INP (1)

Prof. Dr.-Ing. S. Gössner

University of Applied Sciences Lippe & Höxter

Inhalt

Kontakt

Stefan Gössner

FH-Tel.: 488
mobil: +49-173-9771260

Email: stefan.goessner@fh-luh.de
Web: http://goessner.net

Ziele

- Erlernen einer Programmiersprache
- Übung der logischen, algorithmischen Denkweise
- Erstellen einfacher bis mittelschwerer Webanwendungen
- Erleichterung des Übergangs auf andere Programmiersprachen

Anforderungen

- Grundlagen der Rechnertechnik
- Umgang mit Betriebssystem
- mathematische Grundkenntnisse
- eine gewisse Portion Neugierde und Experimentierfreude

Programmieren

Programmieren ist …

- die Tätigkeit, Computerprogramme *(Software)* zu erstellen …
- … und den Computer das machen zu lassen, was man möchte.

Softwareentwicklung vs. Programmieren

- Softwareanforderungsanalyse *(software requirements)*
- Softwareentwurf *(software design)*
- **Softwareprogrammierung** *(software construction)*
- Testen von Software *(software testing)*
- Warten von Software *(software maintenance)*
- Konfigurieren von Software *(software configuration management)*
- Organisatorische Aspekte bei der Softwareentwicklung *(software engineering management)*
- Der Vorgang der Erstellung von Software *(software engineering process)*
- Software-Werkzeuge und Verfahrensweisen bei der Softwareentwicklung *(software engineering tools and methods)*
- Softwarequalitätssicherung *(software quality assurance)*

Was ist ein Programm ?

Ein Programm ist …

- eine **Datei**, die installiert ist und zur Erfüllung einer bestimmten Aufgabe beliebig oft ausgeführt werden kann *(Anwendersicht)*.
- ein **Text**, der in einer höheren, formalen Sprache verfasst ist und Instruktionen für den Computer bereitstellt *(Programmierersicht)*.
- eine **Anweisungsfolge** mittels derer ein Rechner die jeweils aktuellen Daten zur Erfüllung einer bestimmten Aufgabe bearbeitet *(abstrakte Sicht)*.

Vom Quelltext zur Programmausführung

1. Der Quelltext *(source code)* als Anweisungsfolge wird in eine Datei geschrieben.
2. Ein *Compiler* oder *Interpreter* übersetzt den Text in ausführbaren *Maschinencode (Objekt-, Binärcode)*.
3. Ein *Binder (Linker)* fügt ggfs. weitere Programmbibliotheken hinzu und rechnet relative in absolute Adressen um.
4. Ein *Lader (loader)* als Dienst der jeweiligen Laufzeitumgebung *(z.B. Betriebssystem)* lädt das Programm in den Hauptspeicher und führt es aus.

Programmiersprachen (1)

- Natürlichen Sprachen sind zur Beschreibung von Computerberechnungen nicht präzise und eindeutig.
- Stattdessen werden künstlich geschaffene Sprachen *(formale Sprachen)* zur Erstellung von Verarbeitungsanweisungen für Rechnersysteme verwendet – sog. **Programmiersprachen**.
- Programmiersprachen basieren auf (menschen)lesbaren Text und …
- … vermitteln dem Computersystem
  - interne Verarbeitungsschritte
  - beteiligte Daten und
  - deren Struktur

Programmiersprachen (2)

```
#include <stdio.h>

int main(void)
{
  printf("Hallo Welt!\n");
  return 0;
}
```

- Die äussere, textuelle Form der Programmiersprache – die verwendeten Symbole *(Zeichen und Wörter)* – bezeichnen wir als **Syntax**.
- Die Bedeutung der Symbole in einer Programmiersprache nennen wir deren **Semantik**.
- Die Intention des Programmierers hinsichtlich der Gesamtheit des Computerprogramms heisst **Pragmatik**.

