Einführung in das Programmieren von Software

Grundlagen und Prinzipien, Theorie und Praxis

Was es ist

- Ein Mensch erklärt einer Maschine, was sie tun soll
- Maschinen verstehen nur präzise Anweisungen in einer Programmiersprache, die sie kennen
- Programmcode ist eine Liste von Anweisungen.
- Die Software verarbeitet Daten.

Wie es funktioniert

- Der Code wird von einem so genannten Parser gelesen und interpretiert.
- Der Parser prüft die Präzision, nicht den Inhalt.
- Der Code kann dann direkt ausgeführt werden(Skript) oder in binäre Maschinensprache übersetzt(compiliert) werden.
- Code besteht ausschließlich aus Text.

Was für Code gibt es?

- Prinzipiell zwei Arten:

  1.: Anweisungen, die Befehle abarbeiten:

  PHP, Java, Rust, C++, JavaScript, Python etc

  2.: Darstellung und strukturierte Speicherung von Daten:

  XML, HTML, SQL, JSON, CSV etc

Was ist eine Umgebung?

- Software läuft auf einem Betriebsystem im Arbeitsspeicher, es muss für Skripte einen Interpreter und für Binäre Software einen Compiler für dieses Betriebsystem geben.
- Software nutzt, was das Betriebsystem bereitstellt:
  - Eingabe/Ausgabe
  - Dateien lesen und schreiben
  - Datenbanken lesen und schreiben
  - Netzwerkverbindungen nutzen

Ist das alles?

- Nicht ganz:
- Die meiste moderne Software benutzt außer dem Betriebsystem auch Softwarebibliotheken, die Routineaufgaben vereinfachen.
  - Text und Grafik anzeigen, Sound/Video ausgeben/aufnehmen
  - Datenbanken verbinden
  - Dateiformate lesen/schreiben

Beispiel 1: die Skriptsoftware PHP

1.) PHP wird durch Aufruf einer Adresse im Webbrowser aus einer Datei gestartet, die unter der Adresse liegt.

2.) Der PHP-Parser überprüft den Code

3.) Das PHP Programm läuft durch bis zu seiner letzten Zeile, im Webbrowser hat es HTML und JavaSkript ausgegeben.

Beispiel 2: ein Shellskript

1.) Eine Datei mit Anweisungen für !#/bin/bash wird vom Nutzer oder zeitgesteuert oder von einem anderen Programm aufgerufen.

2.) Das Skript fragt nach Bedingungen(meist im Dateisystem) und führt Anweisungen aus:

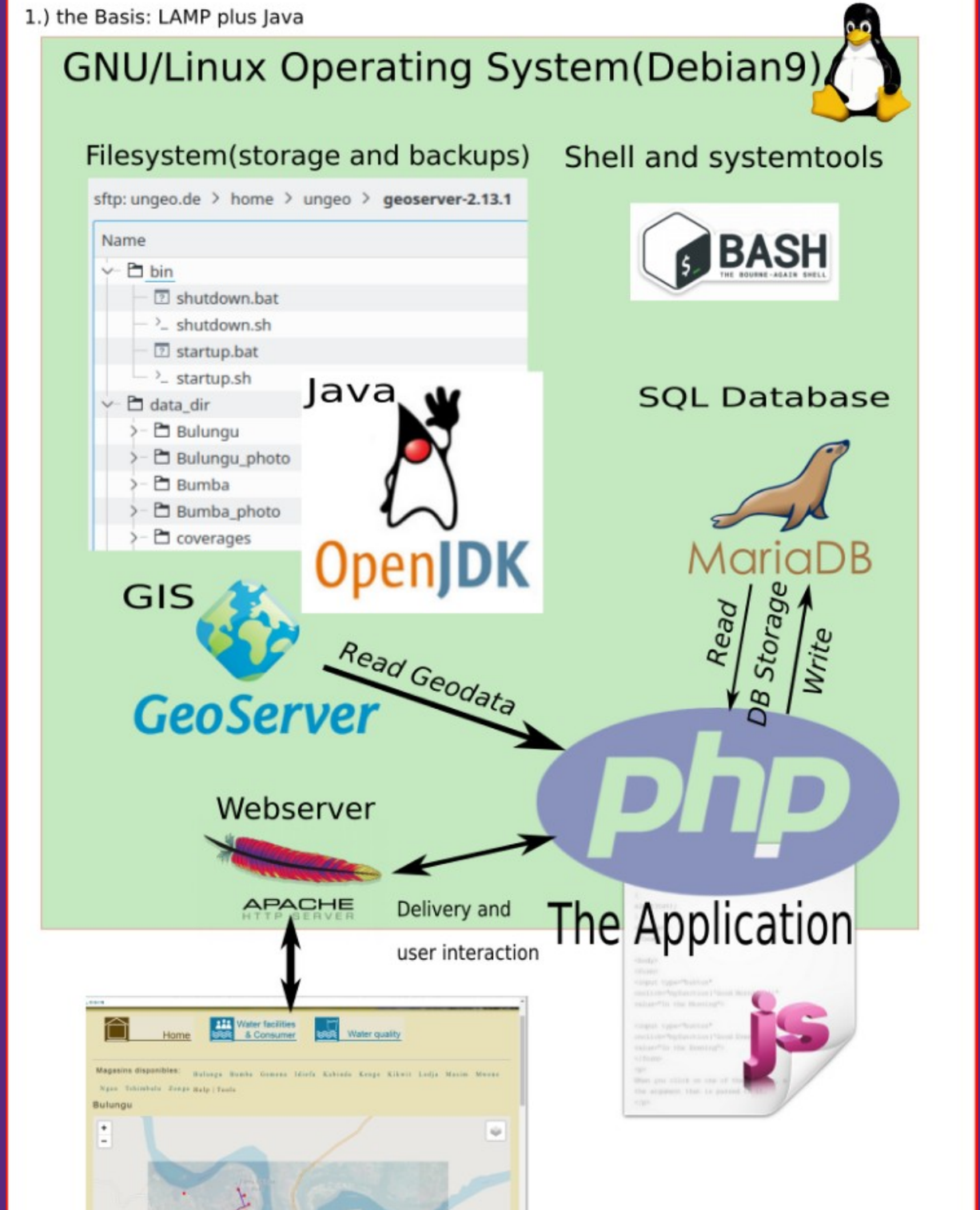
```
if test -d „/home/nutz/ordnername“ #gibts den Ordner überhaupt?
 then
  echo „Ordner existiert“;
 if cp -Rp „var/www/webseite“ „/home/nutz/ordnername“; #Wenn ja, kopiere einen anderen Ordner in diesen
  then
   echo „webseite kopiert“;
  else
   echo „Kopieren von webseite geht iwi nich… sad…“;#Wenn die Kopieranweisung cp „false“ zurückgibt
   exit
 fi #Verzweigungen müssen einen Anfang und ein Ende haben
fi
```

Etwas anspruchsvollere Backupscripte laufen täglich auf Millionen von Computern....

