufgekommen, manche blieben erhalten, manche wurden verworfen. Die Geschichte der Zahlensysteme beginnt bereits im alten Ägypten mit der Erfindung der Schrift. Das erste Zahlensystem war bereits ein dezimales System, aber noch ein Additionssystem. Die bestimmten Hieroglyphen wurden also einfach aneinander gereiht (siehe Abb. 1.1). Eine feste Schreibrichtung gab es auch noch nicht, es war egal ob von rechts nach links oder von links nach rechts geschrieben wurde.

Diese hieratische Schrift entwickelte sich zur demotischen Schrift (siehe Abb. 1.2). Die wesentliche Unterschiede waren, dass es eine feste Schreibrichtung gab (von rechts nach links).

Nun wurden auch Zeichen für jeden Wert eingeführt (ein Zeichen für den Wert 1, 2, … bis 9, 10 , 20, … bis 90, 100, 200, … bis 900 und 1000, 2000, … bis 9000), dadurch ließen sich Zahlen wesentlich schneller schreiben und lesen.
Die nächste Station in der Entwicklung ist Babylon, hier gab es eine Keilschrift bestehend aus 2 Zeichen (siehe Abb. 1.3). Es war wieder ein additives System nur diesmal mit der Basis 60. Das Sexagesimalsystem ist uns heute noch bei der Zeit und Einteilung der Winkel erhalten.
Die Griechen nutzten für Ganze Zahlen ihr attisches System (siehe Abb. 1.4). Es war wieder ein dezimales Additionssystem mit den Anfangsbuchstaben der Zahlwörter als Symbole. Erstmalig wurden Zwischenstufen eingeführt (5/50/500/…). Bei dem später eingeführten hellenischen System, nutzte man Buchstaben aus dem Alphabet, da diese aber nicht reichten, wurden noch 3 Episemen hinzugefügt (siehe Abb. 1.5). Wenn es sich um Zehntausender oder größer handelte hat man noch Vorzeichen hinzugefügt. Außerdem wurden die Zahlen mit einem Strich oben drüber gekennzeichnet, da man sie sonst von Wörtern nicht hätte unterscheiden können.
Die Römer hatten ein den Griechen sehr ähnliches System, allerdings nutzten sie Individualzeichen. Später ersetzten sie das additive System durch ein subtraktives (IIII konnte als IV geschrieben werden).
In China gab es viele verschieden Zahlensystem mit teilweise sehr experimentellen Zeichen und Schriftrichtungen. Gegenüber unserer heutigen Zahlwortbildung hatten sie klare Bildungsgesetze. Bei ihnen hieß Dreißig oder Zwölf übersetzt Dreizehn oder Zehnzwei.
Wir in Europa hingegen entwickelten nie ein eigenes System. Zuerst nutzten wir das der Römer, später wandelten wir es in ein Stellenwertssystem um und nutzten als Symbole die Indisch-Arabischen Ziffern (siehe Abb. 1.6).

3 Dokumentation

3.1 Ziel

Das grobe Ziel meiner Anwendung ist simpel. Ich wollte eine Anwendung programmieren, die für jeden zugänglich ist und natürlich zwischen Zahlensystemen konvergieren kann. Für mich selber habe ich dann noch ein paar „Zusatz-Ziele“ hinzugefügt. Einerseits wollte ich so viele Zahlensysteme wie möglich unterstützen. Mein 2. Ziel war den Algorithmus zur Berechnung zu verallgemeinern, damit man den Quelltext möglich kurz halten kann. Über die Zeit der Entwicklung der Anwendung kamen noch andere, vorher nicht geplante Dinge hinzu, wie zum Beispiel eine leichte Steuerung. Nun kann der Algorithmus zwischen allen Zahlensystemen umrechnen, die wie unser Dezimalsystem aufgebaut sind und als Symbole die Ziffern und Buchstaben unseres Alphabets (mit aufsteigendem Wert, wie bei dem Hexadezimalsystem) nutzen.

3.2 Programmierumgebung

Da ich meine Anwendung für alle nutzbar machen wollte, fand ich es die beste Idee den Algorithmus in Form einer Website umzusetzen. Somit nutzte ich HTML (Hypertext Markup Language) für den Grundkörper meiner Website. Mit HTML habe ich das „Gerüst“ gebaut, und

habe die Textstellen, Input-Felder und Schaltflächen hinzugefügt. Das Wichtigste ist, dass ich allen Elementen eine Klasse zuordne, damit ich später, z.B.: im Algorithmus, auf sie zugreifen kann. HTML ist für eine Website bindend, d.h. man braucht für jede Website eine HTML-Datei.
Das Aussehen und die Anordnungen habe ich in CSS (Cascading Style Sheet) festgelegt. CSS ist der 2. der 3 Grundkomponenten einer Website, hier gibt es aber Möglichkeiten mit Präprozessoren (wie Sass oder LESS) bestimmte Funktionen hinzuzufügen, was die Programmierung erleichtern kann. Ich bin aber bei dem ganz normalen CSS-Code geblieben.
Die 3. Grundkomponente einer Website ist Javascript. Diese ist für die Logik hinter der Website zuständig. Da ich keine Server oder Datenbanksysteme brauchte, konnte ich es bei den Grundkomponenten belassen.
Als Quelltext-Editor habe ich Atom (1.40.1 für macOS) benutzt. Für das Arbeiten mit HTML/ CSS/JS nutze ich ihn gerne. In Atom kann man viele verschiedene Packages installieren und nutzen, welche die Arbeit erleichtern. Dazu habe ich die vorinstallierten autocomplete-Packages genutzt und ein selbst installiertes live-server Package, welches mir erlaubte live im Browser die Website zu sehen. Außerdem habe ich den Google Chrom Webentwickler Bereich zum debuggen vom Quellcode verwendet.

3.3 Funktionsweise

Die Funktionsweise meiner Anwendung lässt sich ganz einfach in 3 Schritte, wie bei dem EVA-Prinzip aufteilen: Am Anfang die Eingabe der Daten, welche größtenteils in HTML abläuft. Danach die Berechnung der Zahl, welche in einer Javascript Funktion passiert und am Ende die Ausgabe der errechneten Zahl, der einfachste und kürzeste Abschnitt.

3.3.1 Eingabe Verarbeitung

Eine „Zahl“ besteht sozusagen aus 2 Grundkomponenten. Einerseits die Ziffern- oder Symbolfolge, dass ist der Teil, den man meistens nur sieht. Was man aber nicht vergessen darf ist, dass in einem Stellenwertssystem die Basis ausschlaggebend ist, welchen Wert die Ziffern- oder Symbolfolge hat. Somit ist der zweite essentielle Teil der Zahl die Basis des dazugehörigen Zahlensystems.
Da ich eine Zahl in ein anderes Zahlensystems umrechnen möchte, benötige ich noch eine Info zu dieser Zahl, da ich sonst eine Gleichung mit 2 Unbekannten hätte. In diesem Fall ist die „Info“ die Basis des Stellenwertsystems, in das die Zahl umgerechnet werden soll. Also habe ich 3 Eingabefelder, welche ich mit dem HTML Input-Tag initialisiert habe. Alle 3 sind, da die Zahlensystem nicht unbedingt nur aus Ziffern bestehen, vom Typ Text (siehe Abb. 2).

3.3.2 Berechnung

Nun zur Berechnung, dem Herzstück der ganzen Anwendung. Der erste Schritt in der Berechnung ist die Überprüfung der eingegebenen Werte. Dazu komme ich aber noch im Abschnitt 3.5. Danach wird der Wert der Zahlenkette ausgerechnet, den Wert speichere ich mir als Dezimalzahl, weil man mit ihnen am besten rechnen kann.

