EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT
FACHBEREICH INFORMATIK
INFORMATIONSDIENSTE

Grundlagen Internet-Technologien

INF3171

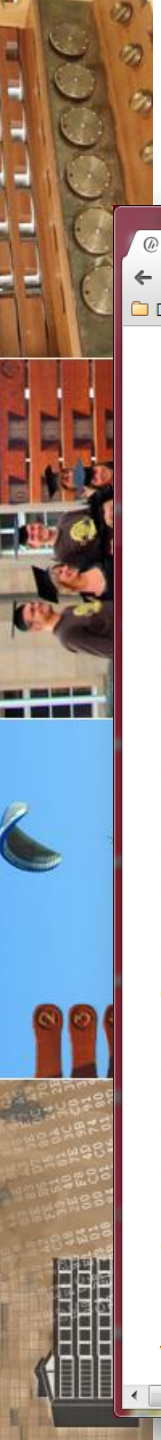
Serverseitige Web-Programmierung mit CGI, Teil I: Einführung in Perl

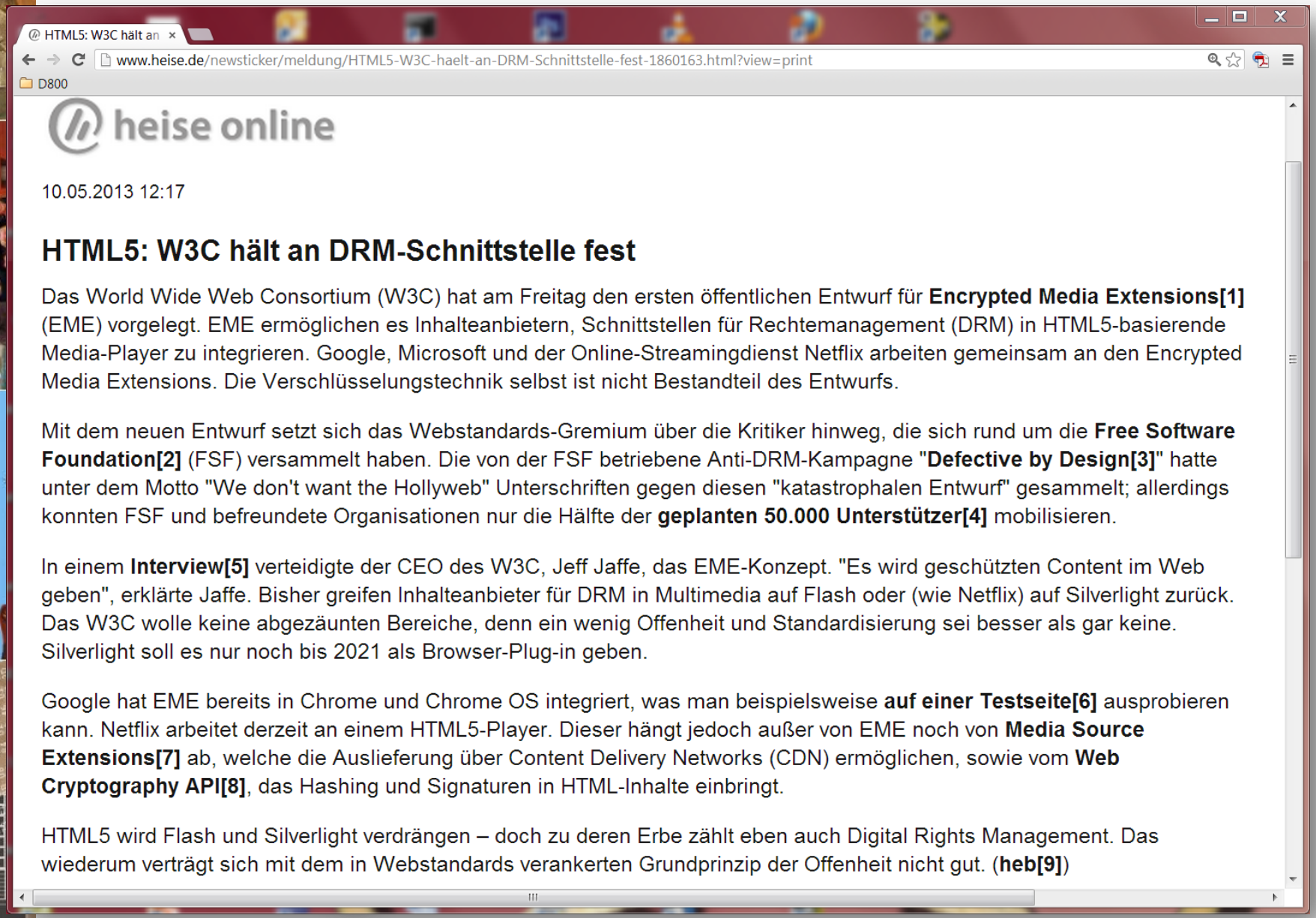
Version 1.0

13.05.2013

aktuelles

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN





HTML5: W3C hält an ×

www.heise.de/newsticker/meldung/HTML5-W3C-haelt-an-DRM-Schnittstelle-fest-1860163.html?view=print

D800

heise online

10.05.2013 12:17

HTML5: W3C hält an DRM-Schnittstelle fest

Das World Wide Web Consortium (W3C) hat am Freitag den ersten öffentlichen Entwurf für **Encrypted Media Extensions[1]** (EME) vorgelegt. EME ermöglichen es Inhalteanbietern, Schnittstellen für Rechtemanagement (DRM) in HTML5-basierende Media-Player zu integrieren. Google, Microsoft und der Online-Streamingdienst Netflix arbeiten gemeinsam an den Encrypted Media Extensions. Die Verschlüsselungstechnik selbst ist nicht Bestandteil des Entwurfs.

Mit dem neuen Entwurf setzt sich das Webstandards-Gremium über die Kritiker hinweg, die sich rund um die **Free Software Foundation[2]** (FSF) versammelt haben. Die von der FSF betriebene Anti-DRM-Kampagne "**Defective by Design[3]**" hatte unter dem Motto "We don't want the Hollyweb" Unterschriften gegen diesen "katastrophalen Entwurf" gesammelt; allerdings konnten FSF und befreundete Organisationen nur die Hälfte der **geplanten 50.000 Unterstützer[4]** mobilisieren.

In einem **Interview[5]** verteidigte der CEO des W3C, Jeff Jaffe, das EME-Konzept. "Es wird geschützten Content im Web geben", erklärte Jaffe. Bisher greifen Inhalteanbieter für DRM in Multimedia auf Flash oder (wie Netflix) auf Silverlight zurück. Das W3C wolle keine abgezäunten Bereiche, denn ein wenig Offenheit und Standardisierung sei besser als gar keine. Silverlight soll es nur noch bis 2021 als Browser-Plug-in geben.

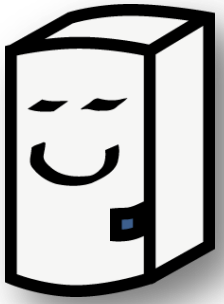
Google hat EME bereits in Chrome und Chrome OS integriert, was man beispielsweise **auf einer Testseite[6]** ausprobieren kann. Netflix arbeitet derzeit an einem HTML5-Player. Dieser hängt jedoch außer von EME noch von **Media Source Extensions[7]** ab, welche die Auslieferung über Content Delivery Networks (CDN) ermöglichen, sowie vom **Web Cryptography API[8]**, das Hashing und Signaturen in HTML-Inhalte einbringt.

HTML5 wird Flash und Silverlight verdrängen – doch zu deren Erbe zählt eben auch Digital Rights Management. Das wiederum verträgt sich mit dem in Webstandards verankerten Grundprinzip der Offenheit nicht gut. (**heb[9]**)

Dynamik im Web

- Dynamik beim Client
  - **JavaScript**
  - Flash
  - Silverlight
  - Java Applets

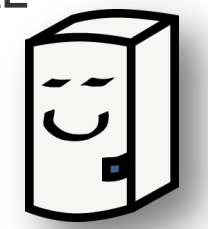
- Dynamik beim Server
  - CGI (mit Perl, C, …)
  - PHP
  - Java Servlets



- Klassiker google, ebay, amazon, facebook, …

Client versus Server

- Client
  - direkte Interaktion
  - keine Netzbelastung
  - CPU des Clients
  - keine DB-Aktion
  - Gefahr für Client
  - verschiedene Clients führen zu verschiedenen Ergebnissen
  - Sourcecode wird ausgeliefert: Kopie
- Server
  - Diensteanbieter hat alles in der Hand
  - Datenbankanbindung
  - zum Client wird nur HTML übertragen
  - Performance: alle teilen sich Server-CPU
  - keine Interaktion



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

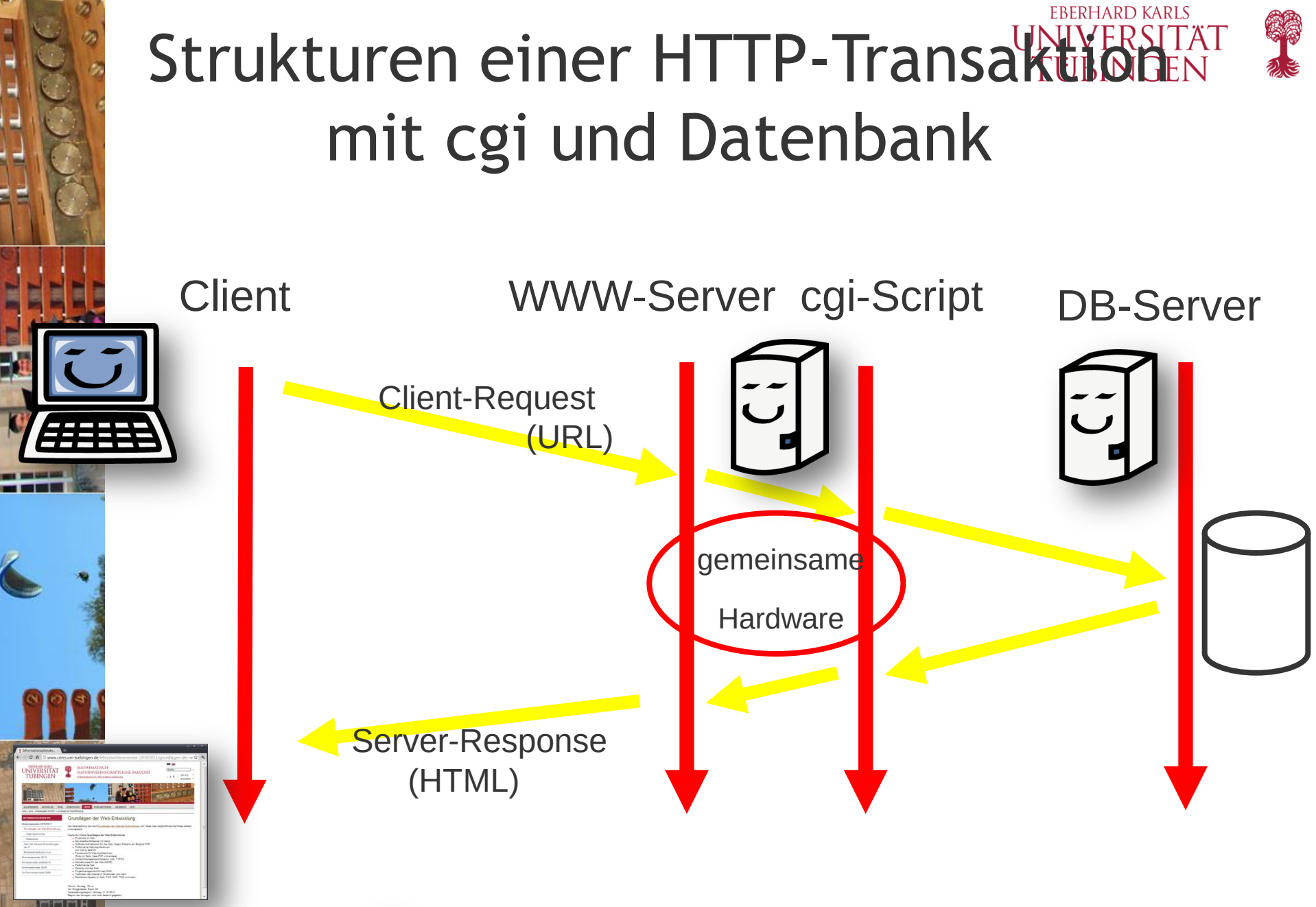
einfach(st)e Servertechnik

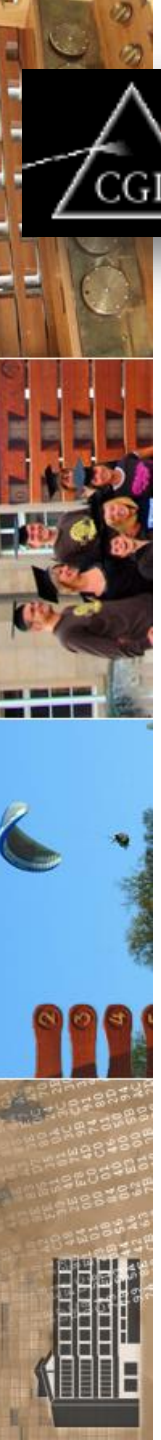
- **Common Gateway Interface** (CGI)

  Möglichkeit, um im WWW serverseitig Programme bereitzustellen, die von "HTML-Seiten gestartet werden" und HTML-Code produzieren

  – Entwicklung ab 1993 (!)

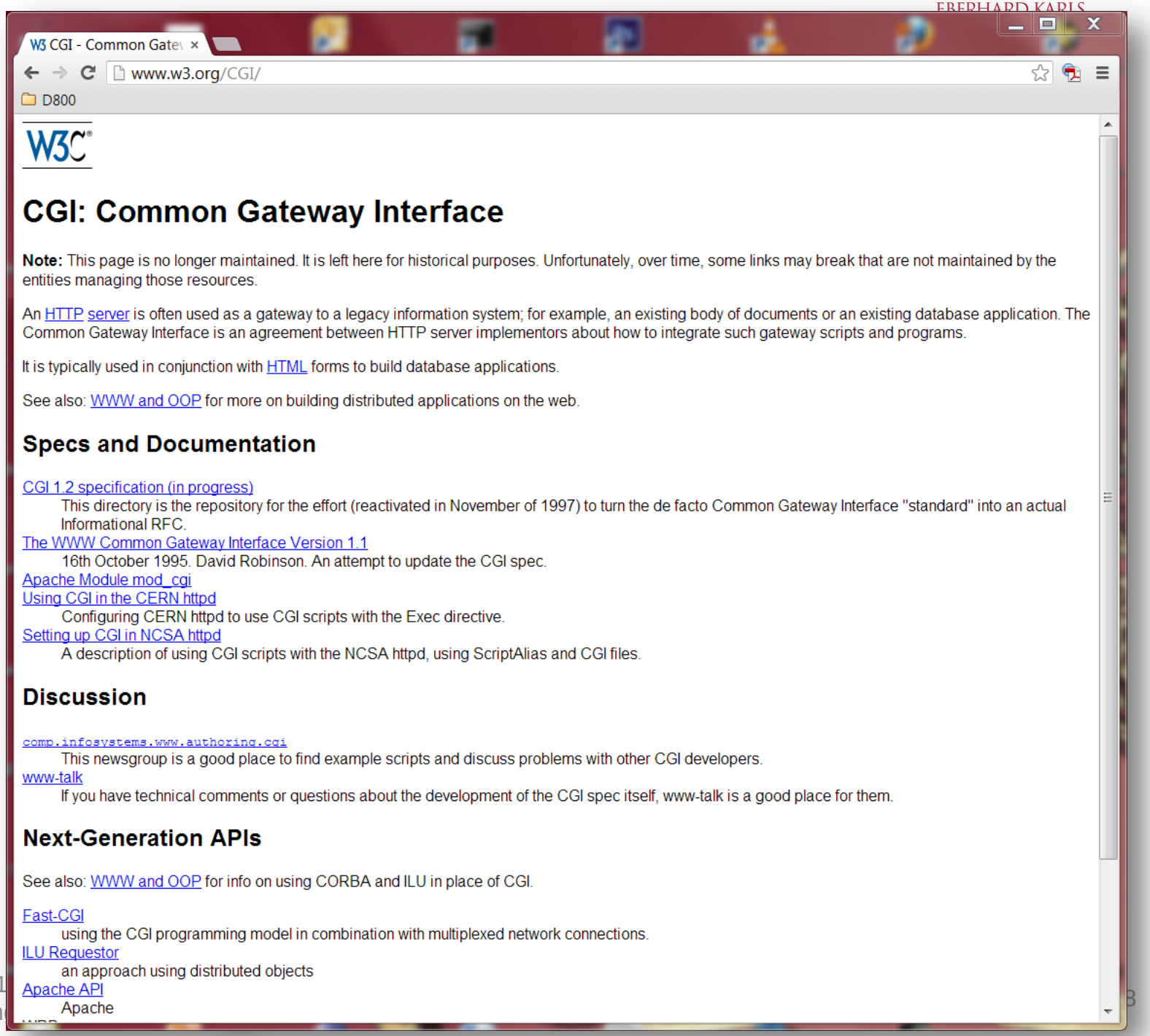
Strukturen einer HTTP-Transaktion mit cgi und Datenbank





cgi: Voraussetzungen

- Webserver ;-)
  - ...der cgi unterstützt (und auf dem cgi erlaubt ist...)
  - Verzeichnis für die Programme (default meist **cgi-bin**)
  - cgi-Umgebunsgvariablen des Webservers
- aktuell cgi Version 1.1 (und das schon lange) http://www.w3.org/CGI/
  - Version 1.2 seit November 1997 in Diskussion



CGI: Common Gateway Interface

Note: This page is no longer maintained. It is left here for historical purposes. Unfortunately, over time, some links may break that are not maintained by the entities managing those resources.

An HTTP server is often used as a gateway to a legacy information system; for example, an existing body of documents or an existing database application. The Common Gateway Interface is an agreement between HTTP server implementors about how to integrate such gateway scripts and programs.

It is typically used in conjunction with HTML forms to build database applications.

See also: WWW and OOP for more on building distributed applications on the web.

Specs and Documentation

CGI 1.2 specification (in progress)
: This directory is the repository for the effort (reactivated in November of 1997) to turn the de facto Common Gateway Interface "standard" into an actual Informational RFC.

The WWW Common Gateway Interface Version 1.1
: 16th October 1995. David Robinson. An attempt to update the CGI spec.

Apache Module mod_cgi

Using CGI in the CERN httpd
: Configuring CERN httpd to use CGI scripts with the Exec directive.

Setting up CGI in NCSA httpd
: A description of using CGI scripts with the NCSA httpd, using ScriptAlias and CGI files.

Discussion

comp.infosystems.www.authoring.cgi
: This newsgroup is a good place to find example scripts and discuss problems with other CGI developers.

www-talk
: If you have technical comments or questions about the development of the CGI spec itself, www-talk is a good place for them.

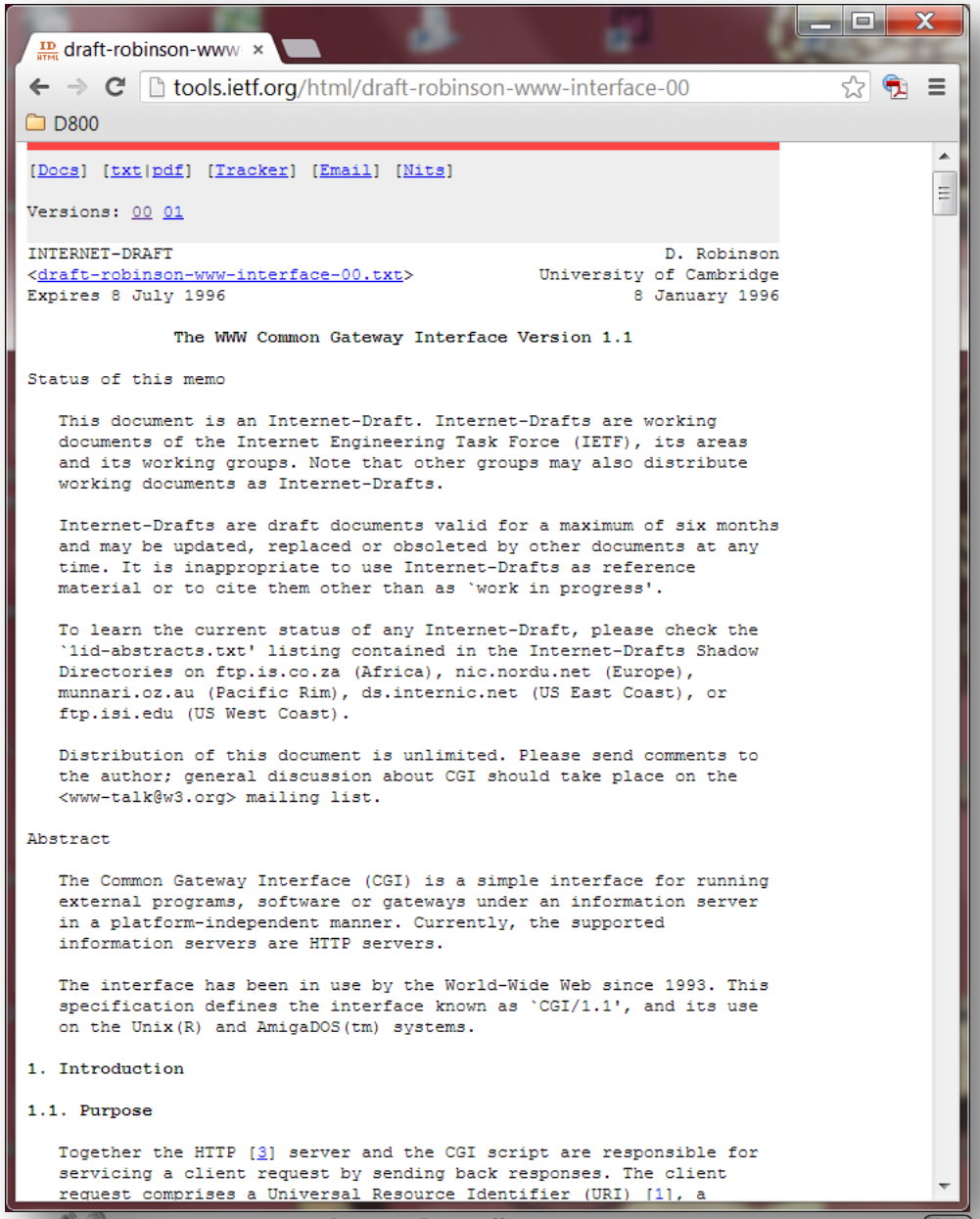
Next-Generation APIs

See also: WWW and OOP for info on using CORBA and ILU in place of CGI.

Fast-CGI
: using the CGI programming model in combination with multiplexed network connections.

ILU Requestor
: an approach using distributed objects

Apache API
: Apache


draft-robinson-www
tools.ietf.org/html/draft-robinson-www-interface-00
D800
[Docs] [txt|pdf] [Tracker] [Email] [Nits]
Versions: 00 01
INTERNET-DRAFT D. Robinson
<draft-robinson-www-interface-00.txt> University of Cambridge
Expires 8 July 1996 8 January 1996
The WWW Common Gateway Interface Version 1.1
Status of this memo
This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as `work in progress'.
To learn the current status of any Internet-Draft, please check the `1id-abstracts.txt' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).
Distribution of this document is unlimited. Please send comments to the author; general discussion about CGI should take place on the <www-talk@w3.org> mailing list.
Abstract
The Common Gateway Interface (CGI) is a simple interface for running external programs, software or gateways under an information server in a platform-independent manner. Currently, the supported information servers are HTTP servers.
The interface has been in use by the World-Wide Web since 1993. This specification defines the interface known as `CGI/1.1', and its use on the Unix(R) and AmigaDOS(tm) systems.
1. Introduction
1.1. Purpose
Together the HTTP [3] server and the CGI script are responsible for servicing a client request by sending back responses. The client request comprises a Universal Resource Identifier (URI) [1], a

serverseitige Voraussetzungen

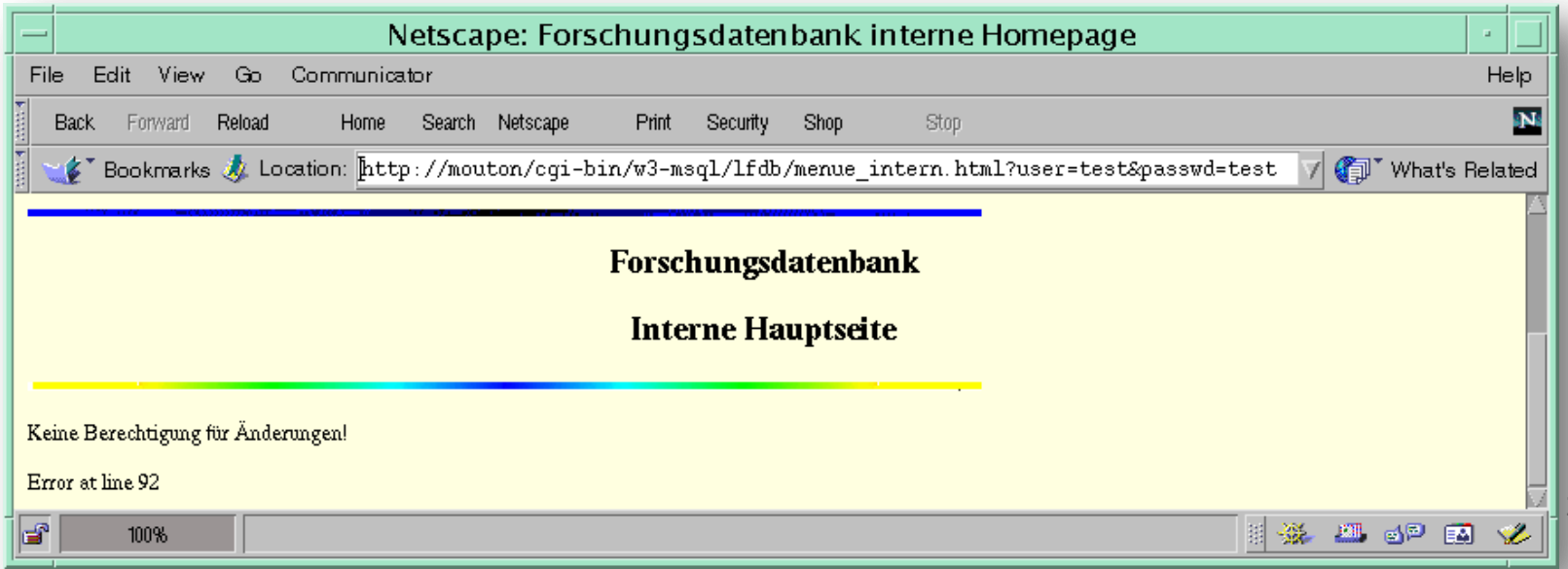
- für den Apache-Webserver: Modul **mod_cgi**
  - wird defaultmäßig integriert
- entsprechende Konfiguration in den Apache-Konfigurationsdateien: **`httpd.conf`**
- Scripte
  - entweder in einem cgi-bin-Verzeichnis (Standard)
    - oder Kennung durch Dateiendung bei entsprechender Konfiguration des Apache

cgi: Aufruf aus HTML

- normaler Hyperlink
  - `<A HREF=/cgi-bin/counter.pl>Zähler</A>`
- über Formulare (➔ Übungen)
- Grafikreferenzen (img, src)
- **SSI**: Server-Side-Includes
  - SSI ist eine einfache und effiziente Form für aktive Websites → später

cgi-Datenübergabe an den Server

- *unterschiedliches* Vorgehen bei den HTTP-Methoden **`GET`** und **`POST`**
  - GET: Codierung der Eingabefelder in URL; Syntax:
    - **`&`** (&) trennt Feldnamen
    - **`=`** trennt Name von Wert
    - **`?`** leitet Datenteil ein



cgi-Datenübergabe an den Server

- Nachteile bei GET:
  - URLs beliebiger Länge???
  - Codierung der »Sonderzeichen«
  - Speichern in Browser-History (➔ Kennwörter?!)
- **POST**: Daten werden über separate I/O-Ströme weitergeleitet, nicht über URL

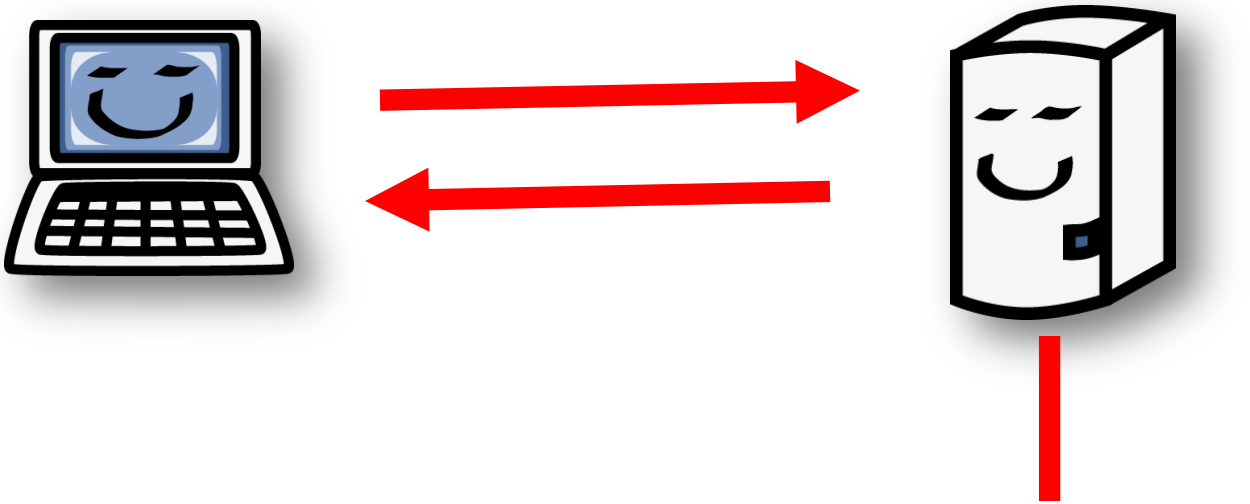
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

cgi Lifecycle

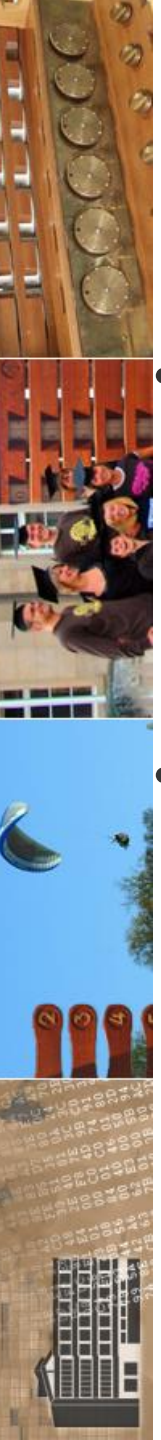
- ein cgi-Aufruf hat folgenden Lifecycle:
  - mit dem http-Request wird ein cgi-Prozeß gestartet, der danach wieder beendet wird
- als Konsequenz ist cgi für viele Anfragen nicht performant, weshalb (viele) Alternativen entwickelt wurden
  - fastCGI, Servlets, Webserver-Module, ...

cgi – das Prinzip

- Web-Client „startet“ cgi-Programm auf Server



CGI-Programm



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

cgi: Umgebungsvariablen

- Server legt bei Aufruf des cgi-Programms spezielle **Umgebungsvariablen** fest, die dem cgi-Programm die notwendigen Informationen liefern
(interner Mechanismus der Datenübergabe)

- einige dieser Umgebungsvariablen:
  - **`SERVER_NAME, SERVER_PROTOCOL, SERVER_PORT`**
  - **`PATH_INFO, SCRIPT_NAME, QUERY_NAME`**
  - **`REMOTE_HOST, REMOTE_ADDR, REMOTE_USER`**

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Standardscipt

- mit Apache wird das Script

  `printenv.pl`

  ausgeliefert

- gibt Überblick über alle Umgebungsvariablen des Servers

134.2.2.38/~zrvwa01/ ×

134.2.2.38/~zrvwa01/cgi-bin/printenv.pl

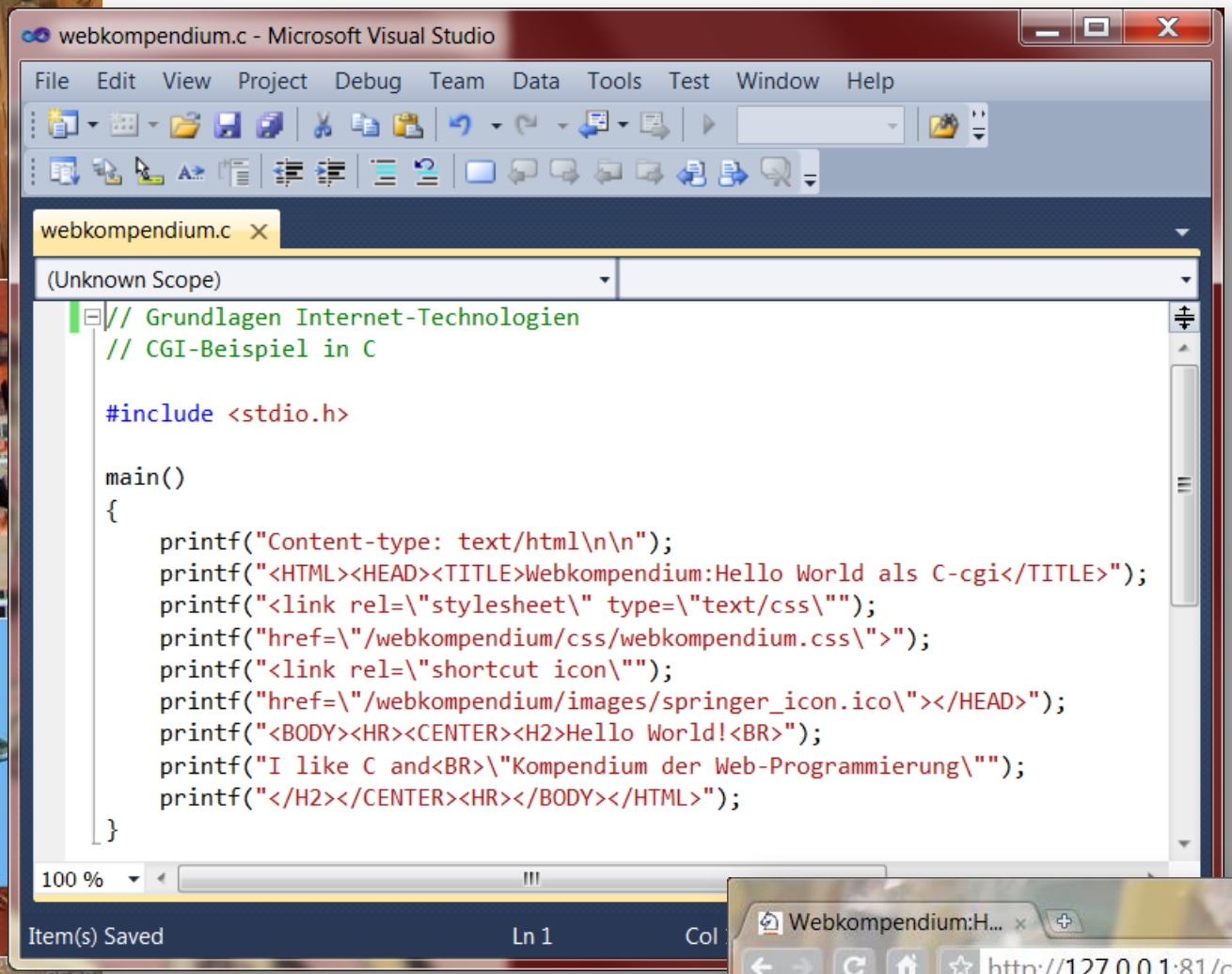
D800

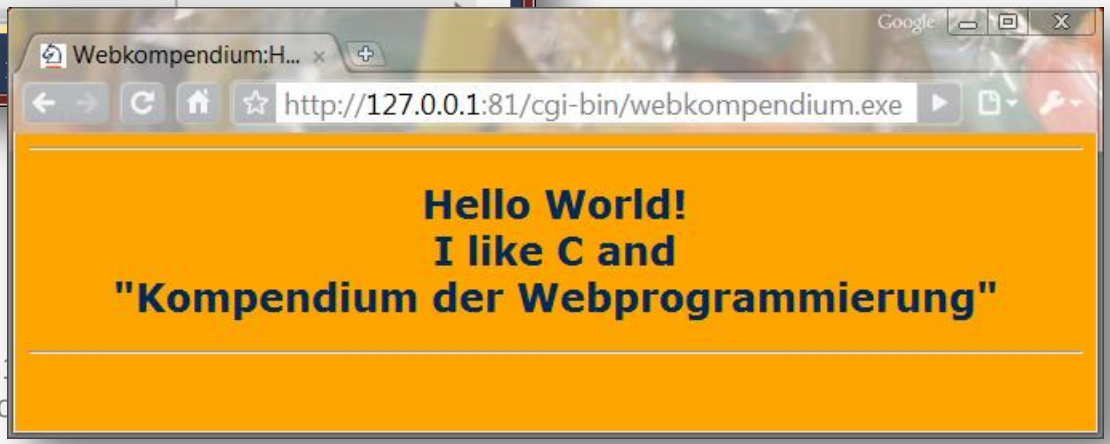
```
DOCUMENT_ROOT="/var/www"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.3"
HTTP_ACCEPT_ENCODING="gzip,deflate,sdch"
HTTP_ACCEPT_LANGUAGE="de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="134.2.2.38"
HTTP_USER_AGENT="Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.31
(KHTML, like Gecko) Chrome/26.0.1410.64 Safari/537.31"
PATH="/usr/local/bin:/usr/bin:/bin"
QUERY_STRING=""
REMOTE_ADDR="134.2.219.193"
REMOTE_PORT="51289"
REQUEST_METHOD="GET"
REQUEST_URI="/~zrvwa01/cgi-bin/printenv.pl"
SCRIPT_FILENAME="/home-link/zrvwa01/public_html/cgi-bin/printenv.pl"
SCRIPT_NAME="/~zrvwa01/cgi-bin/printenv.pl"
SERVER_ADDR="134.2.2.38"
SERVER_ADMIN="webmaster@localhost"
SERVER_NAME="134.2.2.38"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SIGNATURE="<address>Apache/2.2.22 (Debian) Server at 134.2.2.38 Port
80</address>\n"
SERVER_SOFTWARE="Apache/2.2.22 (Debian)"
```


EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Beispiel für CGI mit C

- mit C bekommen wir eine **direkt ausführbare Binärdatei**
- Beispiel **webkompendium.c** gibt nur HTML-formatierte Ausgabe aus





EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

cgi Essentials

- cgi-Programm muss **ausführbar** sein
- liegt im Unterordner cgi-bin
- Ausgabe beginnt mit:

```
Content-type: text/html
\n
```

- Zugriff: URL beginnt mit **cgi-bin**

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Sprachen für CGI

- alles „ausführbare“
- alle compilierten Sprachen (C, C++, C#, …)
- beliebt: Scriptsprachen wie Perl, Python, Ruby
  - Java ist **nicht** möglich – warum?

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

die Scriptsprache Perl

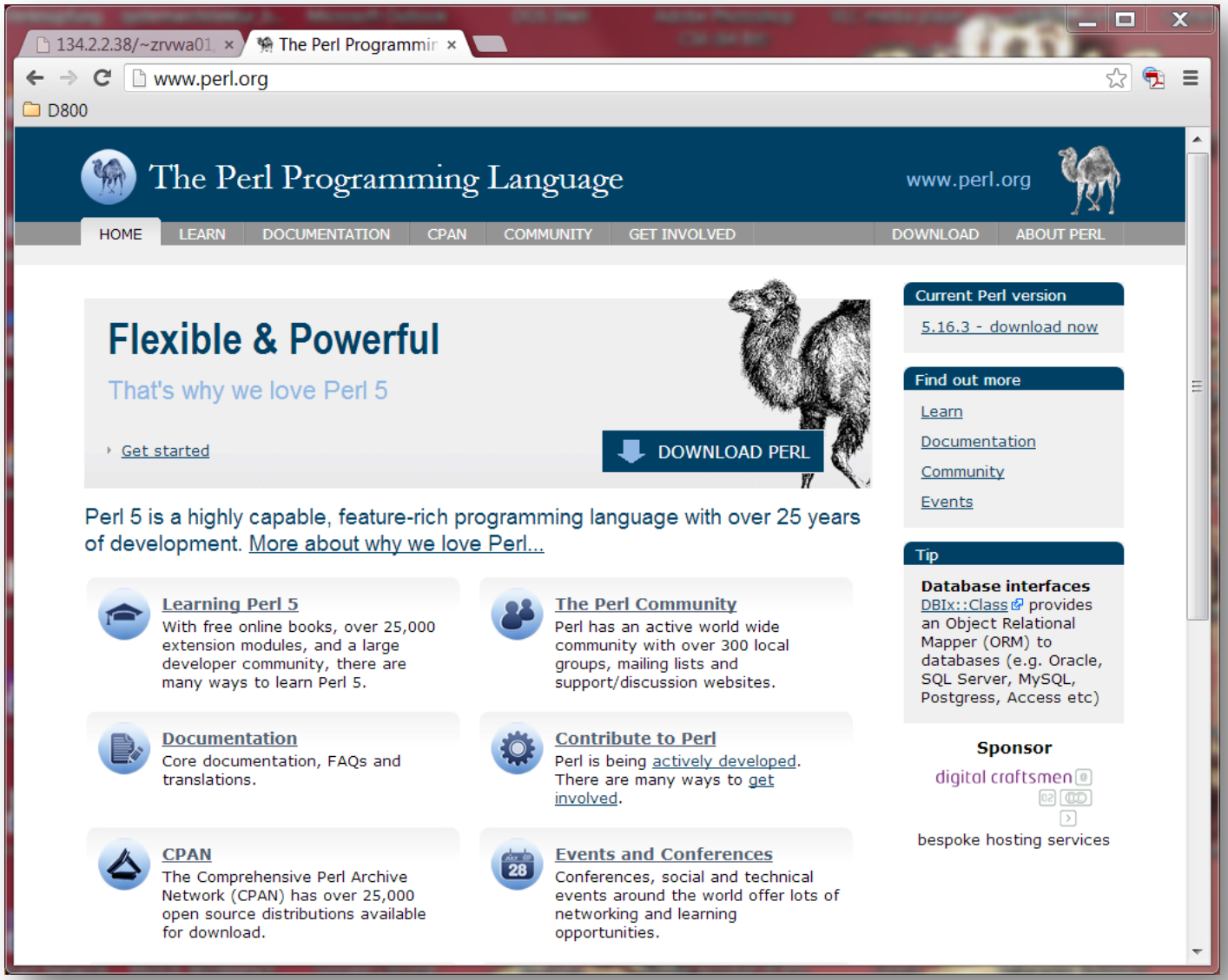
- **PERL: Practical Extraction and Report Language**
  (Praktische Sprache für Datenextraktion und -ausgabe)
  oder:
  Pathologically Electic Rubbish Lister ;-)
  (krankhaft zusammengeschustertes Auflistungsprogramm für wirres Zeug)

- »Erfinder« Larry Wall, 1987

- aktuelle Version: 5.16.3
  (verwenden Sie mindestens 5.004)
  - Dev. PERL 6

Ressourcen für Perl

- offizielle PERL-Seite (Larry Wall):
  `http://www.perl.org`
- PERL-Seite von O'Reilly:
  `http://www.perl.com`
  von dort zahlreiche links, auch Downloads
- Netzwerk mit PERL-Ressourcen:
  CPAN - Comprehensive Perl Archive Network
  `http://www.perl.com/cpan`
- **ActiveState-PERL: `www.activestate.com`**
  - dort die Active-PERL-Distribution
- ...und viele mehr...


134.2.2.38/~zrvwa01
The Perl Programmir
www.perl.org
D800
The Perl Programming Language
www.perl.org
HOME
LEARN
DOCUMENTATION
CPAN
COMMUNITY
GET INVOLVED
DOWNLOAD
ABOUT PERL
Flexible & Powerful
That's why we love Perl 5
Get started
DOWNLOAD PERL
Perl 5 is a highly capable, feature-rich programming language with over 25 years of development. More about why we love Perl...
Learning Perl 5
With free online books, over 25,000 extension modules, and a large developer community, there are many ways to learn Perl 5.
The Perl Community
Perl has an active world wide community with over 300 local groups, mailing lists and support/discussion websites.
Documentation
Core documentation, FAQs and translations.
Contribute to Perl
Perl is being actively developed. There are many ways to get involved.
CPAN
The Comprehensive Perl Archive Network (CPAN) has over 25,000 open source distributions available for download.
Events and Conferences
Conferences, social and technical events around the world offer lots of networking and learning opportunities.
Current Perl version
5.16.3 - download now
Find out more
Learn
Documentation
Community
Events
Tip
Database interfaces
DBIx::Class provides an Object Relational Mapper (ORM) to databases (e.g. Oracle, SQL Server, MySQL, Postgress, Access etc)
Sponsor
digital craftsmen
bespoke hosting services

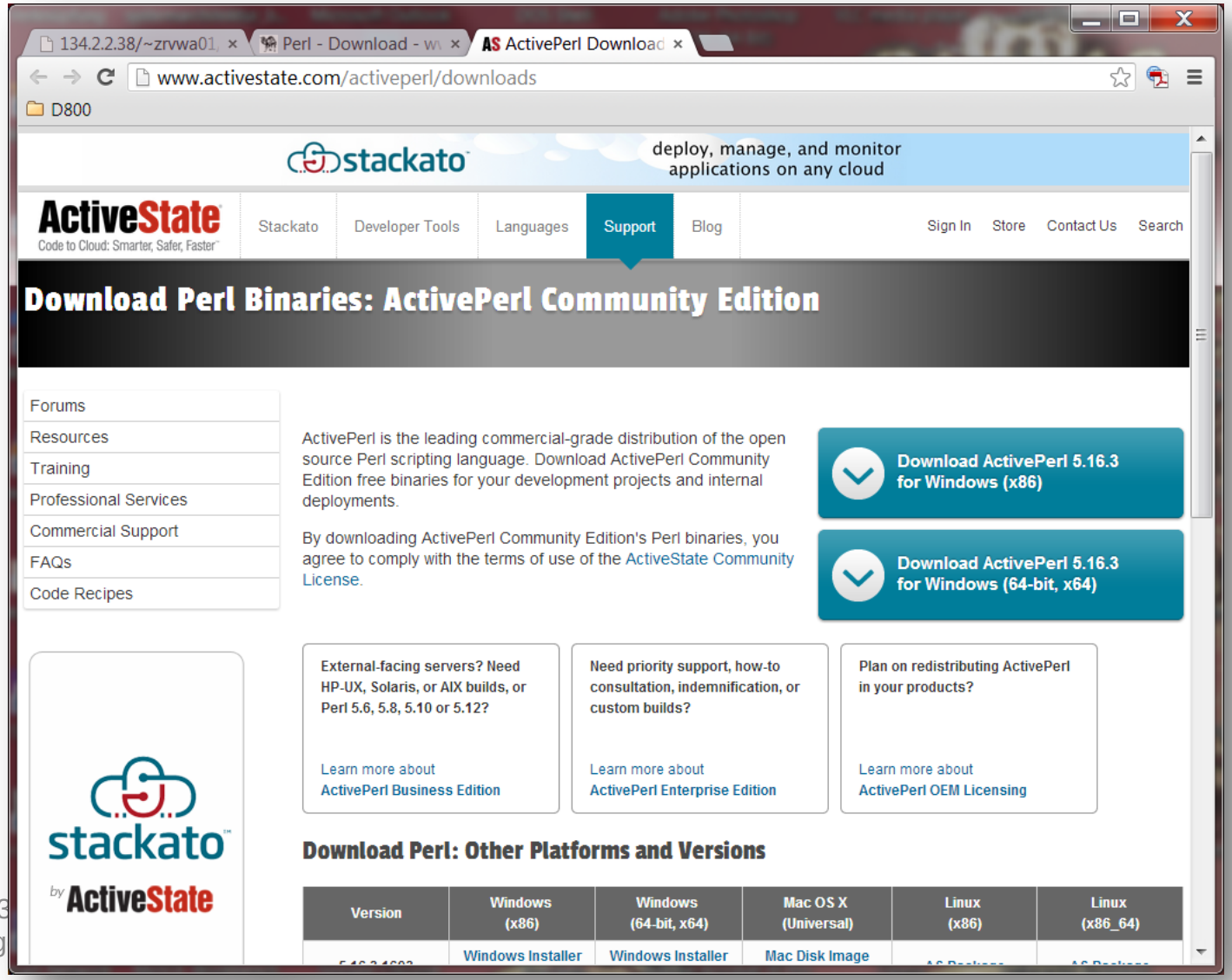
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Literatur zu Perl