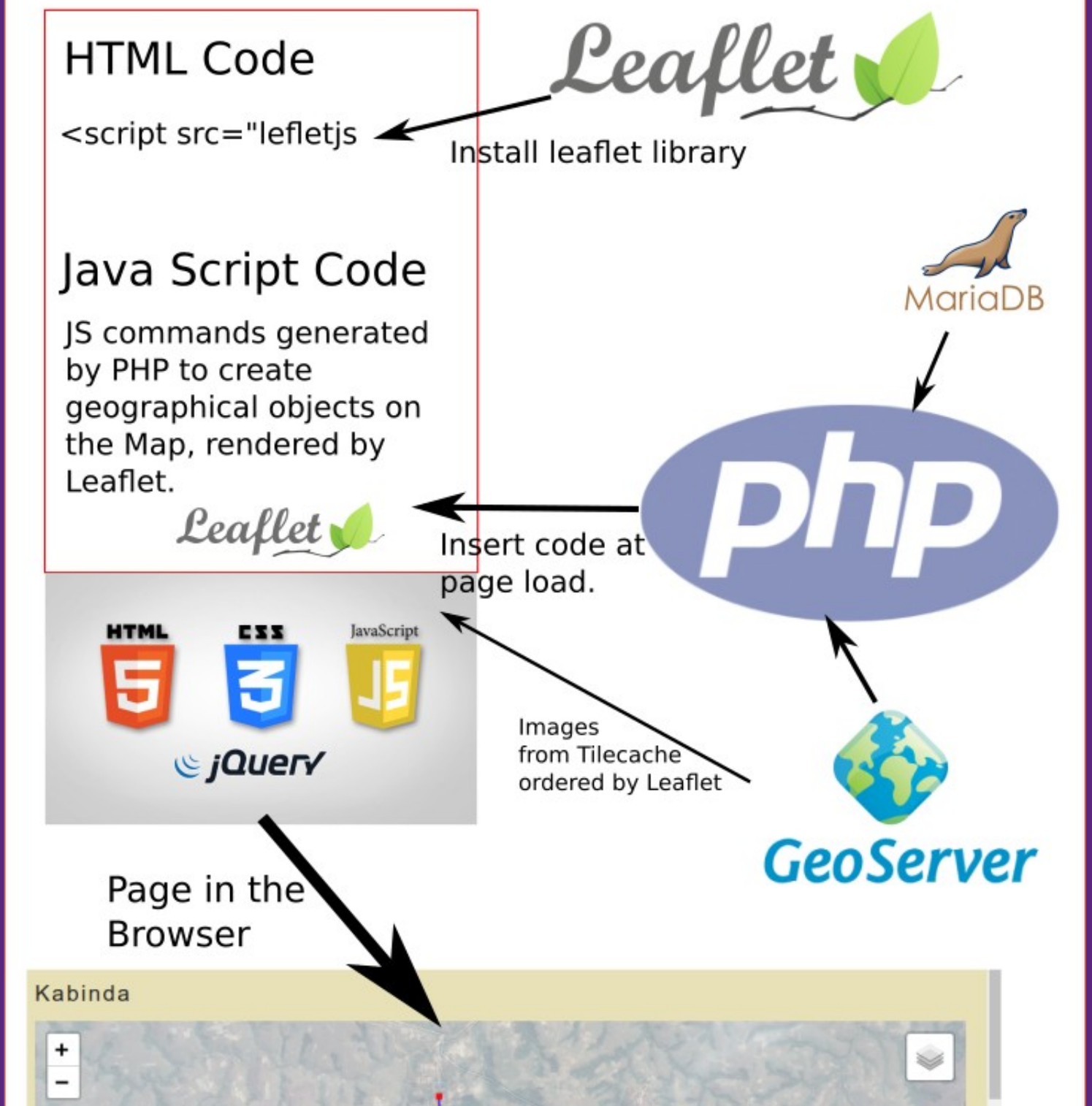
Beispiel 3: ein JavaProgramm

- Javasoftware besteht aus Maschinencode, der compiliert werden muss.
- Der Code wird wie bei Scripten in Text geschrieben.
- Ein Parser prüft die Korektesse
- Der compiler javac wandelt den Code in Maschinen code um
- Das Programm startet in einer JVM fürs Betriebsystem

Ein Beispiel für einen hybriden LAMP/Java Server



Eine komplexe Anwen-dung mit PHP



Welche Software für welchen Zweck ?

- Compilierte Binärsoftware wird vor allem dort gebraucht, wo es auf maximale Leistung und große Datenumsätze ankommt. Auch Bibliotheken sind oft binär.
- Skripte bieten maximale Flexibilität und schnelle Eingriffsmöglichkeiten. Admins müssen sie kennen.
- Beide Arten können alle gängigen Pradigmen umsetzen.

Para Was?

- Ein Paradigma ist ein konzeptioneller Rahmen, in dem technische, politische und wissenschaftliche Aufgaben erledigt werden.
- Die wichtigsten Programmierparadigmen sind:

  - Prozedural: eine Anweisung nach der anderen.

  - Objektorientiert: Anweisungsketten werden in Objekten zusammengefasst.

  - Funktional: Funktionen fassen Anweisungen zusammen und beeinflussen sich gegenseitig.

Welches Paradigma für welchen Zweck?

- Jedes größere Projekt nutzt mehrere Paradigmen gleichzeitig, in den meisten Paradigmen finden sich Elemente anderer Paradigmen.
- Für die Automatisierung von Administration sind nach wie vor prozedurale Skripte in Bash, Perl und Python der Standard.
- Komplexe Anwendungen werden Objektorientiert programmiert.
- Funktionale Paradigmen bilden oft die Grundlage von komplexer Statistik, Steuerung und KI

Werkzeuge

- Ein Editor erlaubt das Aufschreiben von Code und hilft dabei, die korrekte Synthax(Regeln der Sprache) einzuhalten.
- UML Grafiken helfen, Programme zu planen und zu dokumentieren.
- KI hilft, Konzepte und Regeln zu verstehen und liefert brauchbare Beispiele für Lösungen, die allerdings immer geprüft und angepasst werden müssen.

Das wichtigste Werkzeug:

