Cloud-basierte Web-Anwendungen

Inhalt

Hinweis: Die Theorie des folgenden Skripts basiert auf dem im Ingenieurinformatik verwendeten JavaScript-Skript von Prof. Dr. Norbert Frei.

1 Einleitung

Dieses Wissensnugget erklärt wie man cloud-basierte Web-Anwendungen zur Datenvisualisierung programmiert. Es ergänzt das Wissensnugget „Cloud Computing / IoT Plattform“, welches Cloud Computing definiert und zeigt wie Daten in der Google Cloud gespeichert werden können. Mit dem Wissen dieses Nuggets können Sie auf die Daten in der Cloud zwecks Visualisierung zugreifen. Erklärt wird eine typische web-basierte Anwendung, welche HTML, CSS und JavaScript für die Darstellung der Daten verwendet. Mit anderen Worten das Wissensnugget „Cloud Computing / IoT Plattform“ demonstriert wie man Daten in die Cloud zur Speicherung transportiert, während dieses Wissensnugget cloud-basierte Web-Anwendungen zeigt wie man in der Cloud gespeicherte Daten mit Web-Technologien visualisiert.

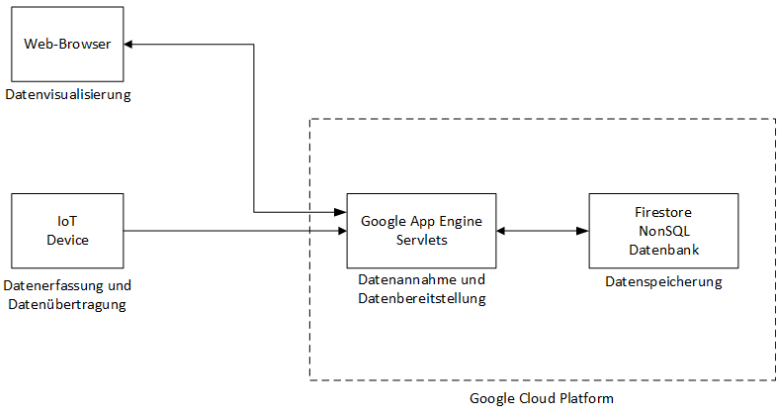

Abbildung 1: Datenerfassung und Datenvisualisierung mit Hilfe einer Cloud

2 HTML (Hypertext Markup Language)

HTML[1] oder ausgeschrieben *Hypertext Markup Language* ist eine maschinenlesbare Sprache, die für die Gliederung von Texten und anderen Daten zuständig ist. Sie strukturiert digitale Dokumente wie Texte mit Hyperlinks, Bilder und andere Inhalte. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

HTML dient dazu, einen Text semantisch zu strukturieren, nicht aber zu formatieren. Die visuelle Darstellung ist nicht Teil der HTML-Spezifikationen und wird durch den Webbrowser und Gestaltungsvorlagen wie CSS (siehe Kapitel 3) bestimmt.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Titel der Webseite</title>
    <link rel="stylesheet" type="text/css" href="stylefolder/mystyle.css">
    <!-- Kommentare werden im Browser nicht angezeigt. -->
  </head>
  <body>
    <p>Inhalt der Webseite</p>
  </body>
```

Figure 1, HTML: Grundgerüst

[1] HTML kann mit einem einfachen Editor programmiert werden, beispielsweise Visual Studio Code. Die HTML Dateien werden im Verzeichnis eines Web-Servers abgelegt, von wo sie der Web-Browser via Netzwerk mit einer ULR (z.B.: http://www.example.com/index.html) abholen kann.


```
<!DOCTYPE HTML>
```

Die Dokumenttyp-Deklaration am Anfang eines jeden HTML-Dokumentes informiert über die Art des Dokumentes. Früher wurden in dieser Zeile weitere Angaben zur HTML-Version gemacht. Der aktuelle Standard für HTML-Dokumente reduziert diese Zeile aber auf das wesentliche.

```
<html>
```

Das HTML-Element umschliesst das gesamte Dokument. Die Struktur eines HTML-Dokumentes kann wie ein Stammbaum dargestellt werden. Ausgangspunkt für diese Struktur ist immer das HTML-Element, weswegen man auch vom Wurzelelement (eng. root element) spricht. Das HTML-Element wird unterteilt in die Elemente HEAD und BODY.

```
<head>
```

Das HEAD-Element (deutsch: Kopf) ist in allen HTML-Standards ein Pflicht-Element und enthält hauptsächlich technische oder dokumentarische Informationen über den Inhalt im BODY. Diese Informationen werden üblicherweise nicht im Anzeigebereich des Browsers dargestellt.
Zwingend ist mindestens die Angabe eines Namens, daher muss jedes valide HTML-Dokument einen Titel haben. Im HEAD können nur 7 sieben Element-Typen verwendet werden. Zum Beispiel das STYLE-Element für CSS-Deklarationen oder das SCRIPT-Element für Skriptsprachen wie JavaScript. Die vollständige Liste der möglichen Element-Typen und ihre Funktion, finden Sie zum Beispiel hier:
https://de.wikipedia.org/wiki/Hypertext_Markup_Language#HTML-Kopf

```
<link rel="stylesheet" type="text/css" href="stylefolder/mystyle.css">
```

Dies ist ein LINK-Element, das eine Referenz auf eine externe CSS-Datei definiert. Das erste Attribut definiert das Verhältnis zwischen dem HTML-Dokument und der verlinkten Datei. Das zweite Attribut definiert den Typ der Datei und das letzte Attribut definiert den Pfad und den Namen der Datei.
Dieses Element wird nicht zwangsläufig benötigt. Es zeigt an dieser Stelle, wie man CSS-Dateien korrekt im HTML-Dokument einbindet. Es gibt weitere Möglichkeiten CSS einzufügen, diese werden aber, auf Grund einer sauberen Trennung von Inhalt und Gestaltung, nicht empfohlen.