- Larry Wall, Tom Christiansen, Randal Schwartz: Programming Perl, O'Reilly
- Randal Schwartz, Tom Christiansen: Einführung in PERL, O'Reilly, 1.Auflage 1998
- Tom Christiansen, Nathan Torkington: PERL Cookbook, O'Reilly

- RRZN: PERL – Eine Einführung (2. Auflage)

Windows-Distribution



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Perl in dieser Veranstaltung

- wir setzen uns in zwei Stufen mit Perl auseinander:
  - Grundlegende Syntax der Sprache
  - Perl im Web
    - cgi-Programme mit Perl
    - Perl-Module
    - auch Web-Clients mit Perl
  - wir behandeln die Datenbankeinbindung nicht mit Perl – sondern mit PHP

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

der Perl-Interpreter

- Perl-Interpreter zum Ausführen des Scriptes
- einige nützliche Flags für den Interpreter:
  - -v : Version des PERL-Interpreters
  - -V : vollständige Information über PERL-System
  - -w : Ausgabe von »warnings« (-W : alle warnings)
  - -help : Übersicht aller flags
  - -c : nur Syntax-Check
  - -e : Übergabe einer PERL-Anweisung als Parameter

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

134.2.2.38 - ID Übung - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
zrvwa01@infodienste:~$ perl -v

This is perl 5, version 14, subversion 2 (v5.14.2) built for x86_64-linux-g
nu-thread-multi
(with 61 registered patches, see perl -V for more detail)

Copyright 1987-2011, Larry Wall

Perl may be copied only under the terms of either the Artistic License or t
he
GNU General Public Licer

Complete documentation f
this system using "man p
Internet, point your bro

zrvwa01@infodienste:~$
```

Connected to 134.2.2.38

DOS Shell

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\users\thomas\documents>perl -v

This is perl 5, version 14, subversion 2 (v5.14.2) built for MSWin32-x64-multi-t
hread
(with 1 registered patch, see perl -V for more detail)

Copyright 1987-2011, Larry Wall

Binary build 1402 [295342] provided by ActiveState http://www.ActiveState.com
Built Oct  7 2011 15:19:36

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl".  If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.


C:\users\thomas\documents>
```

perldoc & Editoren

- perldoc: integrierte online-Hilfe wie man-pages
  Beispiel: **`perldoc -f print`**

- Editoren
  - beliebig
  - Notepad++
  - **Eclipse mit EPIC-Plugin**

www.epic-ide.org/

DOS Shell - perldoc -f print

```
    print FILEHANDLE LIST
    print LIST
    print   Prints a string or a list of strings. Returns true if
            successful. FILEHANDLE may be a scalar variable name, in which
            case the variable contains the name of or a reference to the
            filehandle, thus introducing one level of indirection. (NOTE: If
            FILEHANDLE is a variable and the next token is a term, it may be
            misinterpreted as an operator unless you interpose a "+" or put
            parentheses around the arguments.) If FILEHANDLE is omitted,
            prints by default to standard output (or to the last selected
            output channel--see "select"). If LIST is also omitted, prints
            $_ to the currently selected output channel. To set the default
            output channel to something other than STDOUT use the select
            operation. The current value of $, (if any) is printed between
            each LIST item. The current value of $\ (if any) is printed
            after the entire LIST has been printed. Because print takes a
            LIST, anything in the LIST is evaluated in list context, and any
            subroutine that you call will have one or more of its
            expressions evaluated in list context. Also be careful not to
            follow the print keyword with a left parenthesis unless you want
            the corresponding right parenthesis to terminate the arguments
            to the print--interpose a "+" or put parentheses around all the
            arguments.

            Note that if you're storing FILEHANDLEs in an array, or if
            you're using any other expression more complex than a scalar
            variable to retrieve it, you will have to use a block returning
-- Fortsetzung  --
```

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

das erste Beispiel

- ...das übliche Beispiel

  Die Datei **hello.pl**:

  - ```
    # unser erstes PERL-Script
    #TODO Erklaerung einer skalaren Variablen

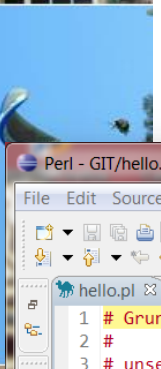
    $phrase =
    "\n Hello WebKompendium!\n I like PERL!\n";

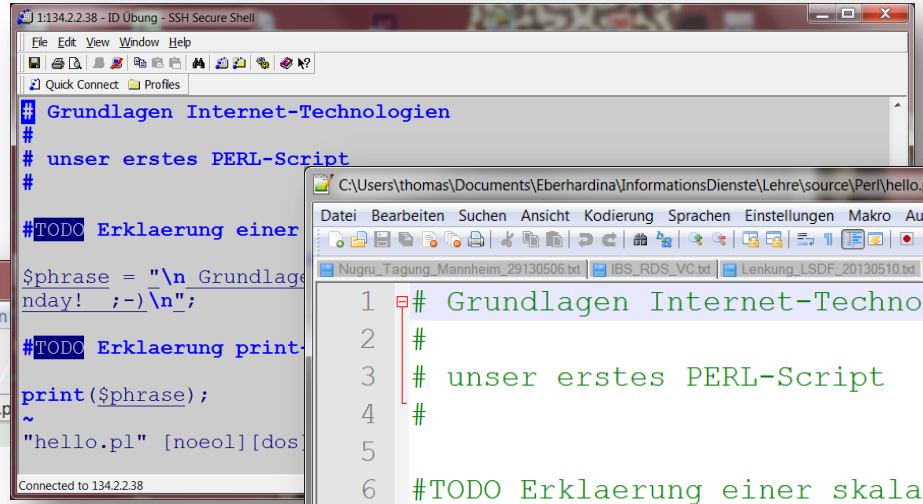
    #TODO Erklaerung print-Anweisung
    print($phrase);
    ```
    ```

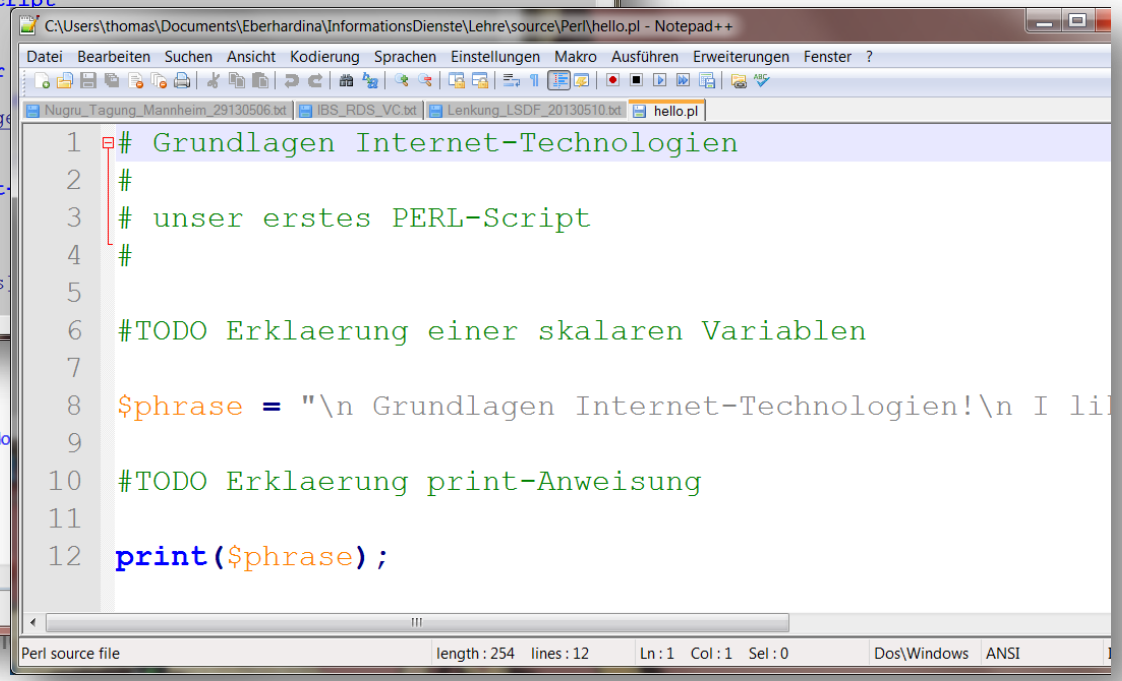
Editoren und Ausführen

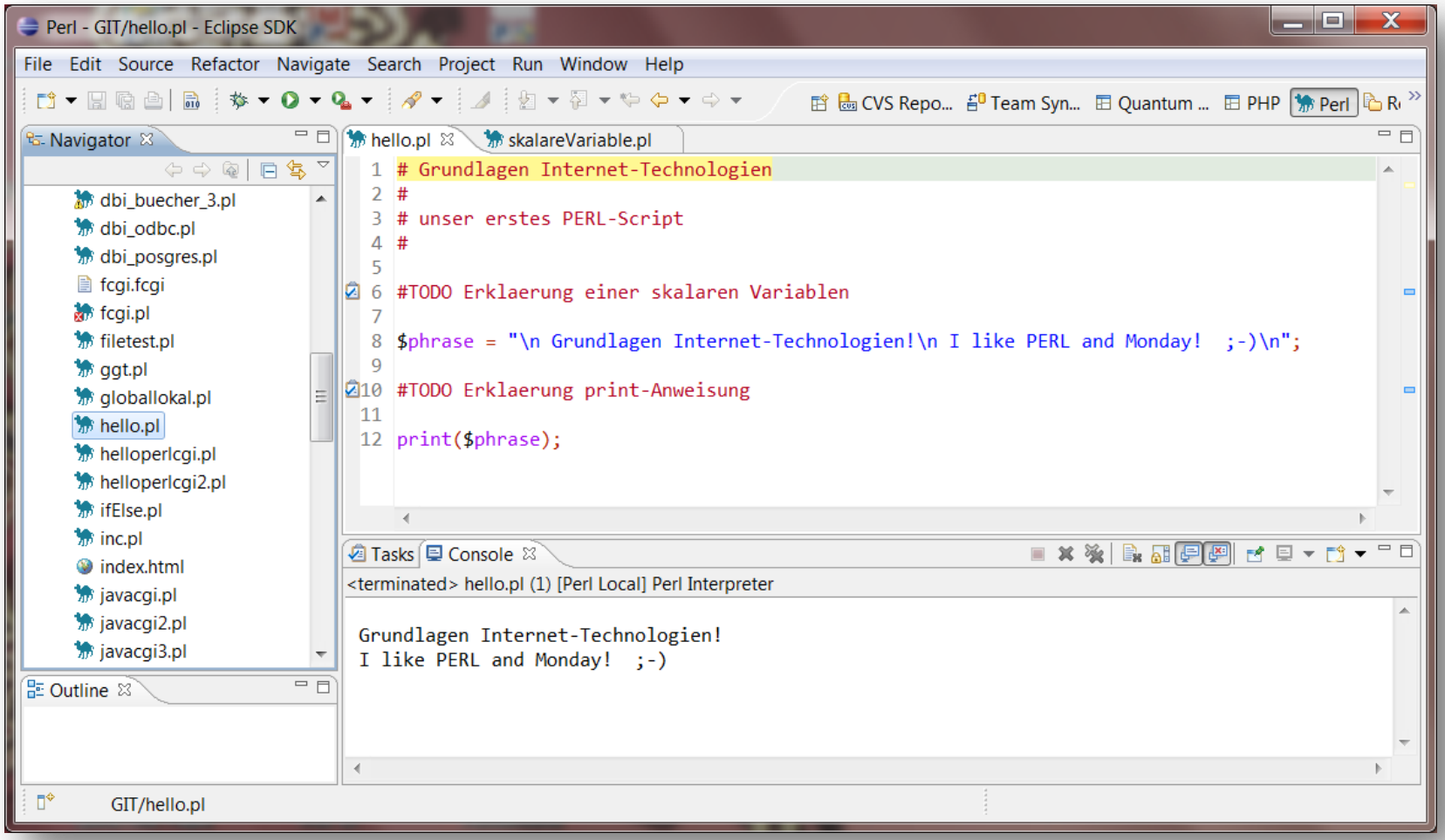
- Notepad++/VI oder Eclipse mit EPIC
  - http://www.epic-ide.org/










Perl - GIT/hello.pl - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
CVS Repo... Team Syn... Quantum ... PHP Perl
Navigator
dbi_buecher_3.pl
dbi_odbc.pl
dbi_posgres.pl
fcgi.fcgi
fcgi.pl
filetest.pl
ggt.pl
globallokal.pl
hello.pl
helloperlcgi.pl
helloperlcgi2.pl
ifElse.pl
inc.pl
index.html
javacgi.pl
javacgi2.pl
javacgi3.pl
Outline
hello.pl
skalareVariable.pl
1 # Grundlagen Internet-Technologien
2 #
3 # unser erstes PERL-Script
4 #
5
6 #TODO Erklaerung einer skalaren Variablen
7
8 $phrase = "\n Grundlagen Internet-Technologien!\n I like PERL and Monday! ;-)\n";
9
10 #TODO Erklaerung print-Anweisung
11
12 print($phrase);
Tasks
Console
<terminated> hello.pl (1) [Perl Local] Perl Interpreter
Grundlagen Internet-Technologien!
I like PERL and Monday! ;-)
GIT/hello.pl

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

das Beispielprogramm

- # Kommentare sind durch # gekennzeichnet (->wie Unix-Shellscripte!)
- ; schließt einfache Anweisung ab
- Skalarvariable **`$phrase`**
- »übliches« newline-Zeichen `\n`
- String-Konstante **`"Hello World!\n"`**

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

more Perl

- skalare Variablen:
  - `$value  = 42;  # integer`
  - `$pi  = 3.1415926;  # float`
  - `$c  = 3.0e08;  # float`
  - `$phrase  = "perl";  # string`
  - `$truth  = "I like $phrase";  # mit Ersetzung`
  - `$costs  = 'perl costs $0';  # ohne Ersetzung`
- skalare: »einfachster« Datentyp in Perl, entweder Zeichenkette oder Zahl (vgl. Variablen in JavaScript)

Regeln für Skalare

- Perl ist wie JavaScript *nicht* typisiert
- Skalare werden *kontextabhängig* als String, »Zahl« oder boolsche Größe interpretiert
- falls möglich erfolgt ein *automatischer Cast*:

```
$value = '123';          # String
print $value + 1;
```

  → Ausgabe: 124

- automatische Initialisierung
  - neue Variablen werden mit 0 oder "" initialisiert

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Zeichenketten

- Sonderzeichen (", ', \) werden durch vorangestellten Backslash codiert:
  - `'tu\'s nicht'`
- übliche Ersetzung, z.B.:
  - \n : newline
  - \t : Tabulator
  - \x7f : Hexadezimalwert 7f
  - \a : Tonsignal
- Strings werden mit . addiert
  - `$string = "Hello " . "World!";`

Perl - GIT/skalareVariable.pl - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

CVS Repo... Team Syn... Quantum ... PHP Perl R

Navigator

- skalareVariable.pl
- stringliterale.pl
- student.pl
- subroutine.pl
- substitutionRegex.pl
- temp.pl
- Uebung11.pl
- uebung20.pl
- uebung21a.pl
- uebung21b.pl
- uebung23a.pl
- uebung23b.pl
- Unicode.pl
- urlaub.pl
- WebKompendium.pm
- wlan.pl
- zaehler.pl
- zeit.pl
- IP Perl 20091
- IP PHP 20091
- PetitMouton

Outline

hello.pl | skalareVariable.pl

```
1  # Grundlagen Internet-Technologien
2  #
3  # skalare Variablen in PERL
4
5  $variable = "42";
6
7  print  'Die Variable $variable hat die Laenge '.length($variable)."\n";
8
9  $variable += 1;
10
11 print 'Der neue Wert von $variable ist '.$variable."\n";
12
13 $variable *= 3.1415926;
14
15 print '...und nun ist $variable ein Gleitkommawert: '.$variable. "\n";
16
17 print 'Die Variable $variable hat nun die Laenge '.length($variable)."\n"
```

Tasks | Console

<terminated> skalareVariable.pl [Perl Local] Perl Interpreter

```
Die Variable $variable hat die Laenge 2
Der neue Wert von $variable ist 43
...und nun ist $variable ein Gleitkommawert: 135.0884818
Die Variable $variable hat nun die Laenge 11
```

Writable | Insert | 1 : 1

Standardeingabe

- eine Eingabemöglichkeit:

```
$a = <STDIN>
```

liest einen Text von der Standardeingabe ein, bis ein Newline eingegeben wird
(**`chomp($a)`** entfernt abschließendes newline, welches mit übergeben wird, **`chop($a)`** entfernt stets letztes Zeichen)

  - werden wir bei der Behandlung von **Streams** genauer verstehen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

weiteres zu Skalaren

- bei numerischen Variablen: führende 0 bedeutet Octaldarstellung, führende 0x hexadezimal
- Operatoren (wie in Java, JavaScript, ...)
  - `$x = $x +1;  <=>  $x += 1;  <=>  ++$x;  <=>  $x++;`
  - für Strings: `$s = $s . $t;  <=>  $s .= $t`
  - spezielle arithmetische Operationen:
    - **$x % $y** : x mod y
    - **$x ** $y** : x^y
  - $s ohne letztes Zeichen: chop($s);
  - $s ohne möglicherweise abschließendes \n: chomp($s);

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

zentrale Sonderzeichen in Perl

Typ	Zeichen	Beispiel	Beschreibng
Skalar	`$`	`$pi`	skalare Variable
Array/Liste	`@`	`@feld`	indizierte Variable
Hash	`%`	`%keyvalue`	assoziatives Array
Unterpro-gramm	`&`	`&sub`	Perlcode

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Nichtinitialisierte Variablen

- nichtinitialisierte Variablen haben den Wert **`undef`**
- bei Zugriff verhalten sie sich wie numerische Nullwerte bzw. wie der leere String oder boolsches false
- Kontrolle über Warnings-Flag **`perl -w`**

- Methode **`defined`** überprüft, ob eine Variable initialisiert ist:
  - **`if (!defined($pi))`** …

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Listen in Perl

- Arrays in PERL sind **Listen**: geordnete, skalare Daten
- Array: „Variable, die Liste enthält“
  (jedes einzelne Element ist Skalar)
- Array-Variablenbezeichner beginnen mit **@**
- Syntax:

```
@beispiel = ("eins", "zwei", "drei");
```

- *keine* Festlegung der Länge
  (dynamische Lineare Liste)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

mehr zu Listen

- Zugriff auf i-tes Element von **`@beispiel`**:
  `$beispiel[$i]`
  wobei $i von 0 bis n - 1 läuft
  - **`print $beispiel[0];`**
- (literale) Ausgabe ganzer Liste, geordnet nach Index:
  - **`print @beispiel;`**
- Erzeugung mittels Listenkonstruktion
  - **`@beispiel2 = (1 .. 100);`** Liste Zahlen von 1 bis 100
  - speziell **leere Liste: ()** - Wert von @a vor erster Zuweisung
- **`$beispiel`** und **`@beispiel`** tun sich nicht weh

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Listen-Operationen

- **`push(@liste, $skalar);`** fügt am Kopf der Liste **`@liste`** den Wert **`$skalar`** hinzu
  (push und unshift auch mit mehreren Skalaren)
- **`$skalar = pop(@liste);`** entfernt einen Wert vom Kopf der Liste **`@liste`** und weist diesen Wert **`$skalar`** zu
- **`unshift(@liste, $skalar);`** fügt den Skalar **`$skalar`** als erstes (unterstes) Elemente in **`@liste`** ein
- **`$skalar = shift(@liste);`** entfernt ersten (untersten) Wert von **`@liste`** und weist ihn **`$skalar`** zu

Perl - GIT/liste.pl - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Quantum ... PHP Perl Resource Java

Navigator

- BWeSciT
- El_Hardouz
- gdi20072
- gdi20082
- GIT
  - .settings
  - basis
  - gandalf
  - temp
  - uebung
  - uebung_alt
  - .includepath
  - .project
  - a.exe
  - anmeldungen.pl
  - assArray.pl
  - auswert_1.pl
  - auswert_2.pl
  - auswert_3.pl
  - CgiFormular.pl
  - CgiOutput.pl

Outline

hello.pl | skalareVariable.pl | liste.pl

```
1  # Grundlagen Internet-Technologien
2  #
3  # Beispiel fuer den Umgang mit Listen
4
5  ### Deklaration und Initialisierung der Liste
6  @myList = (0,1,2);
7  print ("myList mit Index 2: $myList[2]\n");
8
9  ### Erweiterung, Variante 1
10 $myList[3] = 3;
11
12 ### Erweiterung, Variante 2
13 @myList = (@myList,4);
14
15 ### Erweiterung Variante 3
16 push(@myList,"fuenf");
17
18 print ("Die ganze Liste: @myList ");
19
20 print("\nHoechster Index in Liste: $#myList");
```

Tasks | Console

<terminated> liste.pl [Perl Local] Perl Interpreter

```
myList mit Index 2: 2
Die ganze Liste: 0 1 2 3 4 fuenf
Hoechster Index in Liste: 5
```

Writable | Insert | 1 : 1

die Liste @ARGV

- einem PERL-Skript können beim Start explizit Argumente übergeben werden
  - etwa vergleichbar `String [] args` in der Deklaration einer Java-main-Methode

- PERL legt die übergebenen Werte in dem Array **@ARGV** ab
- skalare Variable **$ARGV** enthält Anzahl der Elemente von **@ARGV**

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Hashes

- alternative Bezeichnung: »Assoziatives Array«
- ähnlich zum Array
- verwendet anstelle des Array-Index »natürliche Zahl« einen beliebigen Index in einem key-value-Paar
- Werte (außer dieser Zuordnung) ungeordnet
  - pop, shift, ... machen keinen Sinn für Hashes
- Hash-Variablenbezeichner mit führendem %
  - sehr praktisch für die Verarbeitung von HTML-Formularen!

Beispiel

- ```
  # Hash initialisieren
  %gruppe = (“ChampionsLeague", “Chelsea",
  “nichtgeschafft", “Bayern Muenchen");
  # Zugriff ueber den key:
  print "$gruppe{ChampionsLeague} ist der
  Gewinner";
  ```

- nicht beschränkt auf keys vom Typ String:

  ```
  %gruppe2 = (“ChampionsLeague", “Chelsea",
  “nichtgeschafft", “Bayern Muenchen“, “3”,
  “Kaiserslautern”);
  ```
  ```

