Zusammenfassung Webengineering

Philipp Pütz

31. Januar 2015

Inhaltsverzeichnis

Kapitel 1

Einführung in Webengineering

1.1 Das Internet

Das Internet ist ein Netzwerk das Standard Protokolle zum Austausch von testen, Grafiken und multimedialen Inhalten nutzt. Dabei isst der Inhalt über Webserver erhältlich. Diese Server werden durch sogenannte Web-Clients kontaktiert.

World Wide Web: The World Wide Web is the universe of network-accessible information, an embodiment of human knowledge. [World Wide Web Consortium] Das W3C entwickelt und setzt neue Standards z.B.: zu Internetprotokollen fest und verbreitet diese. Das WWW nutzt HTTP und CSS um Inhalte zu übermitteln und darzustellen.

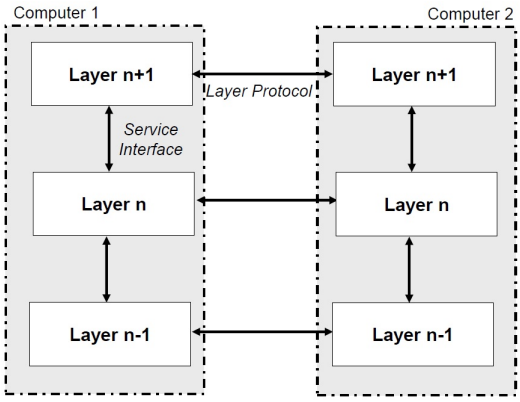
Internet Engineering Task Force (IETF): Das IETF entwickelt ähnlich wie das W3C neue Internetprotokolle oder updatet bestehende. Zudem ist es auf die Verbreitung und Bereitstellung dieser Standards spezialisiert.

1.1.1 Ideen und Ziele des Internets

- Einfacher Zugang zu Informationen mittels eines Standard User Interface.
- Inhalt kann auf verschiedenen Computern bzw. Arten von Endgeräten abgespielt.
- Demokratie ist gegeben, d.h.: jeder darf Informationen bereitstellen und abrufen.

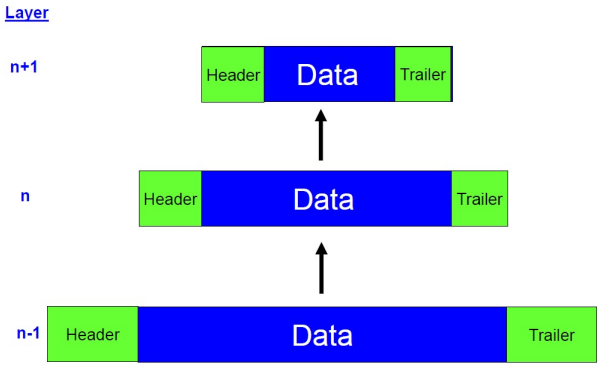
1.2 Schichtenmodell

Je höher das Level, desto größer ist die Abstraktion. Beispiel: Höchstes Level ist ein Textverarbeitungsprogramm und das niedrigste Level ist die Firmware eines Druckers.



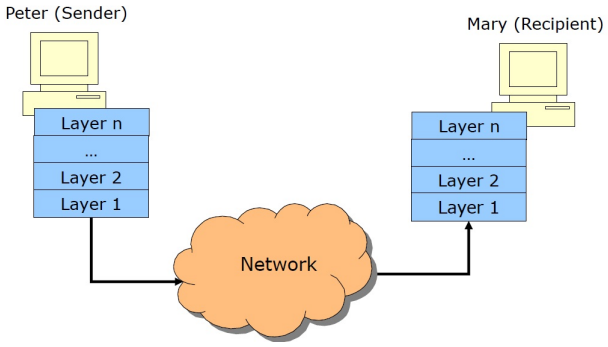
Das Ziel eines Schichtenmodell ist Probleme besser abstrahieren zu können und Probleme bzw. Systemkomponenten in kleinere Teilstücke aufteilen können.
Dieses System hat den Vorteil, dass neue Technologien einfach implementiert werden können und Probleme lassen sich besser eingrenzen.

1.2.1 Datenschichten



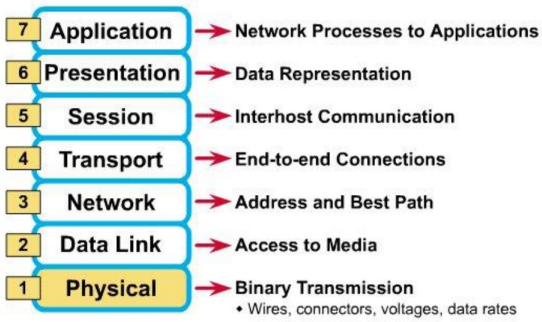
Je höher die Datenebene desto weniger Daten beinhaltet die Datenebene.

Beispiel: Ein E-Mail Programm interagiert nicht direkt mir dem Empfänger E-Mail Programm, sondern nutzt weitere/tiefere Ebenen um eine Nachricht zum Empfänger zu senden.

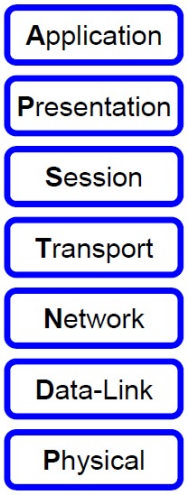


1.2.2 Das OSI Referenz-Modell

Das OSI Referenz Modell teilt die Datenübertragung in 7 Schichten und ist ein Referenzmodell um Netzwerkkommunikation zu verstehen.



Beispiel:



A-: Einen Brief für einen anderes Land schreiben.

P-: Den Brief übersetzen, damit der Empfänger ihn lesen kann

S-: Stellt sicher, dass der Empfänger den Brief auch empfangen kann.

T-: Den Brief aufteilen und Nummerieren und als Pakete versenden.

N-: Den Brief komprimieren, damit dieser besser versendet werden kann.

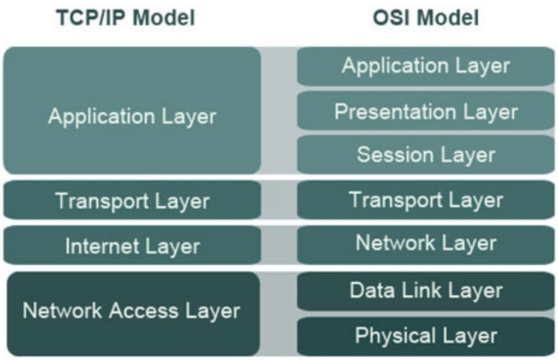
D-: Entscheidet welches Knotenpunkte der Brief passieren soll, damit dieser beim Empfänger ankommt.

P-: (physikalischer)Weg der der Brief auf dem Weg zu dem Empfänger bzw. zwischen den einzelnen Stationen zurücklegt.

Merksatz: All **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing.

