Brückenkurs Programmieren

Für Social Media Systems

SoSe20

THM
TECHNISCHE HOCHSCHULE MITTELHESSEN

Gliederung

Einführung in die THM

Wir lernen Program-mieren

Challenges

Einführung in die THM

- Ziel dieses Brückenkurses:
  - Grundkenntnisse der Programmierung
  - Leichteren Start in das Modul „Webbasierte Programmierung 1“ (webP1)
  - Bonuswissen zum Studiengang

Einführung in die THM

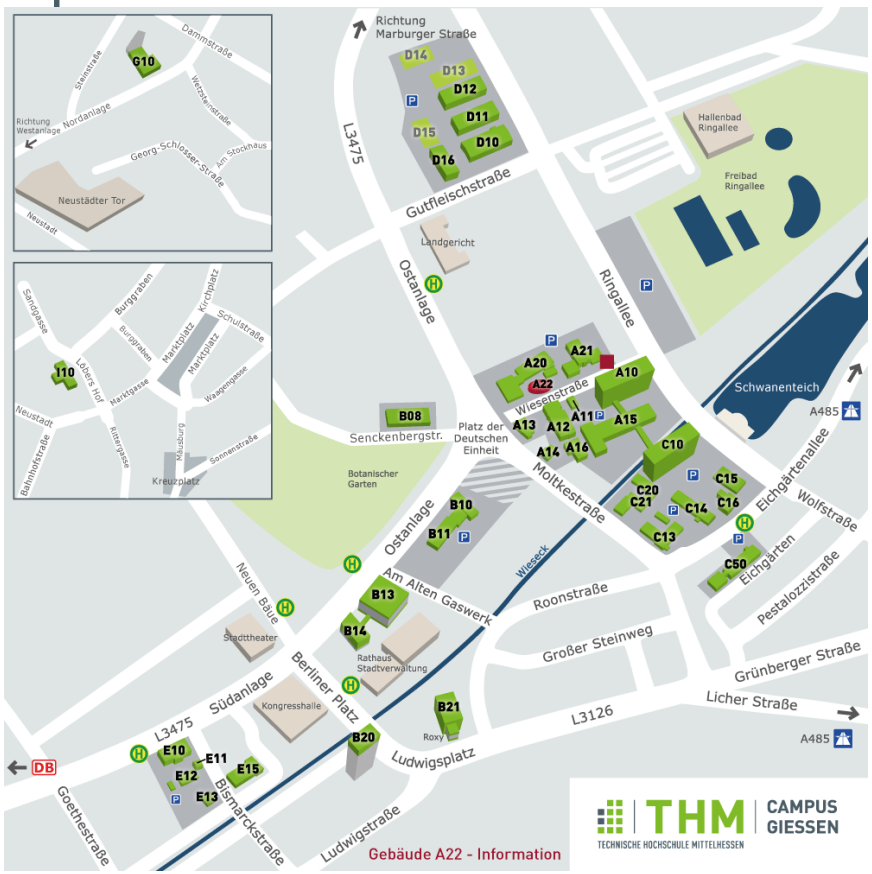
- Zum Ablauf:
  - Montag bis Freitag je 9-16 Uhr
  - Einstündige Mittagspause gegen 12 Uhr
  - Eigenständige Zwischenpausen
  - Evaluation am Ende zum Ende des Kurses
  - Jederzeit Möglichkeiten zum Gespräch mit Dozenten/-innen des Grundstudiums

Einführung in die THM

- Wer sind wir?
  - Campusse in Gießen, Friedberg und Wetzlar
  - Außenstellen in Bad Hersfeld, Bad Wildungen, Bad Vilbel, Biedenkopf und Frankenberg
  - Gegründet 1971 als FH Gießen-Friedberg
  - 2010 Umbenennung in THM
  - Erste Vorgänger schon 1838 in Gießen
  - Aktuell fast 19.000 Studierende somit die größte FH in Hessen und drittgrößte in Deutschland
  - Aktueller Präsident: Prof. Dr. Matthias Willems