Nun gehe ich davon aus, dass die Zahl wenn sie nicht dezimal ist, genauso wie das Binärsystem aufgebaut ist. Also das erste Symbol (von links) hat den höchsten Exponent und das letzte den Exponent 0. Da das Hexadezimal- und das Oktalsystem auch so funktionieren übernehme ich das für alle anderen Zahlensysteme.
Wenn die eingegebene Zahl dezimal ist, muss sie nicht erst umgerechnet werden. Alle anderen Zahlen gehe ich Symbol für Symbol durch, errechne den Wert der Stelle und füge ihn zum Gesamtwert hinzu.
Hierbei mache ich mir zunutze, dass die erste Stelle in einem Array den Index 0 hat, genau wie der Wert der kleinsten Ziffer: 0. Somit hat dann die Ziffer 1 auch den Index, bzw. Wert von 1. Das erleichtert einiges, da ich somit alle unterstützten Zeichen an der richtigen Stelle im Array speichern kann und dann sofort den Wert des Zeichens dazu habe.
Da die Zahlen bereits in Form einer Zeichenkette eingegeben werden, kann ich sie gleich Stelle für Stelle einmal durchgehen. Die Laufvariable in dem Loop nenne ich *i* und lasse sie solange um 1 wachsen, bis sie größer als die länge der Zeichenkette ist. Jetzt gibt es aber das Problem, dass die Stelle, welche den Exponenten 0 hat, ganz am Ende der Zeichenkette ist. Dafür gibt es aber einen kleinen Trick. Ich nehme das Zeichen aus der Kette, welches an Position *Länge der Kette* - 1 (das ist die letzte Stelle der Kette) - *i* ist. Jetzt habe ich für i = 0 die letzte Stelle der Kette, für i = 1 die vorletzte und so weiter. Jetzt muss nur noch der Wert ausgerechnet werden. Der Wert der Stelle ergibt sich nun aus dem Wert des Zeichens (Index in der Liste aller Zeichen) multipliziert mit der Basis ^ *i*.
Jetzt wird der Input in eine Zahl umgewandelt, mit der man ordentlich arbeiten kann. Soll die Zahl vom Output nun Dezimal sein, kann der Wert der Input Zahl (der ja Dezimal war) ausgegeben werden.
Danach muss der Wert in eine Zahl umgerechnet werden. Dazu muss man erst den größtmöglichen Exponenten der Basis des Outputs mit einem Potenzwert, welcher kleiner als der Wert der Zahl ist, herausfinden. Man kann dies mithilfe eines Logarithmus herausfinden: $\log_{\text{Basis vom Output}}$Wert. Da dies meistens eine gebrochene Zahl ist muss man noch abrunden, damit man auch wirklich unterhalb vom Wert der Zahl bleibt. Es ist leider nicht möglich in Javascript einen Logarithmus einer Basis x auszurechnen. Es geht nur mit der Basis 10, deshalb muss man den Logarithmus noch etwas umstellen: $\log_{10}$Wert der Zahl / $\log_{10}$Basis vom Output.
Nun kann die Zahl von links (größter Exponent) nach rechts (Exponent 0) aufgebaut werden. Als Erstes errechne ich den Wert der Potenz aus Basis des Zahlensystems und dem von Stelle zu Stelle kleiner werdenden Exponenten. Jetzt kann ich den Wert der Zahl durch meine eben errechnete Potenz teilen und abrunden. Dann habe ich den Wert des Symbols, welches an dieser Stelle stehen muss. Dieses Symbol wird nun hinten an die anfangs leere Zeichenkette, welche das Ergebnis darstellt, angehängt. Am Ende wird nun noch der Wert, der eben errechneten Stelle, von dem Gesamtwert der Zahl abgezogen. Für die nächste Stelle nutzte ich dann den neuen Exponenten und den nun kleineren Gesamtwert. Falls der Gesamtwert schon vor der letzten Stelle gleich 0 ist, wird dies am Anfang erkannt und die Rechnungen werden

übersprungen und eine 0 wird zur Zeichenkette hinzugefügt. Am Ende wird die fertige Zeichenkette ausgegeben.

3.3.3 Ausgabe

Die Ausgabe funktioniert ganz einfach: Es werden die Inhalte der Inputfelder als Parameter für die gerade beschriebene Funktion genommen und der Text der HTML-Ergebnis-Klasse angepasst.

3.4 Vorstellung ausgewählter Programmsequenzen

Diese Javascript Funktion kontrolliert bereits beim Eingeben des Textes in die Inputfelder, ob nur erlaubte Zeichen verwendet werden:

```
function checkInput(evt) {
  evt = (evt) ? evt : window.event
  var charCode = (evt.which) ? evt.which : evt.keyCode
  if (!((charCode >= 48 && charCode <= 57) || (charCode >= 65
 && charCode <= 90) || (charCode >= 97 && charCode <= 122) ||
 (charCode == 13) || (charCode == 8) || (charCode >= 37 &&
 charCode <= 40))) {
    alert('Sonderzeichen sind nicht erlaubt! Nur [A-Z] [0-9] ')
    return false
  }
}
```

Hier wird überprüft ob die Zeichen, welche in der Zahl verwendet werden, auch zu den erlaubten Zeichen der eingegebenen Basis gehören:

```
for (let i = 0; i<input.length; i++) {
  if (!inputCharacter.includes(input[i])) {
    alert("Invalider Input! Die Zahl passt nicht zur Basis.")
    document.getElementById("UserInputNumber").focus()
    return "Undefiniert"
  }
}
```

So wird der Wert einer Zahl ausgerechnet:

```
if (inputBase == 10) {
    dezimalSum = input
} else {
    for (let i = 0; i<input.length; i++) {
        let wert = allChracter.indexOf(input[input.length-1-i])
        dezimalSum += wert* inputBase**i
    }
}
```