```
<body>
```

In das BODY-Element (deutsch: Körper) kommen alle Informationen, die auf der Webseite sichtbar sind. Das heisst, hier definieren Sie den Inhalt der Webseite.
Alle Inhalte kommen zwischen sogenannte Tags (deutsch: Bezeichner). `<p>` ist ein öffnendes Tag und `</p>` ist ein schliessendes Tag. Diese beiden Tags definieren zum Beispiel ein Paragraph (deutsch: Absatz). Zwischen diesen Elementen steht Text, der auf der Webseite sichtbar ist.
Abgesehen vom sichtbaren Teil, kann hier auch die Logik verlinkt oder direkt implementiert werden. Dafür sind die SCRIPT-Tags vorgesehen.

Weiterführende Informationen, Tutorials und Beispiele zu HTML finden Sie hier:
http://www.w3schools.com/html/default.asp
Diese Seite können Sie als Nachschlagewerk oder als Lernplattform verwenden. Sie finden alle notwendigen Informationen zum Thema HTML und finden Beispiele, die Sie direkt im Browser anpassen können.

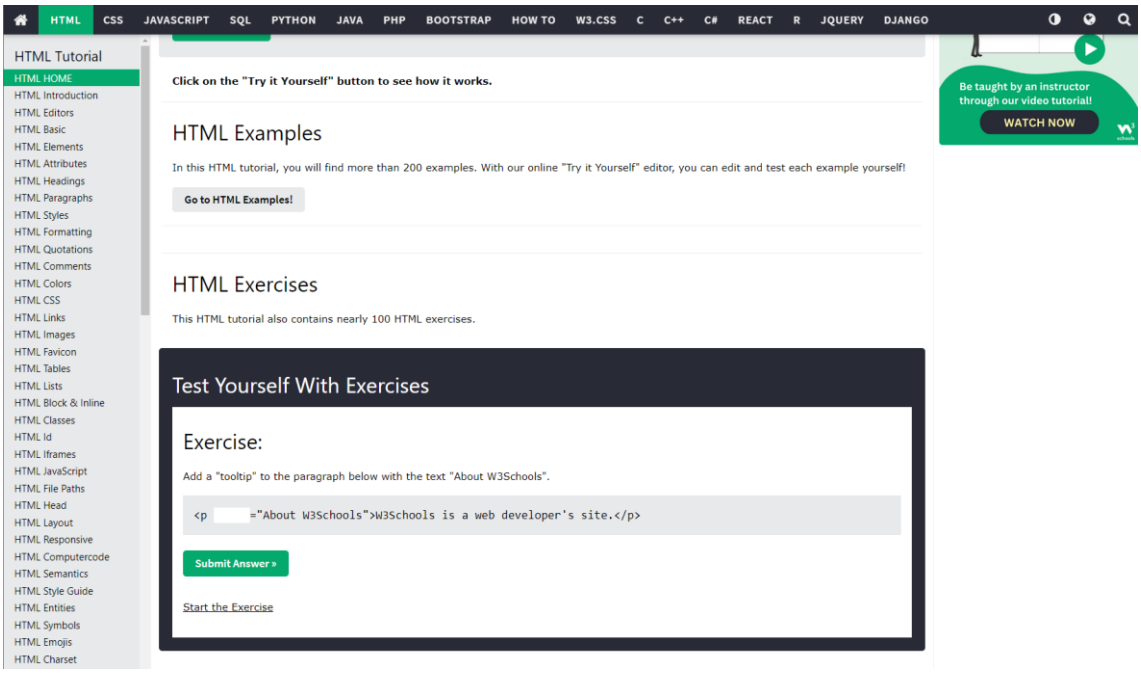

Abbildung 2: w3schools HTML Tutorial

3 CSS (Cascading Style Sheets)

CSS oder ausgeschrieben Cascading Style Sheets ist eine formale Sprache, um das Erscheinungsbild von Benutzeroberflächen festzulegen. Ein Stylesheet ist am ehesten mit einer Formatvorlage zu vergleichen. CSS gehört neben HTML ebenfalls zu den Kernsprachen des World Wide Web.

Grundidee dieser Sprache ist die Trennung von Information (Daten) und Darstellung. Das Programm, das das Stylesheet auswertet, interpretiert die zugewiesenen Daten (Text, Tabellen, Grafiken etc.) und formatiert sie (z. B. für die Bildschirmausgabe) entsprechend den vorgegebenen Regeln. Das heisst, im HTML-Dokument werden nur die inhaltliche Gliederung und die Bedeutung seiner Teile beschrieben. Die Darstellung der Inhalte (Layout, Farben und Typografie) werden ausschliesslich in einer, vorzugsweise separaten CSS-Datei festgelegt.

In CSS sind Eigenschaften innerhalb von Regelsätzen organisiert. Ein Regelsatz besteht aus einem Selektor oder einer Gruppe von Selektoren, gefolgt von einem durch geschweifte Klammern begrenzten Bereich, in dem Eigenschaften ein Wert zugewiesen wird.