Das OSI Referenzmodell und das TCP/IP Protokoll

Das OSI Referenzmodell kann wie folgt mit dem TCP/IP Protokoll verglichen werden:



Internet Protokoll Aufbau

5. Anwendungsschicht: DHCP, DNS, FTP, HTTP, IMAP4, POP3...

4. Transportschicht: TCP, UDP...

3. Vermittlungsschicht: IP, ARP...

2. Sicherungsschicht: WLAN, WiFi, Ethernet, ISDN...

1. Bitübertragungsschicht: Modems, Glasfaserkabel, Koaxialkabel, Patchkabel...

1.3 Internetprotokolle

Protokolle sind formale regeln die das 'Wie' spezifizieren. Die Aufgaben eines Protokolls sind:

- Adressierung von Endkomponenten
- Managen des Datenflusses
- Sicherstellung der sicheren Datenübertragung

1.3.1 IPv4 (Internet-Protokoll)

IPv4 ist die vierte Version des IP-Protokolls und wurde im Jahre 1981 definiert. Es ist das erste Protokoll, dass weltweit verbreitet wurde und bildete eine wichtige Grundlage zum Konzept des Internets. Das Internet-Protokoll wird zur *eindeutigen* Adressierung von Servern verwendet und verwendet 32-Bit Adressen (2^{32} Adressen) (Bsp.: 192.168.178.001:8080).

Die Adressierung:

- Netzwerk Adresse
- Internet Adresse
- Transport Protokoll Adresse
- Port Nummer

'Klassenbehaftete' Netzwerke sind ungünstig, weil man sich vorher überlegen muss, ob man eher viele Server oder mehr Hosts hat. Dies macht Systeme schlecht variable oder skalierbar.

Wie bekommt man eine IP-Adresse?

- Man kann einen Gerät eine statische IP-Adresse eine feste IP zuweisen.
- Der DHCP (Dynamic Host Configuration Protocol) vergibt eine zufällige IP-Adresse dem Gerät. Diese IP verfällt nach einer gewissen Zeit.

RFC1149

RFC1149 ist ein Aprilscherz, indem beschreiben wird, dass es möglich ist Datenpakete auf Papier zu schreiben und diese dann per Brieftaube an den Empfänger zu senden. Dabei ist eine hohe Latenz und Paketverlustrate gegeben.

Classful Network

Das ursprünglich eingesetzte Konzept der IP-Adressen sah nur eine starre Aufteilung vor. Hierbei waren 8 Bit für die Adressierung des Netzes vorgesehen, die übrigen 24 Bit adressierten einen spezifischen Teilnehmer des Netzes. Bei diesem Konzept waren aber nur 256 Netze möglich. Dies wurde als zu wenig erkannt. Daher wurden im September 1981 durch RFC 791 die sogenannten Netzklassen eingeführt, die diese Aufteilung neu gestalteten.
Über die Netzklassen wurde der gesamte Adressraum in zunächst drei (später fünf) Netzklassen unterteilt. Alle Teilnetze einer Netzklasse hatten hierbei dieselbe standardisierte Größe. Die Netzgrößen der Klassen unterschieden sich sehr stark, so waren in einem Netz der Klasse C nur 254 Hosts möglich, wohingegen bei einem Netz der Klasse A über 16 Millionen Hosts ermöglicht wurden. Dies sollte es ermöglichen, einzelnen Organisationen und Einrichtungen verschieden große Netzwerke je nach Bedarf zuzuweisen. Doch führten die starren Netzgrößen zu großer Verschwendung, da z.B. einem Anwender mit 100.000 Hosts ein Netz der Klasse A zugewiesen werden musste. Von diesen standen aber nur insgesamt 126 zur Verfügung und in diesem konkreten Fall wären über 16 Millionen IP-Adressen verschwendet worden.

Netzklasse	Präfix	Adressbereich	Netzmaske	Netzlänge (mit Präfix)	Netzlänge (ohne Präfix)	Hostlänge	Netze	Hosts pro Netz	CIDR Suffix Entsprechung
Klasse A	0...	0.0.0.0 – 127.255.255.255	255.0.0.0	8 Bit	7 Bit	24 Bit	128	16.777.214	/8
Klasse B	10...	128.0.0.0 – 191.255.255.255	255.255.0.0	16 Bit	14 Bit	16 Bit	16.384	65.534	/16
Klasse C	110...	192.0.0.0 – 223.255.255.255	255.255.255.0	24 Bit	21 Bit	8 Bit	2.097.152	254	/24

Class Inter-Domain Routing (CIDR)

Das CIDR wurde im Jahr 1993 von IETF entworfen um Probleme mit den 'klassenbehafteten' Netzwerken zu lösen. Es beschreibt ein Verfahren zur effizienteren Nutzung des bestehenden 32-Bit-IP-Adress-Raumes für IPv4. Zudem wurde es eingeführt um die Größe von Routingtabellen zu reduzieren und um die verfügbaren Adressbereiche besser auszunutzen (z.B.: $172.17.0.0/17$).
Mit CIDR entfällt die feste Zuordnung einer IPv4-Adresse zu einer Netzklasse, aus welcher die Präfixlänge hervorging. Die Präfixlänge ist mit CIDR frei wählbar und muss deshalb beim Aufschreiben eines IP-Subnetzes mit angegeben werden. Dazu verwendet man häufig eine Netzmaske.

1.3.2 IPv6 (Internet-Protokoll)

Das IPv6-Protokoll verwendet 128 Bit-Adressen (2^{128} Adressen) und wurde eingeführt, da die IPv4 Adressen in Asien und Europa drohen zu neige zu gehen.

1.3.3 TCP (Transmission Control Protocol)

Das TCP ist ein Netzwerkprotokoll, das definiert auf welche Art und Weise Daten zwischen Computern ausgetauscht werden sollen. Die erste Standardisierung von TCP erfolgte deshalb erst im Jahre 1981 als RFC 793. TCP setzt in den meisten Fällen auf das IP (Internet-Protokoll) auf, weshalb häufig auch vom „TCP/IP-Protokoll“ die Rede ist. In Protokollstapeln wie dem OSI-Modell sind TCP und IP nicht auf derselben Schicht angesiedelt. TCP ist eine Implementierung der Transportschicht.

1.3.4 Ports

- Ports sind allgemein Eingänge/Zugänge zu einem Host-Computer.
- Ports korrespondieren nicht mit echter Hardware.
- Programme oder Services sind meist mit einem bestimmten Port verbunden.
- Server warten darauf, dass über einen Port Daten versendet oder angefordert werden.
- Ports stellen ein Level der Internetsicherheit dar.
- Allgemein sind niedrige Portnummern für spezielle Programme reserviert.

1.3.5 DNS (Domain Name Service)

Die DNS übersetzt lesbare Internetadressen (z.B.: 'www.google.de') in richtige Internetadressen. Dabei besteht die lesbare Internetadresse aus Sub-Level Domains (z.B.: www.drive.google.de) und Top-Level Domains (z.B.: .de, .com).