Das Umrechnen ist ähnlich aufgebaut, nur etwas komplexer:

```
01 let exp = Math.floor(Math.log(dezimalSum)/
   Math.log(outputBase))
02 for (let i = 0; i<=exp; i++)  {
03   if (dezimalSum == 0) {
04     output += "0"
05     continue
06   }
07   let multiplicator = outputBase**(exp-i)
08   let factor = Math.floor(dezimalSum/multiplicator)
09   let value =  factor*multiplicator
10  dezimalSum -= value
11  output += outputCharacter[factor]
12 }
```

3.5 Fehlerbehandlung

Da die Berechnung der Zahl sehr komplex und sensibel ist, habe ich viele Vorkehrungen getroffen um falsche Eingaben zu vermeiden. Da die Symbole vorgegeben (Ziffern+Großbuchstaben) sind, muss ich darauf achten, dass auch nur Ziffern und Großbuchstaben verwendet werden. Dazu habe ich einerseits einen Javascript Befehl eingebaut, welcher dafür sorgt, dass alle Buchstaben Großbuchstaben sind. Damit der Benutzer das auch sieht, habe ich bei dem Textfeld eingestellt, dass alle Buchstaben als Großbuchstaben angezeigt werden.

Um bereits bei der Eingabe Sonderzeichen abzufangen, habe ich eine Javascript Funktion, welche die eingegebenen Zeichen, anhand des Codes, der betätigten Taste ausfiltert.

Außerdem wird am Anfang der Berechnung eine Teilliste erstellt. Diese enthält alle Zeichen, die anhand der Basis erlaubt sind. Somit ist es möglich die Eingabe-Zeichenkette zu kontrollieren.

Um die Basen zu kontrollieren, wird in und vor der Berechnung kontrolliert, dass sie nicht zu klein oder zu groß sind. Um die Fehler bei der Eingabe zu minimieren sind die Textfelder beider auf eine Länge von maximal 2 Zeichen begrenzt. Auch das Textfeld der Eingabe Zahl ist von der Länge her auf 50 Zeichen begrenzt. Bei sehr langen und großen Zahlen kann es zu Performance Problemen, bis hin zum Absturz kommen.

Zur einfacheren Bedienung der Anwendung habe ich hinzugefügt, dass man mit Enter zum nächsten Textfeld kommt und am Ende auch umrechnet. Gibt es nun an einer Stelle im Programm einen Eingabefehler ist genau das Textfeld mit dem Eingabefehler wieder im Fokus.

3.6 Zusammenfassung

Ich finde, dass die Anwendung meinen Vorstellungen entsprechend sehr gut gelungen ist. Um verschiedene Fehler zu finden und zu korrigieren, habe ich viel ausgetestet. Soweit ich es getestet habe funktioniert alles einwandfrei. Anfangs habe ich sogar die Zahl selber nochmal umgerechnet um sicher zu gehen, dass es wirklich funktioniert. Um ein letztendliches Fazit zu ziehen: Die Anwendung funktioniert fehlerfrei und hat alle, in den Zielen formulierten Funktionen.

4 Anhang

4.1 Quellenverzeichnis

Text:

19.10.2019 http://www.cevis.uni-bremen.de/Binaries/Binary1105/_Skript.pdf

20.10.2019 und davor: https://www.w3schools.com/

Bilder:

Abbildung 1.1, 1.2, 1.4, 1.5, 1.6 http://www.cevis.uni-bremen.de/Binaries/Binary1105/_Skript.pdf

Abbildung 1.3: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d6/Babylonian_numerals.svg/806px-Babylonian_numerals.svg.png

22.10.2019

Um die Formel in der Einleitung hinzuzufügen:

https://www.grund-wissen.de/informatik/latex/mathematischer-formelsatz.html

29.10.2019

4.2 Quelltext und Dokumente

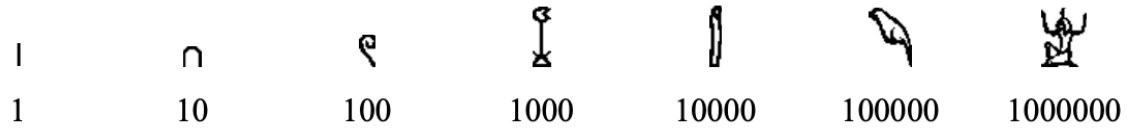


Abbildung 1.1: Die erste Hieroglyphen Schrift der Ägypter

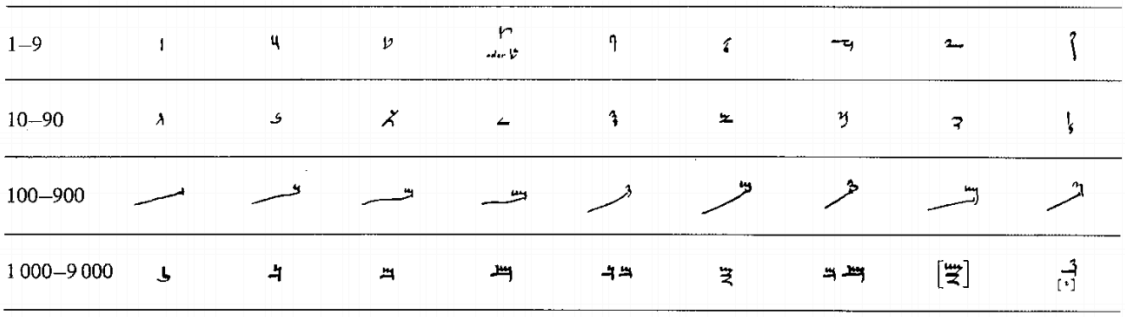


Abbildung 1.2: Die angepassten demotischen Zahlzeichen.

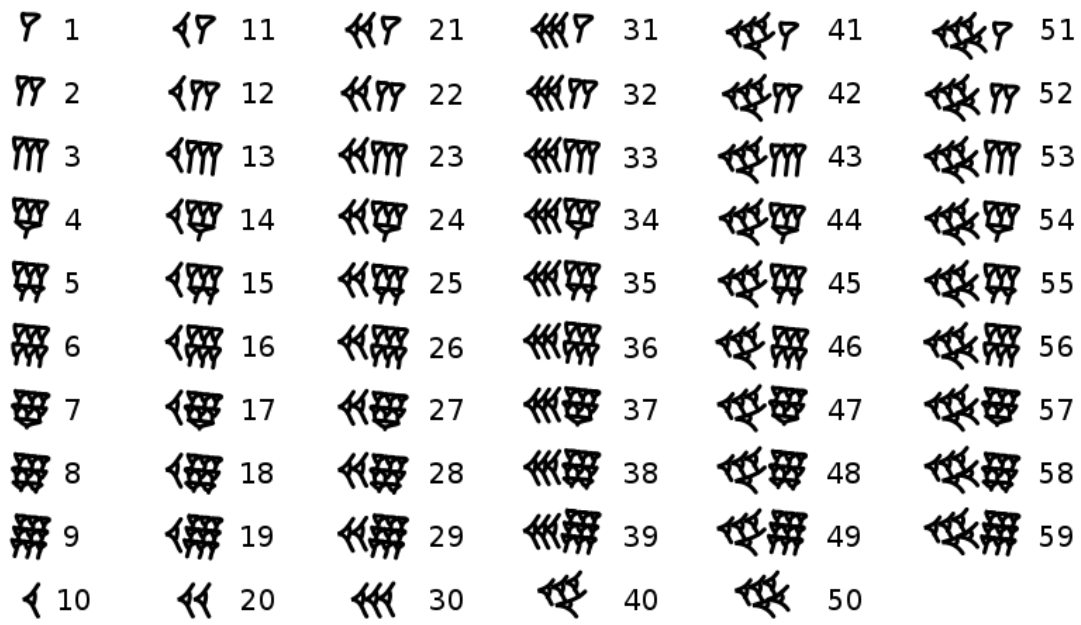


Abbildung 1.3: Die Zahlzeichen der Babylonischen Keilschrift.

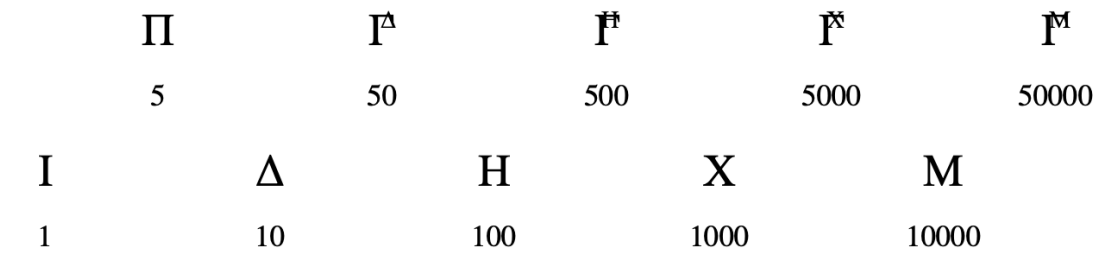


Abbildung 1.4: Die herodianischen Zeichen des attischen Systems.