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

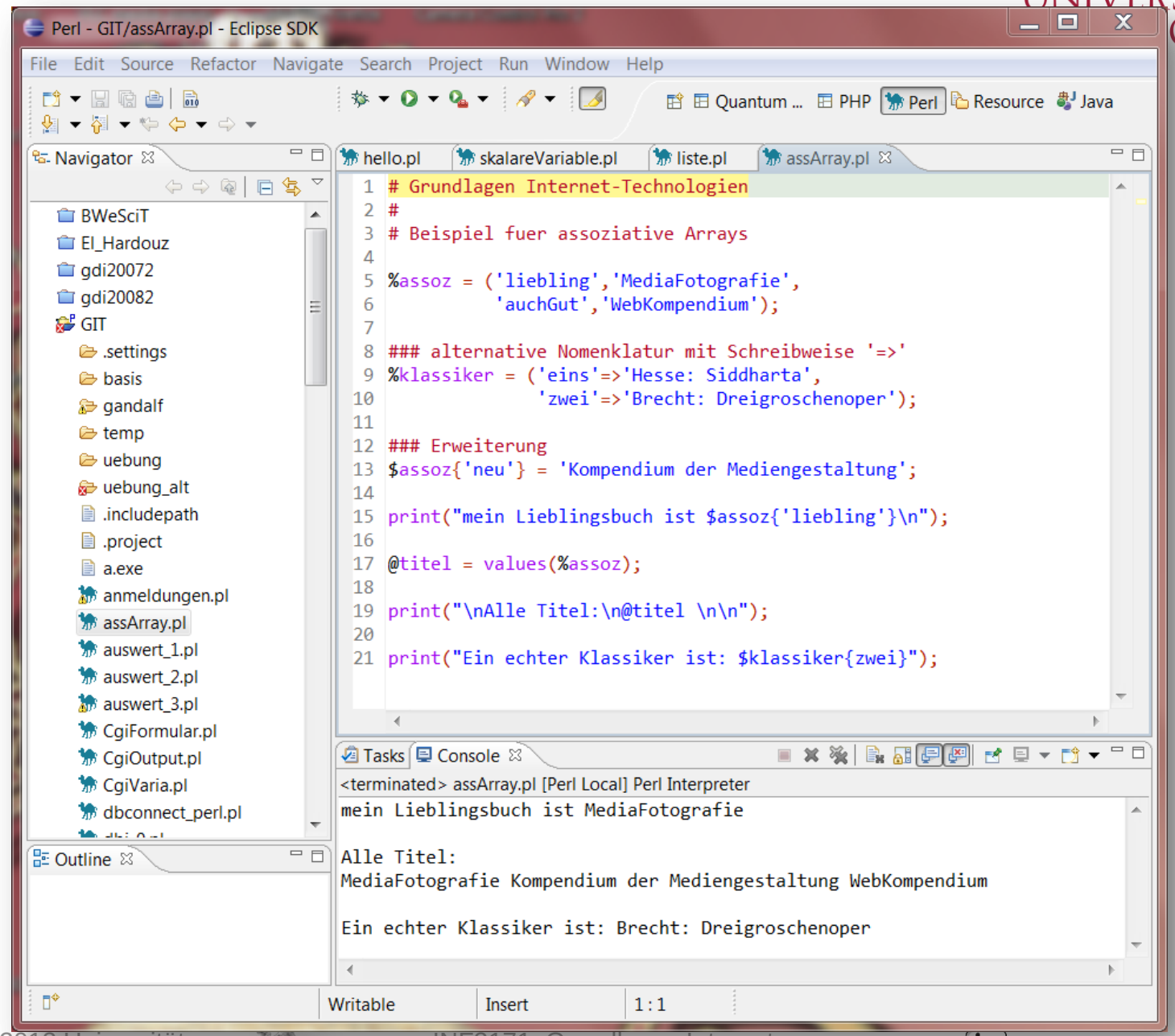
Hash-Funktionen

- **`@gruppe_liste = %gruppe`**
  liefert Liste mit 2n Elementen in zufälliger Reihenfolge
-
- **`%gruppe3 = reverse %gruppe`** liefert Hash %gruppe3 mit vertauschten key-value-Paaren

- **`@list = keys(%hash)`** liefert Liste mit den n key's eines Hash
  - **`@list = keys(%gruppe)`** liefert Liste ("chef", "alterchef", "partner") - nicht notwendig in dieser Reihenfolge

- **`@list = values(%hash)`** liefert analog Liste der n values

- **`each(%hash)`** liefert Liste mit key-value-Paaren

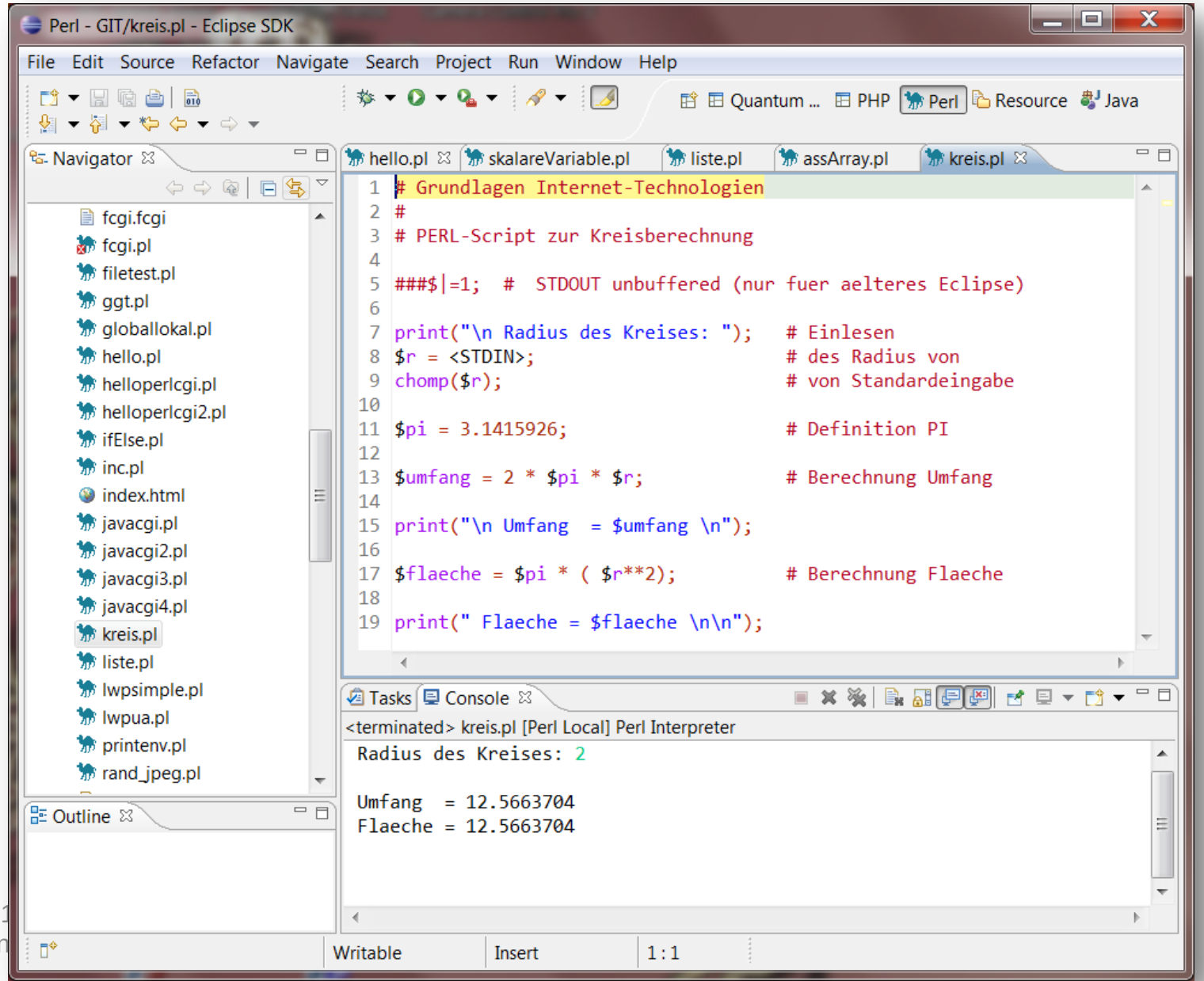

Perl - GIT/assArray.pl - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Quantum ... PHP Perl Resource Java
Navigator
BWeSciT
El_Hardouz
gdi20072
gdi20082
GIT
.settings
basis
gandalf
temp
uebung
uebung_alt
.includepath
.project
a.exe
anmeldungen.pl
assArray.pl
auswert_1.pl
auswert_2.pl
auswert_3.pl
CgiFormular.pl
CgiOutput.pl
CgiVaria.pl
dbconnect_perl.pl
Outline
hello.pl
skalareVariable.pl
liste.pl
assArray.pl
1 # Grundlagen Internet-Technologien
2 #
3 # Beispiel fuer assoziative Arrays
4
5 %assoz = ('liebling','MediaFotografie',
6 'auchGut','WebKompendium');
7
8 ### alternative Nomenklatur mit Schreibweise '=>'
9 %klassiker = ('eins'=>'Hesse: Siddharta',
10 'zwei'=>'Brecht: Dreigroschenoper');
11
12 ### Erweiterung
13 $assoz{'neu'} = 'Kompendium der Mediengestaltung';
14
15 print("mein Lieblingsbuch ist $assoz{'liebling'}\n");
16
17 @titel = values(%assoz);
18
19 print("\nAlle Titel:\n@titel \n\n");
20
21 print("Ein echter Klassiker ist: $klassiker{zwei}");
Tasks Console
<terminated> assArray.pl [Perl Local] Perl Interpreter
mein Lieblingsbuch ist MediaFotografie
Alle Titel:
MediaFotografie Kompendium der Mediengestaltung WebKompendium
Ein echter Klassiker ist: Brecht: Dreigroschenoper
Writable
Insert
1 : 1

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Funktionen in Perl

- PERL hat zahlreiche build-in-Funktionen (Programming PERL, pp. 145-242 (!))
- Beispiele:
  - `print, chomp`
  - `abs, exp, sin, hex`
  - `pop, push, keys, values`
  - `gethostbyaddr, use`

ein Beispiel



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Perl und Logik

- die boolschen Operationen not, and, or:
  wie in C oder in Textform
  - `&&` <=> `and`
  - `||` <=> `or`
  - `!` <=> `not`
- Vergleichsoperatoren: ebenso in beiden Formen
  - `$x == $y` <=> `$x eq $y`
  - `$x != $y` <=> `$x ne $y`
  - `$x < $y` <=> `$x lt $y`
  - `$x <= $y` <=> `$x le $y`
  - **aber: `==` numerischer** Vergleich, **`eq` String**-Vergleich ;-)
    - "123" == 123.00 ist true, "123" eq 123.00 ist false
    - 81 lt 9 ist true

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Kontrollstrukturen in Perl

- Anweisungsblöcke (sequenzielle Komposition)

  Anweisungsblöcke in PERL sind durch geschweifte Klammern begrenzt:

  - {

```
    erste_Anweisung;
    ...
    letzte_Anweisung
}
```

  (das letzte Semikolon vor } kann, muß aber nicht weggelassen werden)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

die if-Verzweigung

- Syntax:

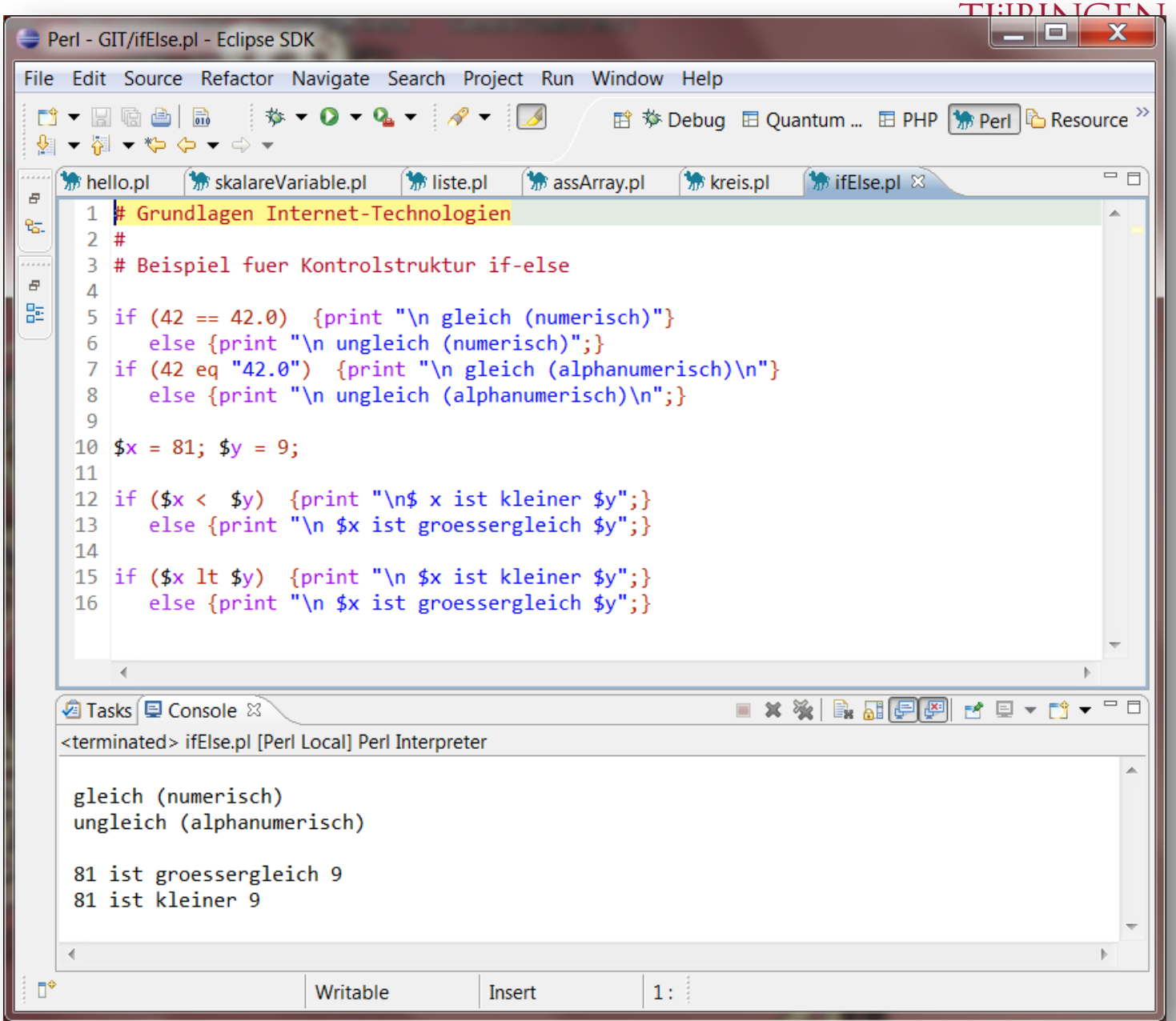
```
if (ausdruck)   {
   wahr_anweisungen;
}  else  {
    falsch_anweisungen;
}
```

der else-Teil kann wegfallen:

```
if (ausdruck)   { wahr_anweisungen; }
```

der if-Teil kann auch wegfallen:

```
unless (ausdruck){ falsch_anweisungen; }
```


Perl - GIT/ifElse.pl - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Quantum ... PHP Perl Resource
hello.pl skalareVariable.pl liste.pl assArray.pl kreis.pl ifElse.pl
1 # Grundlagen Internet-Technologien
2 #
3 # Beispiel fuer Kontrolstruktur if-else
4
5 if (42 == 42.0) {print "\n gleich (numerisch)"}
6 else {print "\n ungleich (numerisch)";}
7 if (42 eq "42.0") {print "\n gleich (alphanumerisch)\n"}
8 else {print "\n ungleich (alphanumerisch)\n";}
9
10 $x = 81; $y = 9;
11
12 if ($x < $y) {print "\n$ x ist kleiner $y";}
13 else {print "\n $x ist groessergleich $y";}
14
15 if ($x lt $y) {print "\n $x ist kleiner $y";}
16 else {print "\n $x ist groessergleich $y";}
Tasks Console
<terminated> ifElse.pl [Perl Local] Perl Interpreter
gleich (numerisch)
ungleich (alphanumerisch)
81 ist groessergleich 9
81 ist kleiner 9
Writable Insert 1 :

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

...noch mehr if...

- mit elsif (ausdruck) lassen sich weitere Kriterien für den else-Block festlegen

```
if (ausdruck_1)     {wahr_1_anweisungen;}
elsif (ausdruck_2)  {wahr_2_anweisungen;}
elsif (ausdruck_3)  {wahr_3_anweisungen;}
...
else  {  alles_falsch_anweisungen;  }
```

(es wird **höchstens** - wenn »else-Block vorhanden« genau - ein Block ausgeführt)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

die while-Schleife

- Grundtyp der Schleifen: führe die while_wahr_anweisungen aus, solange (ausdruck) wahr ist:

```
while (ausdruck)   {
    while_wahr_anweisungen;    }
```

- führe die until_falsch_anweisungen aus, solange (ausdruck) falsch ist:

```
until (ausdruck)   {
    until_falsch_anweisungen;    }
```

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

die do-while-Schleife

- while/until-Schleifen überprüfen die Kontrollbedingung **vor** Schleifendurchlauf, d.h. möglicherweise wird die Schleife nie durchlaufen
- mittels do { } while/until (ausdruck); wird der Schleifenblock mindestens einmal durchlaufen

```
do {
      anweisungen;
}  while (ausdruck);
```

(bzw. anstelle von »while« auch »until«)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

for-Schleife

- die for-Schleife in PERL:
  -
```
for (initialer_ausdr; test_ausdr;
inkrement)  {
    for_anweisungen;
}
```

- ...kann als while geschrieben werden:
  -
```
initialer_ausdruck;
while (test_ausdruck)   {
    anweisungen;
    neuer_ausdruck;
}
```

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Beenden von Schleifen

- mit der Anweisung **last;** wird jede Schleife vorzeitig beendet!

```
– while (ausdruck)  {
        anweisung;
        if (ausdruck2)  {last;}
        anweisung
  }
  # hier weiter, wenn last; ausgeführt wird!
```

- weitere ähnliche Anweisungen: **next;** und **redo;** (springen zu Ende/Anfang des Schleifenblocks)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

foreach-Schleife

- ...noch eine sehr praktische Kontrollstruktur für Listen:
  - `foreach $i (@liste) { anweisungen; }`
- z.B.

```
@a = (1, 2, 3, 4)
foreach $i (reverse @a) { print $i; }
```

  - gibt 4321 aus

Perl - GIT/schleifen.pl - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Quantum ... PHP Perl Resource

hello.pl liste.pl assArray.pl kreis.pl ifElse.pl schleifen.pl

```
1  # Grundlagen Internet-Technologien
2  #
3  # Schleifen in Perl
4
5  ### while ###
6  $n = 100; $i = 0; $summe = 0;
7  while($i <= $n) {
8      $summe += $i;
9      $i++;
10 }
11 print("\nErgebnis mit while-Summierung:     $summe");
12
13 ### do-while ###
14 $n = 100; $i = 0; $summe = 0;
15 do {
16     $summe += $i;
17     $i++;
18 } while($i <= $n);
19 print("\nErgebnis mit do-while-Summierung:  $summe");
20
21 ### for ###
22 $summe = 0;
23 for ($i = 0; $i <= $n; $i++)
24 {
25     $summe += $i;
26 }
27 print("\nErgebnis mit for-Summierung:       $summe");
28
29 ### foreach ###
30 @summe = (0..$n); $summe = 0;
31 foreach $i (@summe)
32 {
33     $summe += $i;
34 }
35 print("\nErgebnis mit foreach-Summierung:   $summe");
```

Writable Insert

Dateizugriff in Perl

- funktioniert über einen *"File-Handle"*: Bezeichner für eine IO-Verbindung
  - **`open(H, "$file");`** oder **`open(H, "<$file");`** erstellt Filehandle H für Datei, deren Namen in der Variablen $file abgelegt ist zum Lesen
  - **`$zeile = <H>;`** liest eine Zeile von H
    - deshalb Tastatureingabe $in = <STDIN>;
  - **`open(H, ">$file");`** erstellt Filehandle H für Datei, deren Namen in der Variablen $file abgelegt ist zum Schreiben
  - **`print(H $text);`** schreibt auf H
  - **`open(H, ">>$file");`** öffnet zum Anfügen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

strukturierte Programmierung

- Definition von Methoden:

```
sub nameDerMethode   {
     #   code
}
```

  - ohne (!!!) Übergabelisten
  - übergebene Parameter sind in der Liste `@_`
    - erster Parameter ist also `$_[0]`

Perl - GIT/subroutine.pl - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

Debug | Quantum ... | PHP | Perl | Resource

assArray.pl | kreis.pl | ifElse.pl | schleifen.pl | subroutine.pl

```
1  # Grundlagen Internet-Technologien
2  #
3  # Subroutines in PERL
4
5  print("\n Argument eingeben: ");
6  $arg = <STDIN>; chomp $arg;
7
8  $erg = fakultaet($arg);
9
10 print("\n Fakultaet($arg) = $erg");
11
12
13 ######################################################
14 sub fakultaet {
15
16     if ($_[0] < 0) {die("unzulaessiges Argument");}
17     if ($_[0] == 0) { return 1;} else {
18     return ($_[0] * fakultaet($_[0] - 1));}
19
20 }   ### sub fakultaet
```

Console

<terminated> subroutine.pl [Perl Local] Perl Interpreter

```
Argument eingeben: 5

Fakultaet(5) = 120
```

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Importieren von Perl-Code

- mittels der Perl-Anweisung

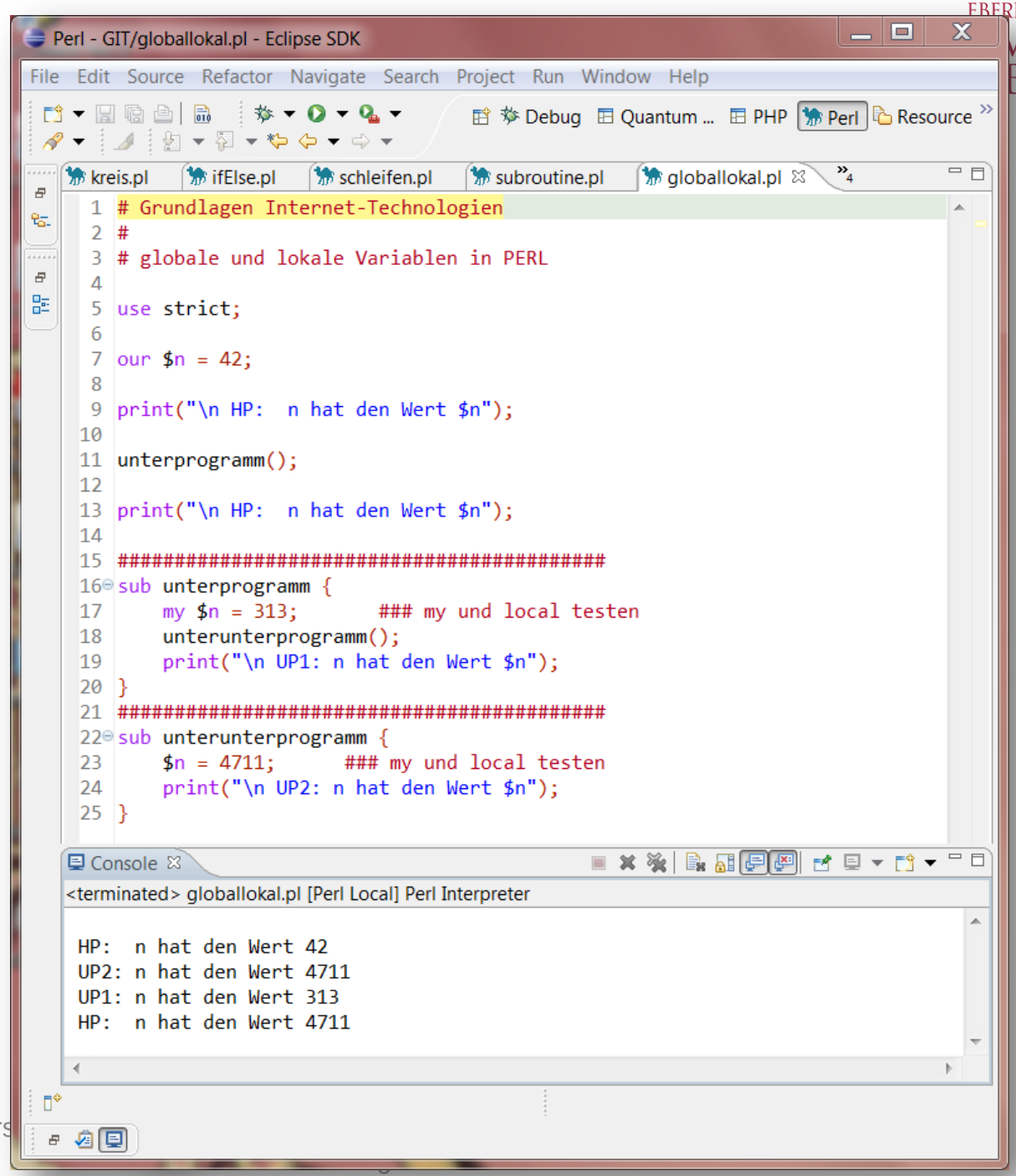
  `do(<filename>);`

  wird die Perl-Datei <filename> geladen und der Code ausgeführt

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

lokale und globale Variablen

- Umbruch mit PERL 5.6.0
  - „normale“ Variablen sind global
  - ab PERL 5.6.0
    - Schlüsselwort **`our`** erzeugt globale Variable
    - zwingend, wenn Modul **`strict`** verwendet wird
  - Schlüsselwörter **`my`** und **`local`** erzeugen lokale Variablen
    - my nur im Block gültig: neue lokale Variable
    - local in Block **und** Unterblöcken gültig



…und nun…

- haben wir die universelle Scriptsprache Perl grundlegend kennen gelernt
  - Prinzip, Variablen, Kontrollstrukturen und mehr

- als nächstes:

  Wir verwenden Perl,
  um effizient CGI-Programme
  für serverseitige Web-
  Applikationen zu schreiben