Einführung in die THM

- Unser Campus in Gießen



Einführung in die THM

- Unser Campus in Gießen

- A20 – Café Campus-tor (Kleine Mensa)
- A22 – Information
- A10 – Mensa & Pastaria (ehem. A)
- A12 – Fachschaft, „-“-Informatiker
- C10 – Bibliothek (ehem. C)

Einführung in die THM

- Hochschulweite Services
  - Benutzerkonto (Kürzel und Passwort)
  - WLAN, Eduroam und VPN (Netzpasswort)
  - Emailpostfach
  - Studentenwerk und Mensen
  - News und Infos
    - Webseiten
    - Emailverteiler
    - Discord
    - Vor Ort

Einführung in die THM

- Erstes Kennenlernen
  - Wohn- / Herkunftsort?
  - Warum SMS?
  - Schon mal programmiert?
  - Welche sozialen Medien nutzen Sie?

Wir lernen Programmieren

Wir lernen Programmieren

- Was ist ein Programm?
  - Geordnete Folge von Anweisungen
  - Vergleichbar mit den Schritten eines Kochrezepts
  - Flexibler Ablauf möglich
    - Wenn … dann … ansonsten …
    - Solange wie … mach …

Wir lernen Programmieren

- Benötigte Programme
  - Webstorm
  - Node.js
  - TypeScript
    (kommt bei Webstorm automatisch mit)

Übung

- THM Ersteinrichtung (Benutzer- und Netzpasswort) ggf. Immatrikulationsunterlagen abholen
- JetBrains Studierendenaccount anlegen
- Benötigte Programme installieren und einrichten
- Alternativ (für diejenigen ohne Laptop): Bei der Installation/Einrichtung zusehen und zeitnah nachholen

Wir lernen Programmieren

- Informatik heißt klassisch auch „Datenverarbeitung“ (EDV)
- Eingangsdaten → Magie → Verarbeitete Daten
- Eingabe
  - Vom Benutzer
  - Von Sensoren
  - Von anderen Programmen / Programmteilen
- Ausgabe
  - An den Benutzer
  - An andere Programme / Programmteile
  - An einen Speicher (Festplatte / Datenbank)

Wir lernen Programmieren

- Für uns erstmal:
  Ein- und Ausgabe von/an den Benutzer
- In der Webentwicklung daher per Webseite

- Ablauf
  - Übersicht „Wie funktioniert das Web?“
  - Grundverständnis HTML
  - Programmieren mit TypeScript

Wir lernen Programmieren

- Wie funktioniert das Web?

- HTML beschreibt den Aufbau einer Webseite
- CSS beschreibt das Design einer Webseite
- Type- / JavaScript bringt die Dynamik

Wir lernen Programmieren

- Wie funktioniert das Web?

- Request / Response System
  - Ein Client (Browser) stellt eine Anfrage an einen Webserver:
    „Hey www.thm.de, schick mir deine Seite!“
  - Der Server antwortet, falls die Anfrage korrekt war:
    „Hey Client, hier hast du die nötigen HTML, CSS und JS-Dateien für die Seite.“
  - Der Browser nimmt diese (Text-) Dateien und stellt sie für den Benutzer dar.

Wir lernen Programmieren

- HTML ist eine Beschreibungs- bzw. Auszeichnungssprache (Markup Language)
- Es gibt nur zwei „syntaktische“ Bestandteile
  - Elemente (Bspw. und
)
    - Stehen in spitzen Klammern
    - Sind umschließend als <element> </element> Paar
    - Oder einzelstehend als <name> oder <name/>
    - Können beliebig hierarchisch verschachtelt werden Z.B.
  - Attribute (Bspw.)
    - Beschreiben Elemente
    - Werden als name=„wert“ Paar angegeben