α	β	γ	δ	ε	ς	ζ	η	θ
1	2	3	4	5	6	7	8	9
ι	κ	λ	μ	ν	ξ	ο	π	ϟ
10	20	30	40	50	60	70	80	90
ρ	σ	τ	υ	φ	χ	ψ	ω	ϡ
100	200	300	400	500	600	700	800	900

Abbildung 1.5: das hellenische System

Abbildung 1.6: Die Entwicklung unserer heutigen indisch-arabischen Ziffern.

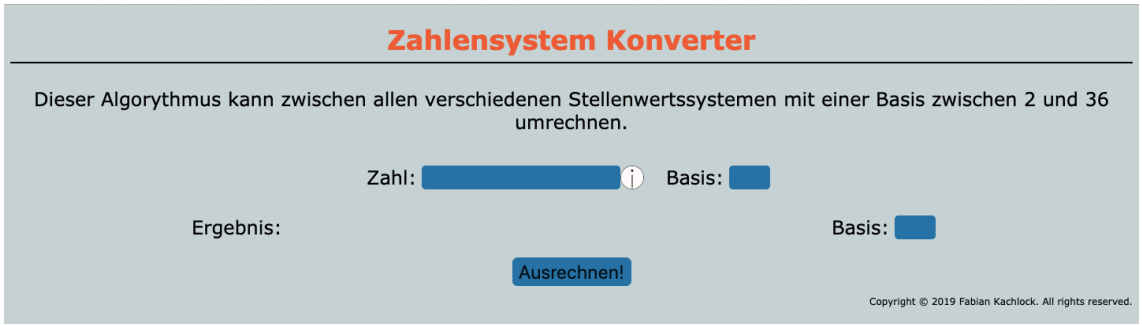


Abbildung 2: Interface der Anwendung

Quelltext:

index.html:

```
<!DOCTYPE html>
<html lang: "de">
  <head>
    <meta charset= "utf-8">
    <meta name="content-language" content="de">
    <link rel="stylesheet" href="main.css">
    <script src="script.js"></script>
  </head>
  <body id="body">
    <div>
      <h1 class="header">Zahlensystem Konverter</h1>
      <hr size="2" color=#000>
      <p class="Anleitung">Dieser Algorithmus kann zwischen allen
   verschiedenen Stellenwertssystemen mit einer Basis zwischen 2
   und 36 umrechnen.</p> <br>
    </div>
    <div class="Input" >
      <div class="InputZahl">
        <p class="UserInputNumberLabel">Zahl:</p>
        <div>
          <input class="UserInputNumber" type="text"
   name="UserInputNumber" id="UserInputNumber" onkeypress="return
   checkInput(event)" maxlength="50" onkeyup="keyUp(event, 1)">
          <button type="button" class="InfoButton"
   onclick="showInfo()"></button>
        </div>
      </div>
      <div class="InputBasis">
        <p class="UserInputBasisLabel" >Basis:</p>
        <input class="UserInputBasis" type="text"
   name="UserInputBasis" id="UserInputBasis" onkeypress="return
   checkInputInBasis(event)" maxlength="2" onkeyup="keyUp(event,
   2)">
      </div>
    </div>
    <div class="Output">
      <div id=„Output">
        <div class="Ergebnis">
          <p id="Ergebnis" class="Ergebnis">Ergebnis:</p>
        </div>
        <div class="InputConvert">
          <p class="UserInputConvertToLabel">Basis:</p>
          <input class="UserInputConvertTo" type="text"
   name="ConvertTo" id="ConvertTo" onkeypress="return
   checkInputOutBasis(event)" maxlength="2" onkeyup="keyUp(event,
   3)" >
        </div>
      </div>
```

```
    </div>
    <div class="submit">
      <button class="submit" type="button"
   onclick="calculate()">Ausrechnen!</button>
    </div>
    <!-- Copyright -->
    <p class="Copyright">Copyright © 2019 Fabian Kachlock. All
   rights reserved.</p>
  </body>
</html>
```

style.css:

```
body {
  background-color: rgba(200, 211, 213, 1);
  font-family: Verdana, Arial, sans-serif;
  word-wrap: none;
  min-width: 500px;
  min-height: 200px;
}
.header {
  color: rgba(247, 92, 3, 1);
  position: relative;
  text-align: center;
  margin-bottom: 5px;
}
.Anleitung {
  text-align: center;
  font-size: 24px;
  margin-top: 30px;
  margin-bottom: 10px;
}
.UserInputNumber{
  background-color: rgba(34, 116, 165, 1);
  font-size: 24px;
  text-transform: uppercase;
  border-style: none;
  border-radius: 5px;
  float: left;
  width: 250px;
  margin-top: 10px;
}
.UserInputNumberLabel{
  font-size: 24px;
  border-style: none;
  float: left;
  width: 70px;
  margin-top: 10px;
}
.UserInputNumber:focus {
  background-color: rgba(34, 116, 165, 1);
  border: 2px solid rgba(247, 92, 3, 1);
}
.InfoButton {
  width: 30px;
  height: 30px;
```

```
  margin-top: 10px;
  border-radius: 15px;
  border: none;
  background-image: url("Info.png");
  background-size: 30px 30px;
  background-repeat: no-repeat;
  background-position: 0px 0px;
  background-color: rgba(200, 211, 213, 1);
  margin-right: 20px;
}
.InfoButton:hover {
  border: 2px solid rgba(34, 116, 165, 1);
  background-position: 0px 0px;
  background-size: 26px 26px;
}
.InputZahl {
  width: 380px;
  float: left;
}
.UserInputBasis{
  background-color: rgba(34, 116, 165, 1);
  font-size: 24px;
  text-transform: uppercase;
  border-style: none;
  border-radius: 5px;
  float: left;
  width: 50px;
  margin-top: 10px;
  text-align: center;
}
.UserInputBasisLabel{
  font-size: 24px;
  border-style: none;
  float: left;
  width: 80px;
  margin-top: 10px;
}
.UserInputBasis:focus {
  background-color: rgba(34, 116, 165, 1);
  border: 2px solid rgba(247, 92, 3, 1);
}
.InputBasis {
  width: 140px;
  float: left;
}
.Input {
  width: 520px;
  overflow: auto;
  left: 50%;
  margin-left: auto;
  margin-right: auto;
}
.UserInputConvertTo {
  background-color: rgba(34, 116, 165, 1);
  font-size: 24px;
```

```
  text-transform: uppercase;
  border-style: none;
  border-radius: 5px;
  float: left;
  width: 50px;
  margin-top: 10px;
  text-align: center;
  white-space: nowrap;
  margin-right: 20px;
}
.UserInputConvertTo:focus {
  background-color: rgba(34, 116, 165, 1);
  border: 2px solid rgba(247, 92, 3, 1);
}
.UserInputConvertToLabel {
  font-size: 24px;
  border-style: none;
  float: left;
  width: 80px;
  margin-top: 10px;
  white-space: nowrap;
}
.Convert {
  width: 140px;
  float: left;
  white-space: nowrap;
  overflow: hidden;
}
.Ergebnis {
  margin-top: 5px;
  font-size: 24px;
  text-align: left;
  width: 55vw;
  margin-right: 20px;
  float: left;
  word-wrap: break-word;
}
.Output {
  display: flex;
  overflow: auto;
  margin-left: auto;
  margin-right: auto;
  justify-content: center;
}
#Output {
  display: inherit;
}
.submit {
  width: 150px;
  height: 35px;
  background-color: rgba(34, 116, 165, 1);
  border: 0.5px solid rgba(34, 116, 165, 1);
  border-radius: 7px;
  font-size: 24px;
  text-align: center;
```

```
  margin-right: auto;
  margin-left: auto;
  cursor: pointer;
}
.submit:hover {
  background-color: rgba(22, 96, 136, 1);
  border-color: rgba(56, 119, 128, 1);
  border-radius: 18px;
}
.Copyright {
  font-size: 8px;
  width: 100%;
  text-align: right;
  margin-bottom: 100%;
}
```

script.js:

```
// Muster für alle Umwandlungen
const allChracter =
   ["0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F"
   ,"G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V"
   ,"W","X","Y","Z"];
// Ausgabe der benötigten Character -> Errorhandling
function getCharacter(base) {
    return allChracter.slice(0,base)
}
function convert(input, inputBase, outputBase) {
    // Errorhandling
    if (inputBase > 36 || inputBase < 2) {
        alert("Die Basis vom Input wird nicht unterstützt!")
        document.getElementById("UserInputBasis").focus()
        return "Undefiniert"
    }
    // Errorhandling
    if (outputBase > 36 || outputBase < 2) {
        alert("Die Basis vom Output wird nicht unterstüzt!")
        document.getElementById("ConvertTo").focus()
        return "Undefiniert"
    }
    const inputCharacter = getCharacter(inputBase)
    const outputCharacter = getCharacter(outputBase)
    let output = ""
    let dezimalSum = 0
    // Errorhandling
    for (let i = 0; i<input.length; i++) {
        if (!inputCharacter.includes(input[i])) {
            alert("Invalider Input! Die Zahl passt nicht zur
  Basis.")
            document.getElementById("UserInputNumber").focus()
            return "Undefiniert"
        }
    }
    // Handle Input
    if (inputBase == 10) {
        dezimalSum = input
```

```
    } else {
        for (let i = 0; i<input.length; i++) {
            let wert = allChracter.indexOf(input[input.length-1-
  i])
            dezimalSum += wert* inputBase**i
        }
    }
    // convert output
    let exp = Math.floor(Math.log(dezimalSum)/
  Math.log(outputBase))
    for (let i = 0; i<=exp; i++)  {
        if (dezimalSum == 0) {
            output += "0"
            continue
        }
        let multiplicator = outputBase**(exp-i)
        let factor = Math.floor(dezimalSum/multiplicator)
        let value =  factor*multiplicator
        dezimalSum -= value
        output += outputCharacter[factor]
    }
    return output
}
function checkInputInBasis(evt) {
  evt = (evt) ? evt : window.event
  var charCode = (evt.which) ? evt.which : evt.keyCode
  if (!((charCode >= 48 && charCode <= 57) || (charCode >= 65 &&
   charCode <= 90) || (charCode >= 97 && charCode <= 122) ||
   (charCode == 13) || (charCode == 8) || (charCode >= 37 &&
   charCode <= 40))){
    alert('Sonderzeichen sind nicht erlaubt! Nur [A-Z] [0-9] ')
    return false
  }
}
function checkInputOutBasis(evt) {
  evt = (evt) ? evt : window.event
  var charCode = (evt.which) ? evt.which : evt.keyCode
  if (!((charCode >= 48 && charCode <= 57) || (charCode >= 65 &&
   charCode <= 90) || (charCode >= 97 && charCode <= 122) ||
   (charCode == 13) || (charCode == 8) || (charCode >= 37 &&
   charCode <= 40))) {
    alert('Sonderzeichen sind nicht erlaubt! Nur [A-Z] [0-9] ')
    return false;
  }
}
function checkInput(evt) {
  evt = (evt) ? evt : window.event
  var charCode = (evt.which) ? evt.which : evt.keyCode
  if (!((charCode >= 48 && charCode <= 57) || (charCode >= 65 &&
   charCode <= 90) || (charCode >= 97 && charCode <= 122) ||
   (charCode == 13) || (charCode == 8) || (charCode >= 37 &&
   charCode <= 40))) {
    alert('Sonderzeichen sind nicht erlaubt! Nur [A-Z] [0-9] ')
    return false
  }
```

```
}
function calculate() {
  let inBasis = document.getElementById("UserInputBasis").value
  let inString =
  document.getElementById("UserInputNumber").value.toUpperCase()
  let outBasis = document.getElementById("ConvertTo").value
  if (inBasis > 36 || inBasis < 2) {
      alert("Die Basis vom Input wird nicht unterstützt!")
      document.getElementById("UserInputBasis").focus()
      document.getElementById("Ergebnis").innerHTML = "Ergebnis:
  Undefiniert"
      return
  }
  if (outBasis > 36 || outBasis < 2) {
      alert("Die Basis vom Output wird nicht unterstüzt!")
      document.getElementById("ConvertTo").focus()
      document.getElementById("Ergebnis").innerHTML = "Ergebnis:
  Undefiniert"
      return
  }
  document.getElementById("Ergebnis").innerHTML = "Ergebnis: " +
  convert(inString, inBasis, outBasis)
}
function keyUp(evt, lf) {
  if (evt.keyCode ==  13) {
    if (lf==1) {
      document.getElementById("UserInputBasis").focus()
    } else if (lf==2) {
      document.getElementById("ConvertTo").focus()
    }else if (lf==3) {
      calculate()
      if (document.getElementById("Ergebnis").innerHTML !=
  "Ergebnis: Undefiniert") {
        document.getElementById("UserInputNumber").focus()
      }
    }
  }
}
function showInfo() {
  alert("Der Algorithmus kann Zahlen von jeden beliebigen
  Stellenwertsystems, welches eine Basis zwischen 2 und 36 hat
  konvertieren. Um diese Zahlen anzugeben werden alle Ziffern
  [0-9] und alle Buchstaben [A-Z] benötigt. Es ist zu beachten,
  dass alle Zahlensysteme (außer das Dezimalsystem) genau
  andersrum aufgebaut sind. Im Gegensatz zum Dezimalsystem hat die
  erste Ziffer von links den höchsten Exponenten.")
}
```

4.3 Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die angeführten Quellen und Hilfsmittel verwendet habe.

Sonntag, 3. November 2019