```css
body {
  background-color: #d0e4fe;
}
h1, h2 {
  color: orange;
  text-align: center;
}
p {
  font-family: "Times New Roman";
  font-size: 20px;
}
```

Figure 2, CSS: Beispiel

`h1, h2`

Ist eine Gruppe von Selektoren. Sie sind der Beginn eines neuen Regelsatzes für die beiden Überschriftgrössen `h1` und `h2`. Es gibt verschiedene Möglichkeiten die Selektoren zu definieren. Die einfachste Art, ist ein Element-Selektor allein oder als Gruppe mit Komma getrennt. Zusätzlich gibt es Klassen, IDs, Pseudoelemente und viele andere Selektoren. Eine Übersicht zu den Selektoren finden Sie hier:
http://www.w3schools.com/cssref/css_selectors.php

`color:`

Das ist ein Property (deutsch: Eigenschaft). Es gibt sehr viele verschiedene Properties die auf unterschiedliche Elemente angewendet werden können. Properties folgen nach den Selektoren innerhalb von geschweiften Klammern und gelten dann für diese Selektoren. Diese Property definiert zum Beispiel die Farbe der beiden Überschriftgrössen h1 und h2.

`orange;`

Die Property Value (deutsch: Eigenschaftswert) definiert den Value der Property. Hier wurde der Property `color` der Wert `orange` zugewiesen. Die möglichen Typen der Property Value werden durch die Property definiert. Handelt es sich um eine `color` Property, dann sind Werte mögliche wie:
red; rgb(255, 0, 0); #ff0000;

Dabei handelt es sich bei allen 3 Werten um die Farbe Rot.

Weiterführende Informationen, Tutorials und Beispiele zu CSS finden Sie hier:
http://www.w3schools.com/css/default.asp
Diese Seite können Sie als Nachschlagewerk oder als Lernplattform verwenden. Sie finden alle notwendigen Informationen zum Thema CSS und finden Beispiele, die Sie direkt im Browser anpassen können.

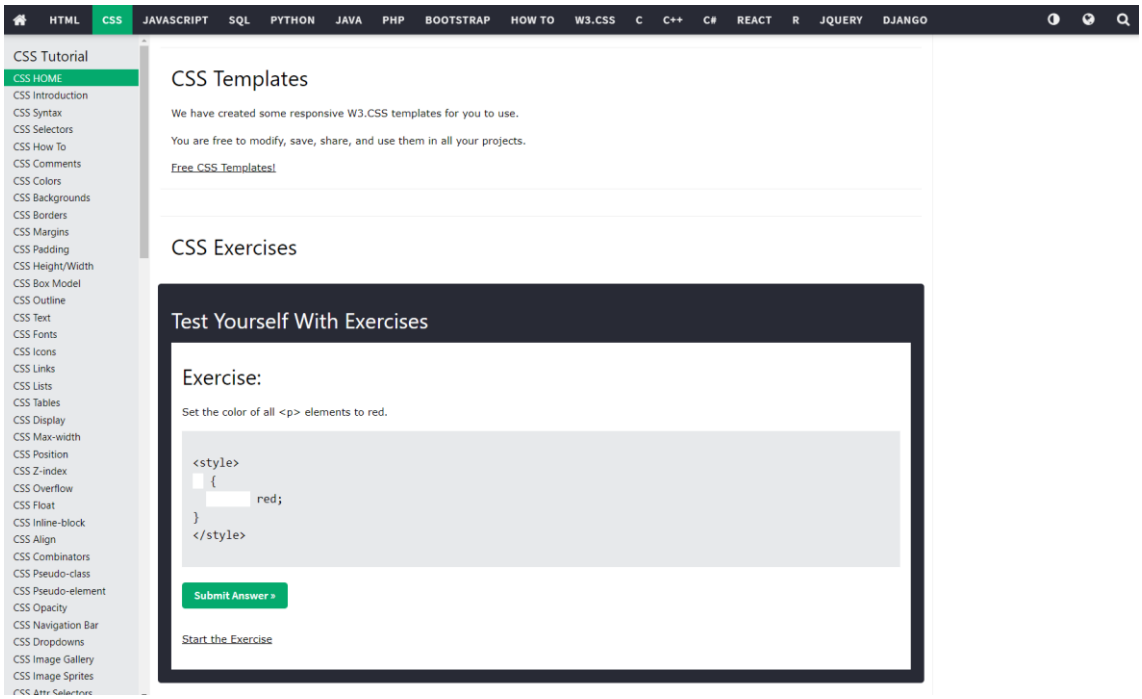

Abbildung 3: w3schools CSS Tutorial

4 JavaScript

JavaScript[2] ist eine Programmiersprache, die über einen Interpreter[3] ausgeführt wird. Sie wurde ursprünglich für dynamisches HTML in Webbrowsern entwickelt. Damit wollte man vor allem Benutzerinteraktionen auswerten, Inhalte verändern, nachladen oder generieren und so die Möglichkeiten von HTML und CSS erweitern. Heute findet JavaScript auch außerhalb von Browsern Anwendung, so etwa auf Servern und in Microcontrollern.

JavaScript ist eine dynamisch typisierte und objektorientierte Sprache. Sie wird den objektorientierten Programmierparadigmen unter anderem auf der Basis von Prototypen gerecht. In JavaScript lässt sich objektorientiert und sowohl prozedural als auch funktional programmieren (Im Folgenden wird der Begriff Funktion manchmal auch für Prozeduren verwendet, weil streng genommen eine Funktion ein Resultat liefert. In JS gibt es nur Funktionen → ohne Resultat→ Prozedur).
Die Version, die von den meisten aktuellen Browsern unterstützt wird, ist ES6.

```
<!DOCTYPE HTML>
<html>
  <head>
    <script>
      function myFunction() {
        document.getElementById("demo").innerHTML = "Paragraph changed.";
      }
    </script>
  </head>
  <body>
    <p id="demo">A Paragraph</p>
    <button type="button" onclick="myFunction()">Click here</button>
  </body>
</html>
```

Figure 3, JavaScript: Beispiel

```
<script>
```

Das SCRIPT-Element umschliesst den gesamten JavaScript-Teil. Darin ist die Funktion `myFunction()` definiert. Eine Funktion ist ein Block aus JavaScript-Code der aufgerufen und ausgeführt werden kann. Dieser Block kann im HEAD-Element oder auch im BODY-Element definiert werden.
In diesem Beispiel wird die Funktion `myFunction()` durch einen Klick auf einen Button aufgerufen. Ist die Funktion aktiv sucht sie das HTML-Element mit der ID „demo“ und tauscht den Wert dieses Elements mit dem oben definierten String „Paragraph changed.“ aus. Die Änderung wird im Browser sofort sichtbar.[4]

```
<button type="button" onclick="myFunction()">Click here</button>
```

[2] JavaScript hat nichts mit Java zu tun. Die Namensgebung beruht rein auf marketingtechnischen Überlegungen. Beide Sprachen wurden ungefähr zur selben Zeit populär und Java durchlebte einen richtigen Hype. JavaScript bekam seinen Namen, um von eben diesem Hype zu profitieren.

[3] Ein Interpreter ist ein Computerprogramm, das in einer Programmier- oder Skriptsprache geschriebene Anweisungen direkt ausführt, ohne dass diese zuvor in ein Maschinensprachprogramm kompiliert worden sein müssen.

[4] Lädt ein Web-Browser eine HTML Seite vom Dateisystem oder via Netzwerk, so wird diese Seite im Speicher abgelegt. Der gespeicherte Inhalt der HTML Seite nennt man das Document Object Model (DOM). Auf dieses Objektmodell kann über eine standardisierte Schnittstelle zugegriffen werden. Die Modifikationen des Objektmodells werden vom Browser unmittelbar visualisiert.

Das ist ein BUTTON-Element auf das geklickt werden kann. Das Attribut `onclick` definiert das Verhalten, wenn auf den Button geklickt wird. In diesem Fall, wird bei einem Click-Event die Funktion `myFunction()` aufgerufen.

```
<p id="demo">A Paragraph</p>
```

Dieses P-Element hat die ID „demo“. Wird die Funktion `myFunction()` aufgerufen, findet diese dank der ID dieses Element und verändert seinen Wert.
JavaScript kann auch in einem externen File definiert werden. Diese Art ist zu bevorzugen. Damit wird die saubere Aufteilung von Inhalt, Formatierung und Funktion ermöglicht und vereinfacht so die einzelne Bearbeitung. Dafür erstellt man eine neue Datei `myScript.js` und schreibt alles was vorher zwischen den script-tags stand in diese Datei. Im HTML-Dokument muss dann ähnlich wie für die CSS-Datei eine Referenz gesetzt werden. Diese Referenz kommt in ein script-tag, welches dann im File selbst nicht mehr verwendet werden muss. Die Referenz kann im HEAD oder im BODY definiert werden. Das Script verhält sich so, als würde es genau an der Stelle, an der die Referenz definiert ist, stehen.

```
<!DOCTYPE HTML>
<html>
  <head>
  </head>
  <body>
    <script src="scriptfolder/myScript.js"></script>
  </body>
</html>
```

Figure 4, JavaScript: Script Referenz im BODY

Verschaffen Sie sich einen Überblick über die Grundlagen von JavaScript in den Webunterlagen von w3schools: http://www.w3schools.com/js/default.asp.

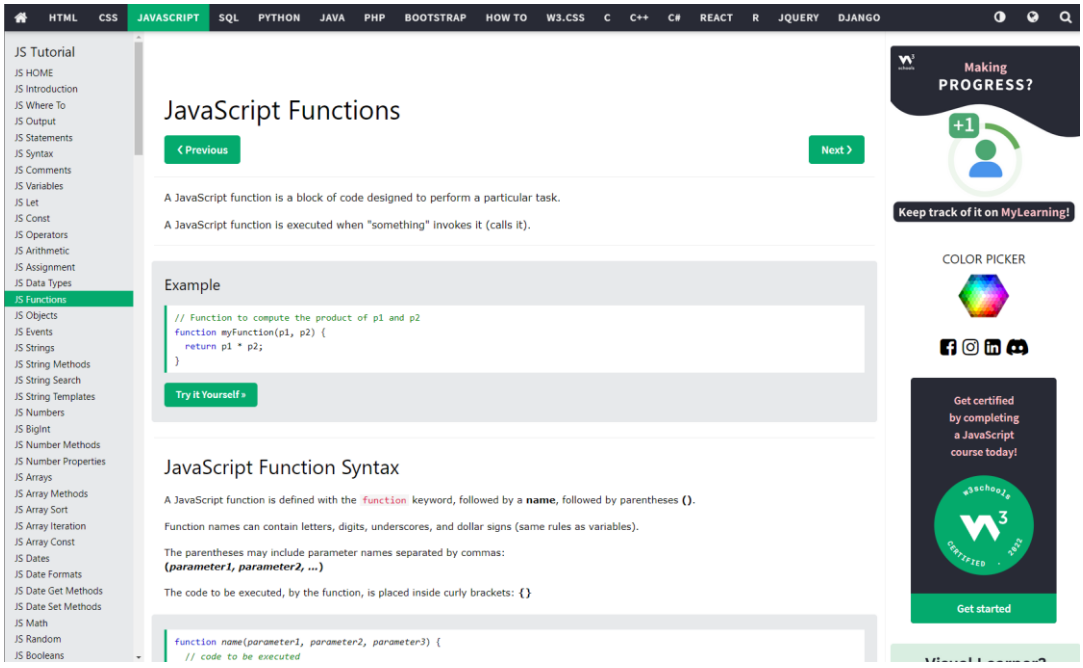

Abbildung 4: w3schools JavaScript Tutorial

Weiterführende Informationen, Tutorials und Beispiele zu JavaScript finden Sie hier:
http://www.w3schools.com/js/default.asp

Diese Seite können Sie als Nachschlagewerk oder als Lernplattform verwenden. Sie finden alle notwendigen Informationen zum Thema JavaScript und finden Beispiele, die Sie direkt im Browser anpassen können.

5 Aufgaben

Im GitLab der OST finden Sie das Projekt «FrontendDevelopment» (https://git-lab.ost.ch/wn_cloudcomputing/wn_frontenddevelopment). Dieses Projekt können Sie klonen. Sofern Sie Ihre Anpassungen in GitLab speichern wollen, empfehle ich Ihnen einen Fork des Projekts zu erstellen. Ein Fork erstellt eine Kopie des Projekts und ermöglicht es Ihnen daran zu arbeiten, ohne dass das ursprüngliche Projekt beeinflusst wird.

Starten Sie das Projekt gemäss dem README des Projektes. Das Projekt beinhaltet einen kleinen Node.js-Server, welcher die erstellten HTML-, CSS-, JS-Files bereitstellt. Zusätzlich generiert er Beispieldaten und stellt diese über eine Schnittstelle zur Verfügung. Bei Interesse können Sie sich das «server.js»-File gerne anschauen und studieren. Ist für die folgenden Aufgaben jedoch nicht zwingend notwendig.

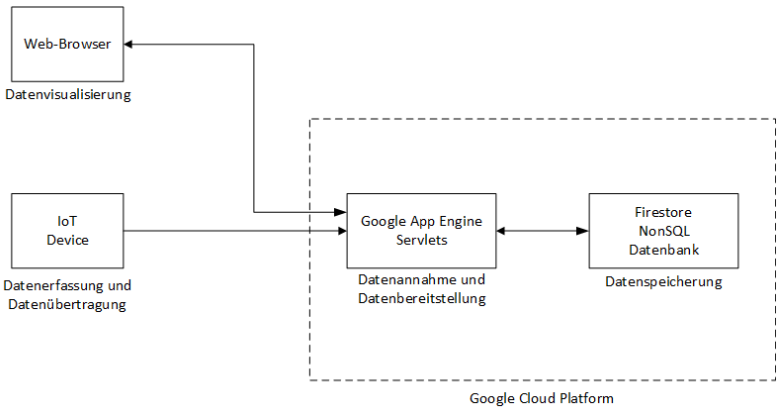

Abbildung 5: Datenerfassung und Datenvisualisierung mit Hilfe einer Cloud

Eine mögliche Lösungsvariante zu den Aufgaben finden Sie im selben GitLab-Projekt unter dem Branch «exercise-solutions».

Es gibt viele Möglichkeiten wie Sie das Aussehen oder den Inhalt manipulieren können. Mit diesen Übungen soll Ihnen eine Basis geboten werden, auf der Sie in Eigeninitiative aufbauen können.

5.1 Einfache Visualisierungsmöglichkeiten

5.1.1 Beispiel: Motorenstatus als Hintergrundfarbe abbilden

Im Ordner «VisualisingMotorData» finden Sie je eine HTML-, CSS- und JavaScript-Datei vor. Das HTML-File bildet ein kleines Grundgerüst und verlinkt das CSS-File, sowie das JavaScript-File.
Schauen Sie sich durch die drei Dateien und versuchen Sie zu verstehen, wie diese miteinander interagieren.

Folgend wird kurz auf die Logik im JavaScript-File eingegangen.
Die Funktion «getMotorData()» erstellt Ihnen eine REST-Anfrage und fragt den Motor Status ab. Sobald diese erfolgreich war, wird der aktuelle Status an die Funktion «visualizeMotorData(motorState)» übergeben. Da werden nun zwei verschiedenen Eigenschaften im HTML-File manipuliert.

Einerseits soll im Element mit der Id «motorState» der enthaltene Text mit dem erhaltenen Motoren Status überschrieben werden.
Anderseits soll die Webpage, je nach Motoren Status eine andere Farbe erhalten. Dazu wird dem body-Element die Hintergrundfarbe gesetzt.

5.1.2 Aufgabe: Motorenstatus mit Bildern darstellen

Eine weitere Möglichkeit den Status des Motors darzustellen ist mithilfe von Bildern. HTML besitzt ein spezifisches Element für Bilder.
Ersetzen Sie das im Beispiel verwendete p-Element mit dem folgenden img-Element:

```
<img id="motorState" src="img/power-off.svg" style="width:200px"/>
```

Dabei geben wir an, wo das verwendete Bild gefunden werden kann und wie breit dies sein soll.

Wenn Sie nun testen, werden Sie schnell feststellen, dass jedes Mal das «Off» Bild dargestellt wird. Um auf den Motoren Status zu reagieren, muss die Logik im JavaScript-File etwas verändert werden.
Anstelle der Hintergrundfarbe muss nun die src-Eigenschaft des img-Elements angepasst werden. Suchen Sie dazu zuerst das img-Element mit der Id «motorState» und setzen dann die src-Eigenschaft neu. Erhalten Sie den Motoren Status «On» so verwenden sie das Bild «img/power-on.svg», beim Status «Off» das Bild «img/power-off.svg».

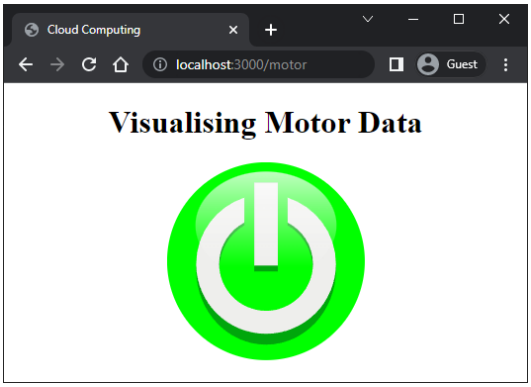

Abbildung 6: Power ON

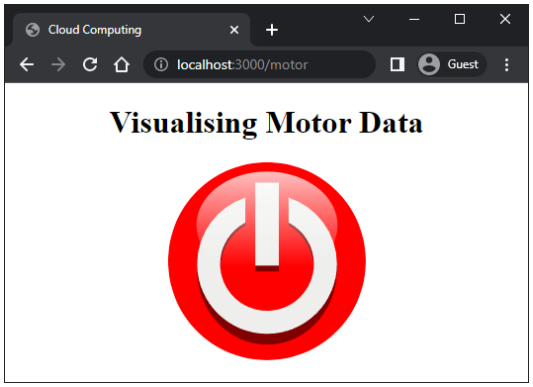

Abbildung 7: Power OFF

5.2 Visualisierung mit Google Charts

Google Charts bietet Ihnen eine einfache Möglichkeit zur Darstellung von Daten mithilfe von Diagrammen. Dabei können Sie unter mehreren Diagrammtypen auswählen, wie Sie unter https://developers.google.com/chart/interactive/docs/gallery entnehmen können.
Folgend gehen wir auf zwei Typen ein, die wir für Sensordaten als sinnvoll erachten.

5.2.1 Beispiel: Darstellung von Sensordaten mit Google Charts Tabelle

In diesem Beispiel ist es das Ziel, die über eine Schnittstelle zugreifbaren Sensordaten in einer Tabelle darzustellen. Öffnen Sie das Beispiel im Ordner «VisualisingSensorData».
Hinweis: Haben Sie bereits vorgängig das Wissensnugget «03.3.1 Cloud Computing / IoT Plattform» bearbeitet, so können Sie die URL in der Funktion «getSensorData()» auf Ihre Cloud verweisen und anstelle der Sample-Daten Ihre eigenen Daten laden.

Google Charts bietet eine sehr saubere Dokumentation mit einigen Beispielen. Deshalb möchte ich folgend nur auf ein paar Punkte eingehen und verweise Sie auf die Google Charts Dokumentation der Tabelle (https://developers.google.com/chart/interactive/docs/gallery/table).

Um eine Tabelle zu zeichnen, muss erst ein div-Element verlinkt werden, welches als Container verwendet wird. Diesem Container kann dann das erstellte «DataTable»-Objekt übergeben werden. Das «DataTable»-Objekt beinhaltet die Daten, welche dargestellt werden sollen.

Testen und verändern Sie die Tabelle mithilfe der Dokumentation.

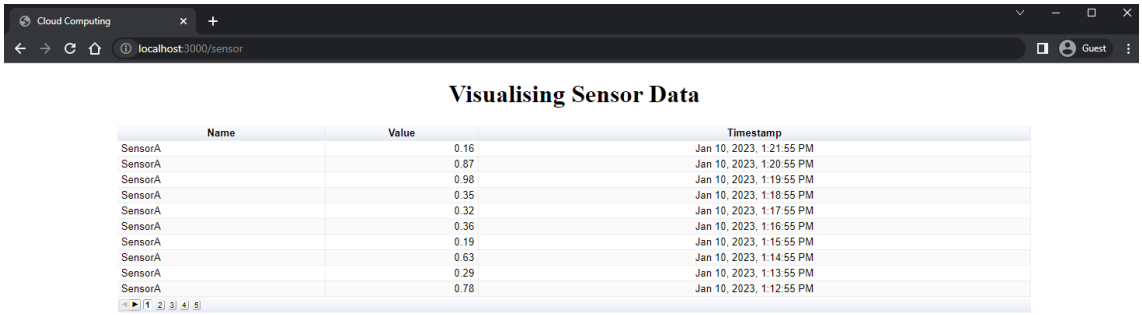

Abbildung 8: Visualisierung der Daten mit Hilfe der Google Charts Tabelle

5.2.2 Aufgabe: Darstellung von Sensordaten mit Google Line Chart

Neben der Tabelle ist es auch denkbar, die Sensordaten in einem Liniendiagramm darzustellen.

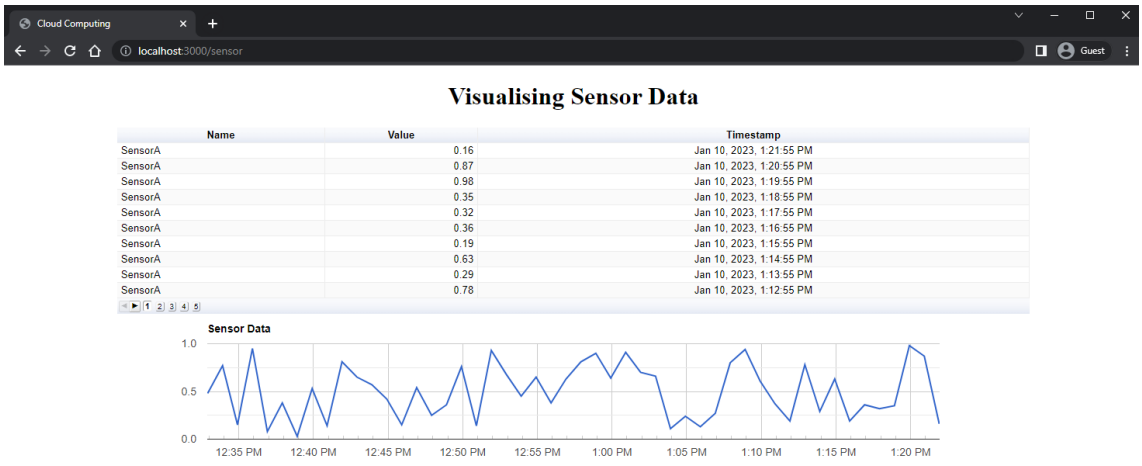

Abbildung 9: Visualisierung der Daten mit Google Line Charts