1.3.6 FTP (File Transfer Protocol)

Der FTP-Service ist ein interaktiver Service mit dem Benutzer mit einem Computer/Server kommunizieren können. Mögliche Befehle sind z.B.:

- **Open:** Verbindung zu einem Computer herstellen
- **Get:** Fragt eine Datei auf dem Server an
- **Put:** Sendet eine Datei zu einem Computer

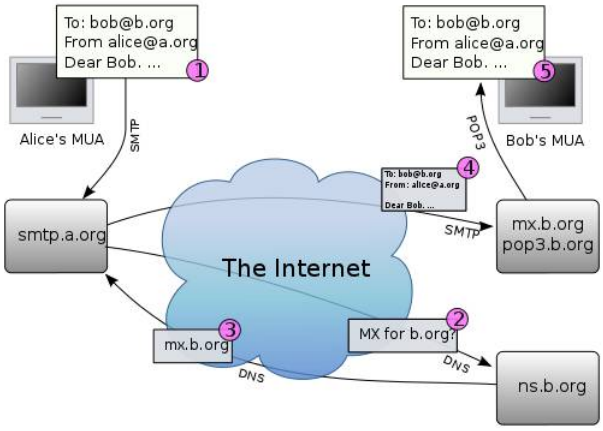
Zudem existieren zwei Übertragungsmodi (Aktive/Passive):

Aktiv: Der der Server managend den Datenstrom

Passiv: Der Client/Computer managend den Datenstrom

1.3.7 E-Mail Services

Um E-Mails von einen E-Mail Server herunterladen zu können existieren verschiedene Protokolle ($smtp$, pop, $imap$).



1.3.8 Telnet

Telnet ist ein Protokoll/Service der es einem User erlaubt mit einem anderen entfernten Computer zu kommunizieren und als Terminal zu nutzen. Dabei existiert aber kein interner Schutz.

1.3.9 Gropher

Gropher ist eine verbesserte Version des anonymen FTP-Service und ist nicht graphisch und ein hierarchisch strukturiertes Textsystem.

1.3.10 HTTP (Hypertext Transfer Protocol)

HTTP ist ein Internetprotokoll, dass Computerclients und Webserver nutzen um untereinander zu kommunizieren. Jede Nachricht, egal ob Anfrage oder Antwort, besteht aus 3 Teilen: HTTP ist ein reines Textübertragungsprotokoll. Zudem ist es ASCII codiert und überträgt Daten via TCP/IP.

1. Die Anfrage oder Antwort
2. Einer Art 'header'
3. Einer Art 'body'

Die wichtigsten HTTP-Befehle sind:

GET: Parameter werden zwischen URLs transferiert.

POST: Parameter werden unsichtbar übertragen.

Eine klassische HTTP-Übertragung hat 4 Phasen:

1. Client öffnet die Verbindung (TCP-Verbindung)
2. Client tätigt eine Anfrage
3. Server sendet eine Antwort
4. Server schließt die Verbindung

Wichtige Features von HTTP

- **Persistente Verbindung** (in HTTP 1.1): Nur eine Verbindung ist benötigt, um Daten zu übertragen. Vor HTTP 1.1 musste für jedes Objekt eine Verbindung hergestellt werden
- **Statuslos** (HTTP ist oder speichert keinen Zustand):
  - Jede Operation oder Transaktion öffnet eine neue Verbindung.
  - Jede Operation ist unabhängig von anderen Verbindungen.
  - Nach dem schließen einer Verbindung, werden keine Informationen über diese gespeichert.
- Der zu übertragende Inhalt ist verhandelbar (ob versendet als: z.B.: als zip oder Bild).

1.3.11 Webserver

Ein Webserver ist eine Implikation des HTTP-Protokolls. Ein Webserver kann auf verschiedenen Geräten wie Server, Toaster oder Routern laufen. Oft genutzte Webserver sind Apache oder MS Internet Informationen Server. Ein Webserver benötigt aber auch etwas serverseitiges Programmieren mittels z.B.: PHP, ASP.NET oder Ruby on Rails.

DIY Web Server

- Anwendung schreiben, die auf Port 80 'lauscht'
- Erkennen, ob Daten verfügbar sind
- Daten analysieren: Überprüfen ob es sich um einen korrekten HTTP-Reguest ist
- Request auswerten
- Request bearbeiten und Response erstellen
- Response zurückschicken

1.3.12 URI/URL

Uniform Resource Identifier (URI)

Eine URI fungiert als Adresse, Name oder als beides.

<uri> ::= <**scheme**>":"<**scheme**−specific part>

<**scheme**>

definiert ein Namensschema für eine URI

<**scheme**−specific part>

beinhaltet eine spezielle ID in Bezug auf das Schema **Beispiele:**

- *file* – Datei im lokalen Dateisystem
- *ftp* – File Transfer Protocol
- *http* – Hypertext Transfer Protocol
- *mailto* – E-Mail Addresse

Uniform Resource Locator (URL)

URL steht für Uniform Resource Locator und benutzt Definitionen wie http, https, ftp, mailto oder telnet Es ist möglich relative und absolute Pfade für URLs anzugeben ('.' man bleibt im selber Ordner, '..' man springt in den übergeordneten Ordner).

Genereller Aufbau einer URL

```
["//"] [user [":"password] "@"] host [":"port] ["/"url-path]
```

user: Optionaler Username (z.B.: ftp)

password: Optionales Passwort

- ⇒ Wenn ein Username oder Passwort angeben wird, folgt nach dem Username ein @-Zeichen. Jedes @-Zeichen, jeder ':' und jeder '/' im Username oder Passwort-Feld muss encodiert dargestellt werden.

host: Ein Domainname eines 'network host' oder die IP-Adresse

port: Die Portnummer zur der Kontakt hergestellt wird.

url-path: Spezifiziert die Details wie der 'Host' kontaktiert wird.

Regeln

Folgende Regeln gelten für Adressen:

- Als Alphabet wird ISO Latin-1 benutzt (ähnlich zu ASCII)
- Es dürfen keine Leerzeichen in der Adresse sein (durch %20 ersetzen)
- verschiedene Zeichen sind reserviert:
  - **%**: Identifiziert spezielle Zeichen
  - /: Trennt Verzeichnisse und Dateinamen
  - #: Trennt die URI von Datenobjekten oder Teilen davon
  - **?**: Trennt Anfragenstrings von Quellstrings

1.4 HTML

1.4.1 SGML

SGML ist eine meta-Sprache mit der man andere Sprachen entwickeln kann.

1.4.2 HTML

HTML und HTTP wurden von Tim Berners-Lee entwickelt. Berners-Lee Ziel war eine einfache Sprache zu erzeugen um Suchergebnisse zu präsentieren. HTML basiert auf SGML. Viele HTML-Standards werden durch das W3C entwickelt und festgelegt.

HTML Konzept und Syntax

- Text und formatierende Tags werden in HTML vereint
- Tags beginnen mit einer 'ı' und schließen mit einer 'ȷ'
- Alles was sich zwischen Klammern befindet gehört zu den Formatierungen
- Tags werden in der Entgegengesetzten Richtung geschlossen wie sie geöffnet wurden

  some text

Document Type Tag

Das Doyument Type Tag definiert welchen Formatierungsstandard das HTML-Dokument nutzt und beinhaltet einen Link zur DTD. Beispiel:

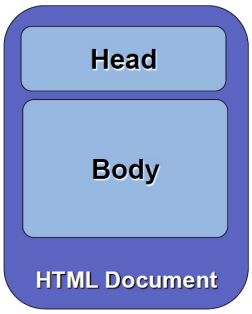
```
<!ELEMENT a %a.content;>
<!ATTLIST a
%attrs;
%focus;
charset         %Charset;       #IMPLIED
type        %ContentType;   #IMPLIED
name            NMTOKEN         #IMPLIED
href            %URI;           #IMPLIED
hreflang    %LanguageCode;  #IMPLIED
rel         %LinkTypes;     #IMPLIED
rev         %LinkTypes;     #IMPLIED
shape       %Shape; "rect"
coords      %Coords;        #IMPLIED
>
```

Aufbau einer Webseite

```
<html>
        <head>

        </head>
        <body>

        </body>
</html>
```



Header-Section

Das HTML-Tag zeigt den Beginn und das Ende eines HTML-Dokumentes an. Die Header-Section beinhaltet Meta-Informationen über den Inhalt der Webseite.

```
<meta name="author" content="John Doe" />
<meta name="keywords" content="Web Engineering, XHTML, HTML" />
```

Zudem kann man spezielle Tags im Header nutzen, die z.B.: den Nutzer nach 5 Sekunden weiterleiten:

```
<meta http-equiv="refresh" content="5; URL=http://www.google.de" />
```

Body-Section

Die Body-Section beinhaltete alle sichtbaren Informationen und kann zudem auch JavaScript Code enthalten.

```
<body onclick="alert('Hello World');">
```

Paragraphs

Text kann mittels des Paragraphen Tags zusammengefasst werden.

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Pellentesque sed arcu. Sed rutrum magna quis lacus. </p>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Pellentesque sed arcu. Sed rutrum magna quis lacus. </p>
```

Textformatierungen

- ** ** : verhindert Zeilenumbrüche bei Leerzeichen
- **** : erlaubt Zeilenumbrüche in Wörter
- <**br/**> : manueller Zeilenumbruch
- <**cite**> : Zitierung
- <**code**> : Quellcode
- <**dfn**> : Definitionsterm
- <**em**> : hervorgehobener Text
- <**kbd**> : Tastaturtext
- <**samp**> : Beispiel Quellcode
- <**strong**> : dicker hervorgehobener Text
- <**var**> : Variablen
- <**hr/**> : horizontale Linie

Headings

- <**h1**>Heading 1</**h1**>
- <**h2**>Heading 2</**h2**>
- <**h3**>Heading 3</**h3**>
- <**h4**>Heading 4</**h4**>
- <**h5**>Heading 5</**h5**>
- <**h6**>Heading 6</**h6**>

Vorformatierer Text

```
<pre>
+-----------------+-----------------------------+
| Name | Student ID |
+-----------------+-----------------------------+
| Peter | 1111 |
| Paul | 2222 |
| Mary | 3333 |
+-----------------+-----------------------------+
</pre>
```

Div-Tags

Div-Tags sind der Nachfolger von Tabellen basierenden Webseiten:

```
<div> Suspendisse massa risus, pellentesque vel, sollicitudin
ac, vestibulum sed, justo. Ut in nibh eu risus vulputate tempor.
</div>
```

geordnete (nummerierte) Liste

```
<ol start="4" style="list-style-type: decimal">
        <li>Element 1</li>
        <li>Element 2 </li>
        <li>Element 3 </li>
</ol>
```

Listen Style:

- decimal : 1.,2.,3. ...
- lower-roman : i.,ii.,iii. ...
- upper-roman : I.,II.,III. ...
- lower-alpha oder lower-latin : a.,b.,c. ...
- upper-alpha oder upper-latin : A.,B.,C. ...
- none : keine Nummerierung

ungeordnete Liste

```
<ul style="list-style-type: disc">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
</ul>
```

Listen Style:

- disk
- circle
- square
- none

Das Anchor Tag

```
<a href="url" title="Some Title">Some Text</a>
```

Das Anchor Tag dient dazu, einen Link hinter Text zu verbergen. Das Argument 'title' erscheint sobald man über dem Text sich mit der Maus befindet. Zudem können Tastatur Shortcuts und die Tab Order festgelegt werden.

```
<a href="url" accesskey="C" tabindex="3">Some Text 1</a>
<a href="url" accesskey="B" tabindex="2">Some Text 2</a>
```

Außerdem könne Links verschiedene Farben bezüglich ihres Status haben:

- **Link** : Standard Linkfarbe in einem Dokument
- **Active** : Link ist einem andern Tab oder Fenster geöffnet
- **Visited** : Link wurde bereits besucht
- **Hover** : Maus befindet sich über dem Link

Das Link Tag Das Link-Tag kann dazu benutzt werden um verschiedene zusätzliche Informationen einzubinden. Dieses muss in der Header-Section positioniert werden.

```
<a href="url" accesskey="C" tabindex="3">Some Text 1</a>
<a href="url" accesskey="B" tabindex="2">Some Text 2</a>
```

Hyperlinks

```
<head>
        <link rel="stylesheet" type="text/css" href="theme.css" />
</head>
```

Bilder

Einbindung von normalen Bildern:

```
<img src="path to image" alt="text for non-graphical user agents"
        width="200" height="200" />
```

Einbindung von anklickbaren Bildern:

```
<a href="mypage.html">
<img src="path to image" alt="text for non-graphical user agents"
        width="200" height="200" /> </a>
```

Image Maps

Image Maps erlauben es sogenannte Hotspots für ein Bild zu erstellen. Dabei werden diese Hotspot mit Koordinaten vom oberen linken Ecke aus angeben.

```
<img name="image_map" src="image_map.jpg"
        usemap="#m_image_map" alt="" />
<map name="m_image_map" id="m_image_map">
<area shape="rect" coords="226,8,392,188" href="right.html" />
<area shape="rect" coords="18,6,184,187" href="left.html" />
</map>
```

Dabei sind folgende Figuren möglich:

- **Rechtecke** werden durch die Angabe der oberen linken Ecke und der unteren rechten Ecke spezifiziert.
- **Kreise** werden durch die Angabe des Mittelpunktes und des Radius spezifiziert.
- **Polygone** werden durch jeden möglichen Punkte spezifiziert.

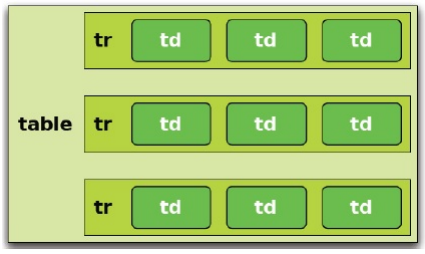
Tabellen

Die Tabellengröße kann relativ und absolut angeben werden. Dabei können absolute Angaben zu Scrollbalken führen.

```
<table border="1" width="50%">
<table border="1" width="350px">
```

Tabelle verwendet 50% der Bildschirmgröße des Benutzers. Die Tabelle der zweiten Zeile würde immer $350px$ einnehmen.