Programmiersprachen (3)

Einteilung nach …

- Maschinensprache und Assembler
- Höhere Sprachen
  - Prozedurale Programmiersprachen *(Pascal, C, Basic, …)*
  - Objektorientierte Programmiersprachen *(C++, C#, Java, Smalltalk, …)*
  - Skriptsprachen *(Python, Php, Javascript, …)*
  - Auszeichnungssprachen *(Latex, HTML, XML, …)*
  - Funktionale Sprachen *(Lisp, Haskell, …)*
  - Logische Sprachen *(Prolog, …)*

Weitere Einteilungen möglich nach

- Anwendungsgebieten
- Programmierparadigma
- Sprachgenerationen

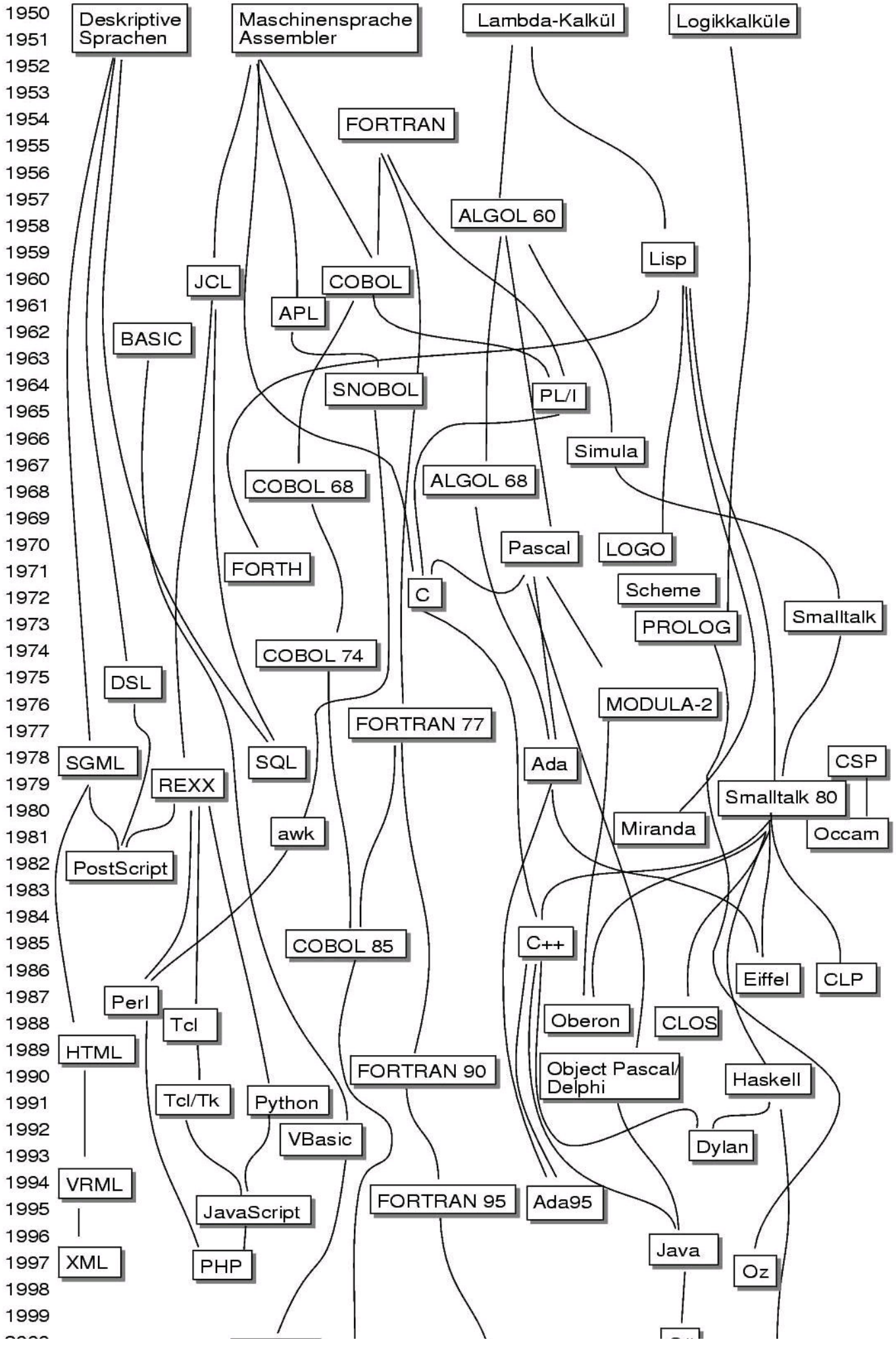
siehe Wikipedia

Programmiersprachen (4)