Lesen Sie in der Dokumentation der Google Line Chart (https://developers.google.com/chart/interactive/docs/gallery/linechart) nach, wie Sie die Sensordaten als Liniendiagramm darstellen können. Versuchen Sie beide Diagramme darzustellen.

Nachfolgend einige Tipps die Ihnen bei der Umsetzung helfen können.

1. Erstellen Sie wie bei der Tabelle ein «DataTable»-Objekt.
2. Überlegen Sie sich gut, welche der Daten, die in der Tabelle vorhanden sind, im Liniendiagramm Sinn ergeben und demnach ins «DataTable»-Objekt gehören.

3. Um beide Diagramme darstellen zu können, benötigen Sie einen zweiten Container (neues div-Element).
4. Stellen Sie sicher, dass die Line Chart Komponente zu Beginn der JavaScript-Datei geladen wurde.

5.2.3 Aufgabe: Mathematische Berechnungen

Bis anhin wurden die Sensordaten lediglich visuell dargestellt. Es ist aber auch denkbar diese zu verarbeiten und Kennzahlen zu berechnen. In den folgenden Aufgaben werden einige Interessante und wissenswerte Kennzahlen berechnet.

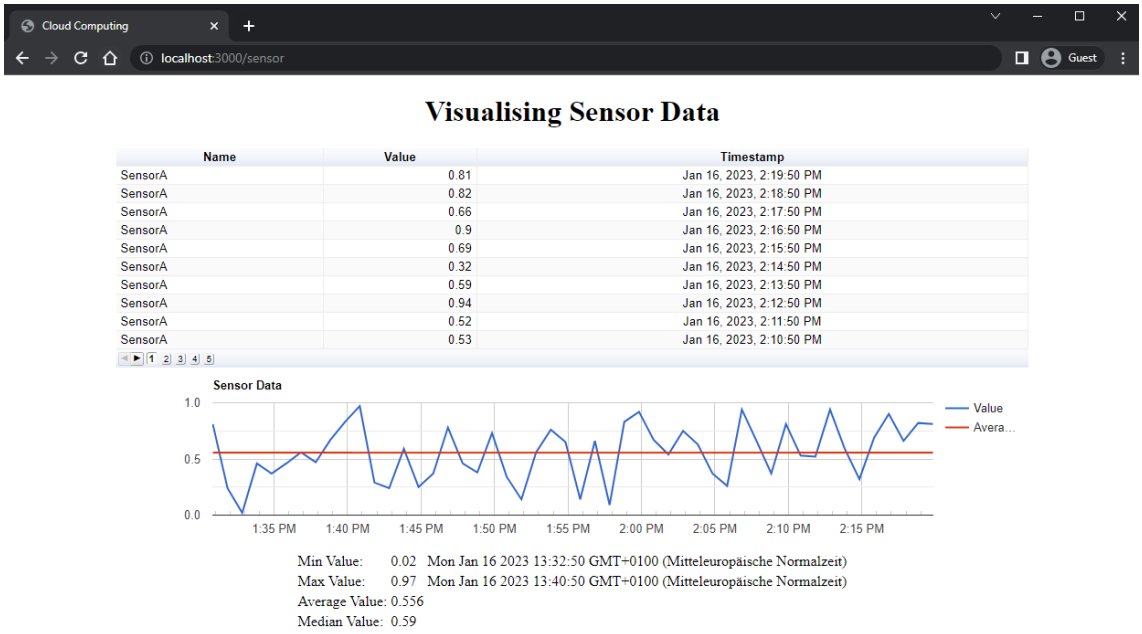

Abbildung 10: Mathematische Berechnungen

5.2.3.1 Darstellung der Kennzahlen als Tabelle

Bevor wir uns damit beschäftigen, wie die Kennzahlen berechnet werden, erstellen wir eine HTML-Tabelle, welche die Kennzahlen anzeigen wird. Lesen Sie auf w3schools nach, wie eine Tabelle erstellt wird. Die Tabelle soll für jede der vier Berechnungen eine Zeile enthalten.
Die Zeile für Median und Durchschnitt benötigt eine Zelle für die Kennzeichnung und eine für den Wert. Die Zeile für das Minimum und Maximum benötigt noch eine weitere Zelle für das Datum.
Denken Sie daran, um im JavaScript den Wert und das Datum zu setzen, sollte die Zelle mit einer ID gekennzeichnet werden. So kann wie in Beispiel 5.1.1 der Wert in JavaScript gesetzt werden.
Im CSS-File wurde für das «table»-Element bereits hinterlegt, dass dieses zentriert sein soll.

5.2.3.2 Berechnung des Minimums

Der Funktionskopf und der Aufruf der Funktion «calcMin» sind bereits im sensorData.js vorhanden. Ziel soll es sein, das Objekt des Arrays zurückzugeben, welches den kleinsten Wert beinhaltet, damit auch der Zeitpunkt, an dem der minimale Wert gemessen wurde, bekannt ist.

Als erstes soll ein initialer Wert für die Variable «minValue» gesetzt werden. Da der der kleinste Wert gesucht ist, sollte dieser sehr gross gewählt werden. Besser noch, setzen Sie den ersten Wert des Arrays, welches als Parameter übergeben wurde, als «minValue».

Danach kann über das gesamte Array iteriert werden, zum Beispiel mit einer for-of-Schleife. Bei jedem Element sollte geprüft werden, ob der im «SensorData»-Element enthaltene «Value» kleiner ist als der im «minValue.value» gespeicherte Wert. Falls dies der Fall ist, so soll das aktuelle Element den «minValue» überschreiben.

Am Ende der Funktion geben Sie den nun im «minValue» gespeicherte Wert zurück.

Der Funktionsaufruf existiert bereits, setzen Sie den nun erhaltenen minimalen Wert in die bereits vorbereitet Tabelle.
Wissen Sie die ID noch, welche Sie der Zelle für «minValue.value» und «minValue.date» gegeben haben?
Wissen Sie noch, wie Sie den Text eines HTML-Elements in JavaScript verändern können? Falls nicht, so schauen Sie sich nochmals das Beispiel 5.1.1 an.

5.2.3.3 Berechnung des Maximums

Die Berechnung des Maximums funktioniert auf gleiche Art und Weise wie die Berechnung des Minimums. Sie dürfen die vorherige Funktion kopieren, überlegen Sie sich aber gut, was der Unterschied ist.

Tipp: Neben dem Variabelnnamen, welcher für das Maximum unpassend ist, gibt es genau ein Zeichen, welches angepasst werden muss.

5.2.3.4 Berechnung des Durchschnitts

Bei der Berechnung des Durchschnitts, wird nur noch ein einzelner Zahlenwert verlangt und kein «SensorData»-Objekt. Deshalb ist einer der ersten Schritte das Transformieren des «SensorData»-Objekts in ein Float-Array. Dieser Schritt ist bereits implementiert und in der Konstante «values» ist das Float-Array enthalten.

Um den Durchschnitt zu berechnen wird die Summe aller Werte und die Länge des Arrays benötigt. Es gibt unterschiedliche Lösungsansätze. Einer wäre wie zuvor über das Array zu iterieren. Zu bevorzugen sind allerdings die Funktionen, welche der Typ «Array» anbietet, als Beispiel die Funktion «reduce(…)». Schauen Sie auch nach, was sonst für Funktionen existieren und wie Sie die Länge des Arrays erhalten.

Sind diese beiden Werte vorhanden, so kann der Durchschnitt berechnet werden.
Eine Möglichkeit ist es, diesen Wert noch zu Runden mit «Math.round(…)». Passen Sie jedoch auf, dieser rundet ganzzahlig und die Sensorwerte liegen zwischen 0 und 1.
Tipp: Den Wert mit X (Zehnerpotenz) multiplizieren und nach dem Runden mit X dividieren.

Neben dem Einfügen des Durchschnittes in die Tabelle, ist es auch denkbar diesen im Line-Chart darzustellen. Dazu muss das «DataTable»-Objekt um die Spalte «Average» vom Typ «number» ergänzt werden. Jedem Datenpunkt soll anschliessend zusätzlich der berechnete Durchschnitt mitgegeben werden.

5.2.3.5 Berechnung des Medians

Eine weitere interessante Kennzahl ist der Median, bei welchem je 50% der Werte oberhalb bzw. unterhalb liegen.

Um den mittleren Index des Arrays zu verwenden, muss das Array erst mal sortiert werden. Wie bereits zuvor erwähnt, bietet der Typ «Array» einige Funktionen an, darunter auch eine Sortiermöglichkeit.

Ist das Array sortiert, so können Sie den mittleren Index des Arrays als Median verwenden. Unterscheiden Sie jedoch zwei Fälle: eine gerade Anzahl und eine ungerade Anzahl an Sensorwerten.