Wir lernen Programmieren

- Das HTML-Grundgerüst ist das Fundament für jede Webseite

```
<!DOCTYPE HTML>
<html>
<head>
   <title> Mein Webseitentitel </title>
</head>
<body>
   Hier wird die Webseite beschrieben
</body>
</html>
```

Wir lernen Programmieren

- Benötigte HTML-Elemente
  - <h1></h1> bis <h6></h6> für Überschriften
  - zum gruppieren
  - für Textabsätze
  - <a> für Links (immer mit href-Attribut)
  - und mit für Listen
  - <input> für Eingabemöglichkeiten
    - type=„text“ für Textfelder
    - type=„button“ für Knöpfe
    - type=„radio“ für Einzelauswahlen
    - ...

Wir lernen Programmieren

- Benötigte HTML-Attribute
  - id um einem Element einen eindeutigen Bezeichner zu geben.
     Hier steht Text
    Die ID wird benötigt um später ein konkretes Element auszuwählen um dessen Inhalt zu lesen oder zu verändern
  - class um mehreren Elementen einen gemeinsamen Bezeichner zu geben.
    <div class=„gruppe1 farbe3“> ...
    Klassen werden meist für das spätere Styling in gleichbleibender Optik verwendet

Wir lernen Programmieren

- Mit TypeScript kommt nun die „Action“
- Grundlagen:
  - Der Browser „versteht“ nur JavaScript, daher übersetzt Webstorm unser gutes TS in böses JS
  - Die Webseite muss wissen, dass ein Skript zu ihr gehört, dazu wird es im HTML-Code referenziert <script src=„script.js“></script> im Head
  - TS ist eventgesteuert, dh. eine Aktion löst eine Reaktion aus
    Z.B. der Klick auf einen Button

Wir lernen Programmieren

- Zuerst muss abgewartet werden, bis die Webseite vollständig geladen wurde, da sonst Selektionen nicht möglich sind

```
$(() => {
    Hier kommt später der ganze Code hin
});
```

Wir lernen Programmieren

- Eingabe z.B. per Button und Textfeld

```
<input id=„machWas“ type=„button“ value=„Klick mich“>
<input id=„eingabe“ type=„text“>
```

- Ausgabe z.B. per Textabsatz

```
<p id=„ausgabe“></p>
```

- Action per Eventlistener

```
$(„#machWas“).on(„click“, ()=>{
    Hier kommt die Magie rein
});
```

Wir lernen Programmieren

Wir lernen Programmieren

- Grundelement jeder prozeduralen / objektorientierten Sprache ist die Variable
  - Eine Variable ist einfach nur ein Speicherplatz mit einem beliebigen Namen
  - In diesen Speicherplatz kann man genau ein Ding ablegen
  - Legt man etwas neues ab, so überschreibt / löscht man den vorherigen Wert
  - Den Inhalt einer Variable kann man beliebig oft lesen / benutzen ohne ihn zu ändern

Wir lernen Programmieren

- In typisierten Sprachen passt in eine Variable nur Dinge des gleichen Typs
  - Ein Typ von Dingen sind z.B. Zahlen (number), oder Texte (string)
  - Bei der Erzeugung einer Variablen (Deklaration) muss man angeben, welchen Typ von Dingen man darin speichern kann
  - In eine Variable vom Typ number kann man keine Texte sondern ausschließlich Zahlen speichern etc.

Wir lernen Programmieren

- Und so sieht es konkret aus:
  - let x: number; — Erzeugt eine neue Variable namens „x“ vom Typ number.
  - x = 5; — Speichert die Zahl 5 in die Variable x.
  - x = 10; — Speichert die Zahl 10 in die Variable x, die 5 wurde damit überschrieben.