Unterscheidung gemäss Übersetzungsprozess …

- **Kompilierende** Programmiersprachen *(C, C++, Pascal, …)*
  Der Programmtext wird *"als Ganzes"* in Maschinensprache übersetzt und kann dann ausgeführt werden
- **Halbkompilierende** Programmiersprachen *(Java, C#, …)*
  Der Quelltext wird in einen (prozessorunabhängigen) Zwischencode übersetzt und innerhalb einer Laufzeitumgebung *(JVM, CLR)* interpretierend ausgeführt.
- **Interpretierende** Programmiersprachen *(Javascript, Python, VBScript, …)*
  Der Programmtext wird von einem mitlaufenden Programm *(Interpreter)* schrittweise übersetzt und ausgeführt.

Übersicht der Programmiersprachen


1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
Deskriptive Sprachen
Maschinensprache Assembler
Lambda-Kalkül
Logikkalküle
FORTRAN
ALGOL 60
Lisp
JCL
COBOL
APL
BASIC
SNOBOL
PL/I
Simula
COBOL 68
ALGOL 68
Pascal
LOGO
FORTH
C
Scheme
PROLOG
Smalltalk
COBOL 74
DSL
MODULA-2
FORTRAN 77
SGML
SQL
Ada
CSP
REXX
Smalltalk 80
awk
Miranda
Occam
PostScript
COBOL 85
C++
Eiffel
CLP
Perl
Tcl
Oberon
CLOS
HTML
FORTRAN 90
Object Pascal/ Delphi
Haskell
Tcl/Tk
Python
VBasic
Dylan
VRML
FORTRAN 95
Ada95
JavaScript
Java
XML
PHP
Oz

Elemente einer Programmiersprache

Arbeiten mit Variablen
Zuweisen, Auslesen und Ändern von Speicherinhalten

Elementare Mathematik
zumindest die vier Grundrechenarten

Bedingte Verzweigung
Abhängig von einer Bedingung wird ein anderer Programmfluss ausgewählt

Schleifen
Wiederholen von Programmteilen

Blockbildung
Zusammenfassung mehrerer Befehle

Umgang mit nicht mathematischen Elementen
zum Beispiel mit Text, Bildern, Sound …

Kommentare
Kommentare beeinflussen den Programmablauf nicht. Sie dokumentieren den Quellcode.

Wahl der Programmiersprache

Javascript …

- ist einfach
- ist prozedural
- ist objektorientiert
- wird interpretiert
- hat hohen Nutzwert
- erleichtert den Übergang auf syntaktisch verwandte Sprachen *(C-Sprachfamilie)*

Javascript ...

- ist eine Skriptsprache.
- wurde Mitte der 90er Jahre von Brendan Eich, Fa. Netscape erfunden
- hieß dort zunächst *Livescript* und
- wurde wegen der damaligen Popularität von *Java* in *Javascript* umbenannt
- wurde als *ECMAScript* im Jahre 1999 standardisiert.
- ist objektbasiert, d.h basiert auf Prototypen und nicht auf Klassen.
- gehört der C-Syntax Familie an.
- ist schwach typisiert.
- benötigt eine Laufzeitumgebung, die sowohl den *Interpreter*, als auch problembezogene Objekte bereitstellt *(Browser)*.
- besitzt keine Funktionen für Dateizugriff.
- ist recht leicht zu erlernen.

Javascript - Beispiel

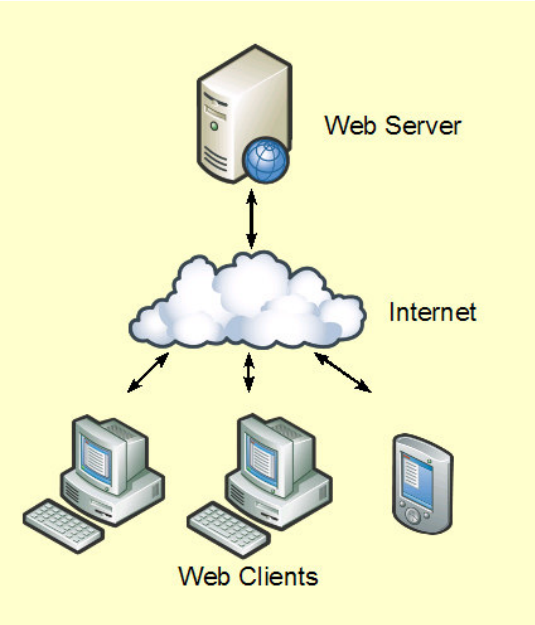
```
var x = [25, 38, 142, 66, 83, 12, 19, 87];
var max = 0;

for (var i=0; i < x.length; i++) {
  if (x[i] > max)
    max = x[i];
}

WScript.Stdout.write("Maximum = " + max);
```

Javascript ausgeführt vom *Windows Scripting Host*.

Browser als Laufzeitumgebung



HTML (1)

HTML ist …

- die **H**yper**t**ext **M**arkup **L**anguage
- seit 1990 das Dokumentformat der Webseiten.
- entworfen worden, um Dokumente untereinander zu verlinken *(Hypertext)*.
- eine Auszeichnungssprache *(Markup)*.
  Beispiel: `<b>fett</b> und <i>schräg</i>`
- reines Textformat und damit hochgradig portabel.
- bewusst einfach gehalten.
- ein Abkömmling der SGML.
- ein Format zur Beschreibung der logischen Dokumentstruktur.
- keine Seitenbeschreibungssprache wie Postscript oder PDF.
- multimedial.
- in seiner gegenwärtigen Version 4.01 nicht mehr zeitgemäss.

HTML (2)

Die Grundstruktur von HTML ist einfach.

- eine Dokumenttypdeklaration (DTD) am Anfang der Dokumentdatei definiert den erlaubten Sprachumfang (Syntax).
- der *head*-Bereich enthält nicht sichtbare Information, Stilangaben, Programmcode und Meta-Angaben.
- der *body*-Bereich beinhaltet die strukturelle Information des sichtbaren Dokumentinhalts.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Titel der Webseite</title>
    <!--Evtl. weitere Kopfinformationen-->
  </head>
  <body>
     Inhalt der Webseite
  </body>
</html>
```

Wie geht's weiter ?

1. HTML Grundlagen
2. Javascript Einführung
3. Datentypen
4. Operatoren, Ausdrücke
5. Anweisungen
6. Arrays, Objekte
7. Funktionen, Methoden
8. HTML Formulare, Events, DOM
9. Fehlerbehandlung
10. Anwendungsbeispiele
11. Repetitorium
12. Klausur

Werkzeuge, Literatur

- Programmeditoren
  - Editplus
    - Download
    - Username: stefan goessner
    - Regcode: 74D1F-84A70-B0017-B835A-F527B
  - Notepad++ *(kostenlos)*
- Laufzeitumgebungen
  - Firefox
  - Internet Explorer
  - Opera
  - Windows Scripting Host
- Dokumentation
  - Jscript *(en)*
  - Javascript Referenz *(en)*
  - SelfHTML *(de)*
- Literatur
  - W. Noack (Hrsg.): JavaScript - Eine Einführung Hannover, 3. Auflage Januar 2004
  - David Flanagan; JavaScript - Das umfassende Referenzwerk, O'Reilly, 2002, ISBN 3-89721-330-3