Wir lernen Programmieren

- Und so sieht es konkret aus:
  - let hans: string; — Erzeugt eine neue Variable namens „hans“ vom Typ string.
  - hans = „Hallo“; — Speichert den Text Hallo in die Variable hans.
  - hans = „Welt“; — Speichert den Text Welt in die Variable hans, der Text Hallo wurde damit überschrieben

Wir lernen Programmieren

- Und so sieht es konkret aus:
  - let zahl1: number = 5;
  - let zahl2: number = 8;
  - zahl2 = zahl1;
  - zahl1 = 3 * zahl1;

Es geht auch in einem Schritt.

Der alte Wert von zahl2 (8) wird mit dem Wert aus zahl1 überschrieben.

Der Wert von zahl1 wird mal drei genommen und wieder abgespeichert.

Wir lernen Programmieren

- Zu dem Typ number stehen uns die mathematischen (+, -, *, /, %, **) und relationalen (>, <, >=, <=, ==, !=) Operatoren zur Verfügung
- Zu dem Typ string hauptsächlich der Konkatenationsoperator (+) um Zeichenketten zu verbinden.

Wir lernen Programmieren

- Das waren schon die wichtigsten Grundlagen, jetzt kommt die Erweiterung!
- Oftmals ist das Programm abhängig von Entscheidungen
- Außerdem gilt immer das Informatiker-Mantra: „Don‘t repeat yourself!“

Wir lernen Programmieren

Bedingte Anweisungen

Wir lernen Programmieren

- Programmabläufe sind nicht immer in Stein gemeißelt, sondern müssen flexibel sein
- Dazu gibt es bedingte Anweisungen, die nur ausgeführt werden, wenn eine Bedingung zutrifft (Wenn ... dann ...)
- Im Negativfall kann eine Alternative ausgeführt werden (Wenn ... dann ... ansonsten ...)

Wir lernen Programmieren

- Wenn ... dann ...

  if (*Bedingung*) {

  Anweisungen

  }

  - Die Anweisungen werden nur ausgeführt, wenn die Bedingung wahr ist

Wir lernen Programmieren

- Wenn ... dann ... ansonsten

```
if (Bedingung) {
  Anweisungen
} else {
  Anweisungen
}
```

  - Es werden nur die oberen oder die unteren Anweisungen ausgeführt, je nach dem, ob die Bedingung wahr oder falsch ist

Wir lernen Programmieren

- Als Bedingung dient immer ein Wahrheitswert (true, false)
- Wahrheitswerte sind vom Typ boolean
- Booleans sind auch das Ergebnis von relationalen Operationen
  - 3 < 4 → true
  - 3 >= 4 → false
  - 5 === 6 → false
- Zudem sind logische Operationen möglich (! nicht, || oder, && und)

Wir lernen Programmieren

- Bedingte Anweisungen können beliebig häufig wiederholt werden
- Stehen sie nacheinander, so werden sie auch alle einzeln geprüft und ggf. ausgeführt
  - Beschreiben mehrere Bedingungen den gleichen Sachverhalt kann diese Einzelprüfung leicht zu Fehlern führen
  - Eine Verschachtelung der Bedingungen ist dann sinnvoll
- Bedingte Anweisungen sollten gleichartig aufgebaut sein um Fehler zu vermeiden

Wir lernen Programmieren

Wir lernen Programmieren

- Wiederholte Anweisungen in gleicher oder ähnlicher Form bieten ausgeschrieben ein hohes Fehlerpotential
- Zudem kann die Wiederholung so häufig sein, dass man sie nicht ausschreiben kann/will
- Die Anzahl der Wiederholungen können auch unbekannt oder abhängig von Benutzereingaben sein

Wir lernen Programmieren

- Die Lösung ist in diesem Fall eine zählergesteuerte Schleife (sog. For-Schleife)
- Eine For-Schleife besteht aus zwei Hauptteilen ähnlich dem If
  - Im Kopf steht die Schleifensteuerung bestehend aus Zählvariable und Startwert, Abbruchbedingung und Schrittweite
  - Im Körper stehen die zu wiederholenden Anweisungen

Wir lernen Programmieren

- Start, Ende, Schritt

```
for (let i = startwert; i < endwert; i = i + schritt) {
  Anweisungen
}
```

Wir lernen Programmieren

- Bei zählergesteuerten Schleifen können viele Dinge beliebig eingestellt werden
  - Zählervariable: Meist i genannt
  - Startwert: Meist ab 0, wenn hochgezählt wird
  - Abbruchbedingung (Endwert): Lesbar als „solange wie“
  - Schrittweite: Meist +1 bzw. -1
- Lässt sich der Endwert nicht erreichen spricht man von einer Endlosschleife
  - Bsp. Start 10, Schrittweite -3, Endwert 15

Wir lernen Programmieren

- Mit der For-Schleife wurde das Problem „wiederhole x-mal“ gelöst, wobei x definiert sein muss (fix oder als Benutzereingabe)
- Häufig gibt es Problemstellungen wie „wiederhole solange bis“, wobei das solange dann unbekannt viel ist
- Dafür gibt es kopfgesteuerten Schleifen (While-Schleifen), die solange laufen, bis eine Bedingung nicht mehr zutrifft

Wir lernen Programmieren

- Solange wie …

```
while (Bedingung) {
  Anweisungen
}
```

Wir lernen Programmieren

- Um bei While-Schleifen keinen endlosen Zyklus zu verursachen muss sich die Variable der Bedingung im Körper ändern können
- Die einfachste Endlos-Schleife wäre dementsprechend die unbedingte Ausführung
  - while(true) {...}

Wir lernen Programmieren

- Schleifen lassen sich ähnlich wie Bedingungen beliebig verschachteln
- Dabei ist darauf zu achten, dass die Schleifenvariablen nicht mehrfach gleich genannt werden

Wir lernen Programmieren

Wir lernen Programmieren

- Bestimmte Vorgänge bzw. Abläufe kommen immer wieder vor
- Schleifen bilden nur direkt aufeinander folgende Wiederholungen ab
- Mit Funktionen können Anweisungen zusammengefasst und wiederverwendbar gemacht werden

Wir lernen Programmieren

- Funktionen sind ähnlich wie Variablen benannt
- Als Eingabewerte nehmen sie Parameter an eine Art Variable
- Funktionen können einen Wert zurückgeben
  - Jedoch nur Werte eines vorher definierten Typs
- Um eine Funktion zu benutzen „ruft“ man sie

Wir lernen Programmieren

- Funktionen die nichts zurückgeben, da sie z.B. direkt eine Ausgabe machen, sind vom Typ void
- Void ist der Datentyp für das geplante Nichts
- Daneben existieren noch die Typen „null“ für das ungeplante Nichts und „undefined“ für undefinierte Fälle
- Im Gegensatz dazu steht der Typ „any“ im positiven Sinne für undefiniert, in ihm können ausnahmslos alle Werte abgelegt werden

Wir lernen Programmieren

Wir lernen Programmieren

- Bisher galt: Eine Variable enthält einen Wert
- Bei mehreren gleichartigen Dingen einer Gruppe führt das zu Problemen
  - Bsp. Schüler einer Klasse, Gezogene Zahlen im Lotto
- Das Array bietet eine Datenstruktur, in der mehrere Werte eines Typs mit nur einem Namen abgelegt werden können
  - Die einzelnen Werte sind dann per Indexnummer zugreifbar

Wir lernen Programmieren

- Arrays können exakt wie normale Variablen verwendet werden
  - Lesender und schreibender Zugriff erfolgt zusätzlich mit der Angabe der Position
  - Das erste Element hat die Position 0, da die Zahl ursprünglich den Abstand zum Beginn des Arrays im Speicher angegeben hat
- Zusätzlich kann ein Array auch en bloc z.B. an eine Funktion übergeben werden
  - Dazu nennt man nur den Namen ohne Index

Wir lernen Programmieren

- Dadurch, dass die Indizes fortlaufend numerisch sind (0, 1, ..., n) harmonieren sie sehr gut mit den bekannten For-Schleifen