```
<table border="1">
        <tr>
                <td>First</td>
                <td>Second</td>
        </tr>
        <tr>
                <td>Third</td>
                <td>Fourth</td>
        </tr>
</table>
```



Tabellenüberschrift:

```
<caption>My Caption</caption>
```

Zelle überspannt zwei Spalten:

```
<th colspan="2">Second</td>
```

Zelle überspannt zwei Reihen:

```
<th rowspan="2">E-Mail:</th>
```

Forms

Die Form-Funktion überträgt Daten mittels HTTP GET (Daten werden an die URL angehangen) oder mittels HTTP POST (Daten werden unsichtbar im Body der HTTP Anfrage gespeichert). Dabei müssen alle Formelemente ein Namensattribut haben und eine eindeutige ID besitzen.

```
<form id="myform" action="form_handler_URL" method="get|post">
<!-- form elements -->
</form>
```

Text Input Box

```
<input type="text" name="MyText" id="idMyText" value="initialvalue"
        size="size_of_field" maxlength="max_characters"_allowed" />
```

Password Input Box

```
<p>Password:
        <input type="password" name="Password"
        id="ID_Password" value="" size="20" maxlength="40" />
</p>
```

Labels

```
<label for="id_of_related_tag">Label Text</label>
```

Radio Buttons

```
<input type="radio" name="control_group_name" id="ID_control"
        checked="checked" value="value_if_selected" />
```

Checkboxen

```
<input type="checkbox" name="field_name" id="id_of_checkbox"
        checked="checked" value="value_if_checked" />
```

Text Areas

Text Areas können bis zu 1024 Zeichen und Zeilenumbrüche beinhalten.

```
<textarea name="..." cols="number_of_columns"
        rows="number_of_rows">default_value</textarea>
```

Hidden Fields

Hidden Fields sind unsichtbar und werden benutzt um Daten zwischen verschiedenen Zeichen zu speichern.

```
<input type="hidden" name="field_name"
        value="field_value" />
```

Buttons

```
<input type="button" name="button_name"
        value="button_text" />
```

Bilder als Button

Bilder können als eine Art graphischer Button verwendet. Diese sollten aber nur dann verwendet werden wenn ein JavaScript event handler benutzt wird.

```
<input type="image" name="field_name"
        src="url_to_image" />
```

File Fields

File Fields werden dazu verwendet, um Dateien an den HTTP Request anzuhängen. Die Datei wird dann mittels POST-Methode übertragen

```
<input type="file" name="field_name"
        size="displayd_size" />
```

Submit-Button

Ein Submit-Button sendet das Formular zum form handler.

```
<input type="submit" name="submit" id="submit"
       value="button_text_submit" />
```

Reset-Button

Der Reset-Button setzt alle Felder/Formulare in den default Zustand zurück.

```
<input type="reset" name="reset" id="reset"
       value="button_text_reset" />
```

Fieldset

Formularelemente können mittels Fieldset zusammengefasst werden.

```
<fieldset>
       <legend>Caption for grouped fields </legend>
       <input ... />
       <input ... />
</fieldset>
```

Tabindex

Der Tabindex legt fest, in welcher Reihenfolge die Eingabefelder fokussiert werden, wenn die Tabulator Taste gedrückt wird.

```
<fieldset>
       <legend>Caption for grouped fields </legend>
       <input ... tabindex="1" accesskey="F" />
       <input ... tabindex="2" accesskey="E" />
</fieldset>
```

Felder disabeln/readonly

Felder können auf readonly gesetzt werden oder ganz gesperrt werden, damit keine Änderungen vorgenommen werden.

```
<input type="text" name="t1" id="id_t2"
       value="do_not_modify" readonly="readonly" />
<input type="text" name="t2" id="id_t2"
       value="do_not_modify" disabled="disabled" />
```

Plug-Ins

Plug-Ins werden verwendet um nicht-HTML Inhalte darzustellen (Java Applets, Flashvideos, MIDI). Dabei ersetzt das <**object**> Tag die "deprecated" Tags <embed> und <**applet**>.

Das <embed>-Tag bestimmt dabei selber den Content-Handler, sollte aber kein handler gefunden werden, wird einfach nichts angezeigt.
Der Syntax ist dabei: <embed **src**="URL_or_Path"</embed>.

Das <**object**>-Tag ersetzt das <embed>-Tag, doch haben dadurch neuere als auch ältere Browser Probleme, da meist der Ältere nur das <embed>-Tag kennt und <**object**>-Tag nicht. Bei neueren Browsern verhält es sich mit den Tags genau andersherum. Die Lösung ist beides zu kombinieren:

```
<object width="425" height="355">
       <param name="movie"
       value="http://www.youtube.com/v/lJwgP44Ap9E"></param>
```

```
        <param name="wmode" value="transparent"></param>
        <embed src="http://www.youtube.com/v/lJwgP44Ap9E"

        type="application/x- shockwave-flash"
        wmode="transparent" width="425" height="355"></embed>
</object>
```

1.4.3 XML Extensible Markup Language

XML basiert auch auf SGML und dient zum Austausch von Informationen. Zudem ist XML auch ASCII basierend und XML bietet den Nutzer die Möglichkeit eigene Tags zu definieren.

1.4.4 CSS (Cascading Style Sheets)

CSS beschreibt wie Daten oder Strukturen einer Webseite dargestellt werden. Dabei werden die .CSS-Dateien im Header der HTML-Datei eingebunden.

```
<style type="MIME_type" media="destination_media">
        /* comments in a stylesheet */
</style>

<head>
        <link rel="stylesheet" type="text/css" href="styles.css" />
        <link rel="stylesheet" type="text/css" href="another.css" />
</head>
```

"Cascading" bedeutet, dass einzelne Definitionen der Darstellung durch andere Definitionen oder spezifischere überschrieben werden können. Einzelne Elemente können dabei wie folgt angesprochen werden:

- **by (tag)name**:

  ```
  h1, h2, h4 { color: green;}
  ```

- **wildcard (universal selector)**:

  → betrifft alle Elemnte im Dokument:

  ```
  * {color: blue;}

  tr td ol { color: red;}
  /* Matches all ol elements following a td
  element following a tr element */
  ```

- **by class**:

  ```
  <p class="dark_background"> ... </p>
  CSS: p.dark_background {...}
  oder mittels Wildcard: .dark_background {...}
  oder: *.dark_background {...}
  ```

- **by Pseudoclass**:

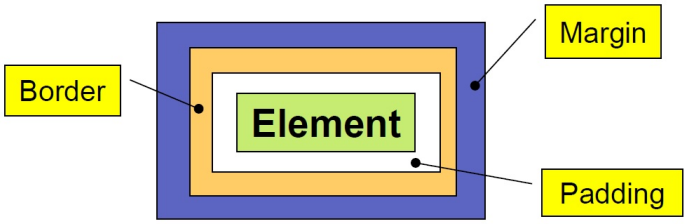
  - :link: unbesuchter Link

  - : visited : besuchter Link
  - : active: aktiver Link
  - :hover: Maus über Link
  - :focus: Tab auf den Link

- **by IDs**:

```
#source { ... }
<p id="source"> ... </p>
```

CSS Boxmodell



Standardmäßig werden alle XHTML-Elemente in sogenannten Boxen zusammengefasst. Diese Boxen kann man mit CSS wie folgt editieren:

- padding, padding−top, padding−right, padding−left, padding−bottom
- border−width, border−top−width, border−right−width, border−left−width,border−bottom−width
- boder-style: None, Hidden, Dotted, Dashed, Solid, Double...
- margin, margin−top, ...

Postions Elemente

CSS erlaubt vier Positionsmethoden:

- static (default mode)
- relative
- absolute
- fixed

```
p.rel {
        position: relative;
        top: 50px;
        left: 50px;
}
```

z-index

Mittels z-index kann man die dritte Dimension in eine Webseite einbauen. Der z-index gibt an, ob ein Element vor dem anderen oder dahinter ist.

```
z−index:1; (default: 0)
```

1.4.5 HTML Geschichte

- HTML 1.0 (1992)
  - Wurde nicht vom W3C spezifiziert
  - Unterstützte nur wenige Elemente (keine Bilder, Tabellen etc.)
  - Wurde nur vom Browser Mosaic 1.0 unterstützt
- HTML 2.0 (11/1995)
  - Hintergundfarben und Bilder wurden unterstützt
  - Einführung von Tabellen
- HTML 3.2 (01/1997)
  - Einführung von CSS
  - Neue Layout Tags
- HTML 4.0 (12/1997)
  - CSS wird Hauptbestandteil
- XML 1.0 (02/1998)
  - Anwendung die auf SGML aufbaut
  - Universalle Sprache um Daten zu beschreiben und zu formatieren
- CSS 1.0 (1996) and 2.0 (1998)
  - Trennung von Inhalt und Design
  - Design kann als 'sytlesheet' (*.css) oder direkt in der HTML-Datei eingebunden werden
  - 'cascading' bedeutet dass verschiedene Styles durch untergeordnete Style Definitionen überschrieben werden können.
- HTML 4.01(12/1999)
  - Fehlerbehebung von HTML 4.0
- XHTML 1.0 (01/2000)
  - Ähnlich zu HTML
  - Kann ähnlich wieder der XML-Syntax durch Programme auf Fehler überprüft werden
  - Seiten basieren auf drei Vorlagen ('document type definitions (DTDs)')
  - Restriktiver als HTML
- XHTML 1.1 (05/2001)
  - Design/Formatierungen nur mittels CSS möglich
- XHTML 2.0
  - Ziel: Nur XML basierend
  - 2009: W3C erklärt XHTML 2.0 als beendet
- CSS3
  - Neuer Sprachoutput, mehr Farben, Hintergründe, Texteffekte

- HTML5
  - Zuerst als 'Web Applications 1.0' im Jahr 2004 veröffentlicht
  - Erster Entwurf im Jar 2008
  - Oktober 2014: Neudefinition
  - Von vielen Browsern unterstützt, aber nicht vollständig implementiert

1.4.6 ACID2 Test

Der ACID Test ist ein Testprogramm vom Webstandard Projekt 1995 realisiert wurde. Es testet HTML und CSS Funktionen.

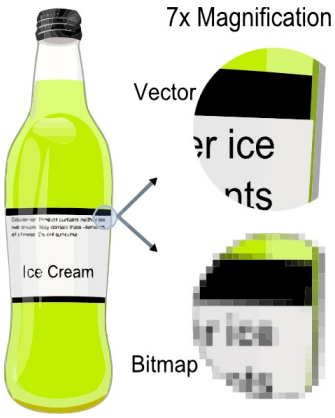
1.4.7 Nichtfunktionale Anforderungen

"Nichtfunktionale Anforderungen (nonfunctional requirements, kurz NFRs), auch Technische Anforderungen oder Quality of Service (QoS) genannt, beschreiben Aspekte, die typischerweise mehrere oder alle funktionale Anforderungen betreffen bzw. überschneiden (cross-cut). Zu den NFRs zählen u. a. Genauigkeit, Verfügbarkeit, Nebenläufigkeit, Konsumierbarkeit (eine Obermenge der Benutzbarkeit), Internationalisierung, Betriebseigenschaften, Zuverlässigkeit, Sicherheit, Service-Anforderungen, Support." [Balzert 2009], s. 463

- [Pohl07, S. 16 ff.] in [Balzert 2009, s. 463]: Nichtfunktionale Anforderungen sind entweder Qualitätsanforderungen oder unterspezifizierte funktionale Anforderungen
- "Eine Qualitätsanforderung legt eine qualitative und/oder quantitative Eigenschaft des Softwaresystems oder einer seiner Komponenten fest." [Balzert 2009], s. 463

1.5 Bilder

1.5.1 Arten von Bildern



Vektorgrafiken: Vektorgrafiken werden als eine Sammlung von Vektoren gespeichert und benötigen nur sehr wenig Speicher. Dafür ist das Rendering der Bilder komplexer, aber dadurch hat man bei Vergrößerung der Bilder keinen Qualitätsverlust.

Bitmap/Raster Grafiken: Rastergrafiken werden als eine Art Array aus Pixels gespeichert und benötigen etwas mehr Speicher als Vektorgrafiken. Dafür ist das Rendering viel weniger Aufwändig.

Bitmap Grafiken

- Moderne Displays basieren auf Raster, indem jedes Pixel eine bestimmte Farbe annehmen kann
- Die Farbe wird oft durch sogenannte Subpixel erzeugt die rot, grün und blau sind (RGB)

Bitbreite von Bilder

Bilder können verschiedene Bitbreiten haben und daraus resultieren auch verschiedene Bildqualitäten. 24-Bit Bitmap Bilder haben Fotoqualität und jeder Pixel wird als RGB-Farbe dargestellt. Daraus resultieren $256^3 (16.777.216)$ verschiedene Farbkombinationen.

8-Bit Bitmap Bilder benutzen weniger Farbkombinationen, sodass 8 Bit Bilder sogenannte CLUT (color lock-up tables) benötigen um zu entscheiden welche von den Millionen möglichen Farbkombinationen, die ein Bildschirm darstellen kann, für die Farben im Bild verwendet werden. Dabei benötigen 8 Bit Bilder aber auch nur ca. $\frac{1}{3}$ des Speicherplatzes eines 24 Bit Bildes.

Graustufen Bilder und Monochrome Bilder

Graustufenbilder verwenden Grautöne um das Bild darzustellen. Dabei wird jedem Pixel ein Wert zwischen 0 und 255 zugeordnet.
Hingegen wird bei Monochrombildern (schwarz/weiß) für jeden Pixel nur der Wert 0 oder 1 gespeichert. Das Bild benötigt daher auch sehr viel weniger Speicherplatz.

RGB Farbenmodell

RGB ist ein additives Farbenmodell, welche die Farben rot, grün und blau verwendet. Farben werden durch die 'Höhe'/Level der einzelnen Farben bestimmt. $(255, 255, 255 \rightarrow$ Weiß, $0, 0, 0 \rightarrow$ Schwarz)

RGBA Farbenmodell

Das RGBA Farbenmodell nutzt zusätzlich zu den RGB-Farben noch einen Alpha-Kanal der die Transparenz der Farbe angibt.

CMYK Farbenmodell

Das CMYK Farbenmodell nutzt die Farben cyan, magenta, yellow und black. Farben werden hier auch durch das Level der Farben bestimmt ($255, 255, 255, 255 \rightarrow$ Black, $0, 0, 0, 0 \rightarrow$ White). Das CMYK Farbenmodell wird häufig für Drucker verwendet. Das CMYK Farbenmodell kann folgendermaßen in das RGB Farbenmodell umgerechnet werden:

- $R = 1.0 - (C + K)$
- $G = 1.0 - (M + K)$
- $R = 1.0 - (Y + K)$

Bitmap Grafikformate

- **GIF (Grafic Interchange Format):**
  - benutzt Lempel-Ziv-Welch (LZW) komprimierung (Wörterbuchbasierend)
  - unterstützt eine transparente Farbe
  - verlustfreie Komprimierung
  - Unterstützt einfache Informationen
  - Unterstützt bis zu 256 verschiedene Farben
- **JPG (Joint Photographic Experts Group):**
  - Unterstützt das 24 Bit Farbensystem, lässt aber keine transparenten Farben zu

  - ist ein nicht-verlustfreies Komprimierungsformat, aber die Kompressionsrate für visuell verlustfrei beträgt $1:15$

- **PNG (Portable Network Graphics)):**
  - Unterstützt das 24 Bit Farbensystem und transparente Farben
  - verlustfreies Komprimierungsformat

Vektorgrafiken

- Vektorgrafikdateien werden häufig benutzt um Formel für analytische Geometrie darzustellen (Linien, Rechtecke, Ellipsen)
- Werden für einfache Grafiken oder technische Zeichnungen benutzt
- Vektorgrafiken können einfach durch Veränderung der Koeffizienten skaliert werden.

Postscript: Vektorgrafiken und Text

Postscript ist eine Programmiersprache die von Adobe im Jahr 1982 einwickelt wurde. Die Dateianhängung ist *.ps und wird heute noch zum Teil in PDFs verwendet.

SVG - Scalable Vector Graphic Format

SVG ist ein XML basierendes Vektorformat das durch W3C entwickelt wurde. SVG wird im Browser mittels Plug-In (IE) oder implementierter SVG Unterstützung.

transparente Bilder

Nichtransparente Bilder (z.B.: .jpg) verwenden den gesamten Bereich, hingegen kann man mit .png Farben definieren, die während des Rendering transparent werden.

1.5.2 Kompression

Fast alle Grafik- oder Videoformate besitzen verschiedene Varianten um Bilder zu verkleinern. Diese Kompressionsarten kann man in verlustfreie und nicht-verlustfreie Methoden unterteilt werden.

verlustfreie Methoden: Format beinhaltet alle Daten des original Bildes. Das Bild wird komprimiert ohne das Daten verloren gehen.

nicht-verlustfreie Methoden: Aufgrund der Beschränktheit des menschlichen Seh-Hör-Vermögens können Informationen gelöscht oder eingespart werden. Dadurch gehen einige Informationen verloren, die aber nicht Sichtbar oder Hörbar sind.

1.6 Display Technologien

- CRT (cathode ray tube)
- LCD (liqiud crystal display)
- TFT (thin film transistor)
- Plasma

1.7 Zeichendarstellung

- **ASCII (American Standard for Information Interchange)**
  - 7-Bit Code = 128 Zeichen
  - keine Sonderzeichen
- **ISO 8859-1**
  - latin Alphabet encoding (256 Zeichen)
  - beinhaltet Sonderzeichen
- **Windows-1252**
  - basiert auf ISO 8859-1
- **Unicode**
  - mehr als 100.000 Zeichen werden unterstützt
  - ersten 128 Zeichen = ASCII, ersten 256 Zeichen = ISO 8859-1
  - die wichtigsten Zeichensätze sind UTF-8 und UTF-16

1.7.1 Spezifizierung der Zeichendarstellung

XML:

```
<?xml version="1.0" encoding="utf-8"?>
```

HTML:

```
<meta http-equiv="Content-Type"
      content="text/html;charset=ascii" >
```

Wenn beides angegeben ist, ist wird die XML Spezifizierung ausgeführt. Die Zeichendarstellung kann nur dann selber deifniert werden, wenn Server seitig keine Encodierung vorgeschrieben wird. Diese Standardencodierung kann nur vom Serveradmin oder mittels Server seitigem Skript angepasst werden (z.B.: PHP, ASP.NET)

1.8 JavaScript

JavaScript ist eine interpretierte und objektorientierte Programmiersprache. JavaScript wurde 1995 veröffentlicht und wird vom Browser selbst interpretiert. JavaScript wurde mittels ECMAScript (ECMA: European Computer Manufacturer's Association) standardisiert. Trotzdem muss JavaScript Code für verschiedene Browser und Systeme angepasst und getestet werden.

1.8.1 Aufgaben von Javascript

- Eingaben validieren
- Dynamische Modifizierung von XHTML-Webseiten
- Dynamische Seiteneffekt

1.8.2 Syntax

```
<html>
        <body>
                <script language="JavaScript">
                statement 1;
                statement 2;
                ......
                statement n;
                </script>
        </body>
</html>
```

Beispiel Programm:

```
<script language="JavaScript">
        document.write ("Hello World!");
        /* multi line comment: Here the methode write()
        of the object document is called. */
        //single line comment
</script>
```

Beispiel Programm mit HTML:

```
<html>
        <head>
                <title>My First Java Script</title>
        </head>
        <body>
                <p>
                <b>This is a line of code before the script</b><br />

                <SCRIPT language="JavaScript">
                        var rightNow = new Date();
                        document.write("This text was written
                        with JavaScript! ");
                        document.write("It is now!" + rightNow);
                        /* Here the methode write() of the
                        object document is called. */
                </SCRIPT>

                <br /><b>This is a line of code after the script</b>
                </b>
        </body>
</html>
```

JavaScript kann im Header oder Body Bereich definiert werden und unbegrenzt viele verschiedene Codeschnipsel einbinden.

Beispiel Programm mit XHTML:

```
<script>
document.write("Hello World");
</script>
```

```
<body>
<script language="JavaScript" src="hello.js">
</script>
</body>
```

Link

```
<a href="javascript:window.alert( 'Hello World!');">
```

Event Handler

```
<a href="#" onclick="alert('Hello World!');">Hello 1</a>

<!-- OR -->

<a href="page2.html" onclick="alert(Hello World2!');">Hello World2</a>
```

Variablen

Variablen müssen einen Namen haben und speichern Daten. Dabei wird auch zwischen lokalen und globalen Variablen unterschieden.

```
var i = 1; //define and initialise i
i++; //(add 1 to i)
var x = 2.71;
x = x-1;
var y = Math.sqrt(25);
```

Variablenarten

- numerische Variablen: PI = 3,14156
- Strings: sCourse = "Web Engineering I"; // or 'Web Engineering I'
  - \n: neue Zeile
  - \r: Zeilenumbruch
  - \t: Tab
  - \b: Backspace
  - \f: page forward
- Boolean Type: bIsNumeric = true;
- Object Type: myObject = new object();

Jede Variable die nicht näher spezifiziert wurde (der kein Wert zugewiesen wurde), wird standardmäßig als undefiniert initialisiert.

Arrays

Arrays sind Objekte und können jegliche Daten enthalten. Zudem muss die Größe des Arrays nicht angeben werden.

```
var arr = new Array();
arr[0] = "Element1";
arr[2] = "Element1";
```

String Operationen

- +: Strings zusammenfügen

```
sComplete = "Hello" + "World" + "!";
```

- .length: Anzahl der Zeichen

```
sMyString = "123456"; sLen = sMyString.length; //=6
```

- .charAt(x): Gibt das Zeichen das an der Stelle x steht zurück

```
sMyString = "654321"; sLen = sMyString.charAt("2"); //=4
```

- .substring(start, end): Gibt einen Substring zurück

```
sVar1 = "Hello World"; sVar2 = sVar1.substring(6,11);
```

Exception Handling

Der Code im try-Block wird solange ausgeführt bis ein Fehler auftritt, sollte dies geschehen wird der Code im catch-Block ausgeführt.

```
try { // Statements 1
} catch (ex) {
// Statements 2
}
```

Zudem ist es möglich eigene Exceptions zu erzeugen: throw("My Exception");

Schleifen

Schleifen oder Switch-Case Anweisungen sind ähnlich bzw. gleich denen in C. Folgende Anweisungen existieren aber zusätzlich:

Iteration durch ein Array:

```
for (Property_Name in Object){
        Statements;
}
```

Weitere Beispiele:

```
for (var i=0, s=""; s.length<=10; i++, s += i) {
        document.write(s + "<br />");
}
```

```
var course;
var myCourses = new Array();
myCourses[0] = "Web Engineering I";
myCourses[1] = "Web Engineering II";
myCourses[2] = "Java Programming";
for (course in myCourses){
        document.write(myCourses[course] + "<br />");
}
```

Funktionen

Funktionsparameter sind undefiniert und jede Funktion muss ein return-Statement enthalten.

```
function myFunction (Param1, Param2,...) {
        Statements;
        return Value;
}
```

Eval-Funktion

Mittels der eval-Funktion kann ein gegebener Text als Code aufgefasst werden:

```
var car1 ="BMW";
var car2 ="VW";
var car3="Volvo";
for(i=0;i<3;i++)
document.write( eval("car"+(i+1)) + "<br/>");
```

Objekte

JavaScript ist objektorientiert. Objekte können Sprachelemente, Browserobjekte oder selbst definierte Objekte sein. Eigene Objekte können mit der new()-Funktion erzeugt werden.

Erzeugung von Objekten:

(a) direkte Instanziierung:

```
myObject = new Object();
myObject.name = "Peter Parker";
myObject.ZIP = "12345";
```

(b) Erzeugung mittels Template (Klasse):

```
function ba_student(firstname,lastname,employer){
this.firstname=firstname;
this.lastname=lastname;
this.employer=employer;
}
```

String Objekt

- bold(): Zeigt einen String fett an
- charAt(): Gibt das Zeichen an einer bestimmten Stelle zurück
- link(): Zeigt einen String als Hyperlink an
- replace(): Ersetzt Zeichen durch andere
- slice(): Erzeugt einen Teilstring
- toLowerCase(): Zeigt einen String in Kleinbuchstaben an length: Gibt die Anzahl der Zeichen in einem String zurück

Array Objekte

- concat(): Fügt zwei Arrays zusammen
- pop(): Entfernt das letzte Element und gibt dieses zurück
- reverse (): Dreht den String um
- sort (): Sortiert die Elemente eines Strings
- toString (): Konvertiert ein Array zu einem String length: Gibt die Anzahl der Zeichen in einem Array zurück

Browser Informationen abfragen

- Window - steht für das Browserfenster
- Navigator - Beinhaltet Informationen über den Browser
- Screen - Informationen über den Display
- History - Besuchte Seiten im Browserfenster
- Location - Informationen über die aktuelle URL

Browser und Client Informationen sind im navigator Objekt gespeichert.

%ADD

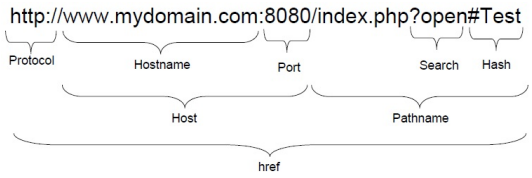
Window Objekt

- [window.] alert (<**text**>) – Alarm Information
- [window.]confirm(<**text**>) – Ja/Nein Popup
- [window.]prompt(<**text**>, <default_value>) – Eingabewert abfragen

Neues Fenster öffnen:

```
var mywin = window.open("http://www.google.de",
"Google", "width=400,status=1");
```

Location Objekt




```
location.href = "http://www.google.de";
```

Event Handling

Events sind asynchrone Aktionen die bei bestimmten Aktionen auftreten.

```
<html><head>
<title>Mous Events</title></head>
<body onload="window.status='';">
        <h1>Mouse Events</h1>
        <form>
                <input type="button" value="Click me!"
                onmousedown="window.status+='[mousedown]';"
                onmouseup="window.status+='[mouseup]';"
                onclick="window.status+='[click]';" />
        </form>
</body></html>
```

Cookie Handling

Ein Cookie ist eine gespeicherte Variable auf dem PC des Besuchers. Diese kann verschiedene Werte wie Usernamen, Passwörter und ein Datum speichern.

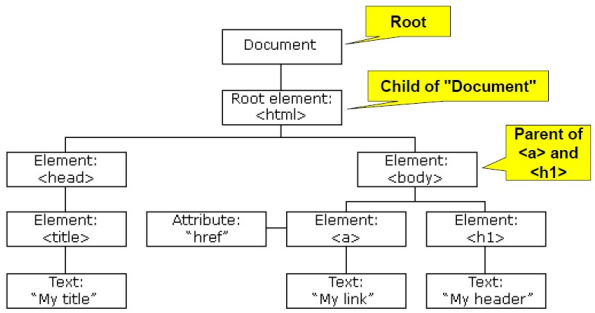
Navigation

Mit Hilfe der History kann man eine Art 'Navigation' erzeugen:

```
<a href="javascript:history.back();">Back</a> -
<a href="javascript:history.forward();">Forward</a>
```

1.8.3 Document Object Model (DOM)

Die DOM repräsentiert den Aufbau einer HTML Datei und bietet Zugriff auf die verschiedenen HTML-Elemente. DOM ist dabei ein W3C Standard und sollte in allen Browsern auch Standard sein. Der Aufbau einer Webseite lässt sich als Baumstruktur darstellen:



Um auf die verschiedenen HTML-Elemente zugreifen zu können gibt es folgende Tags:

- document.getElementById("someID");
- document.getElementsByTagName("tagname");
- document.getElementsByName("name");

- document.body: direkter Zugriff auf den Body selbst

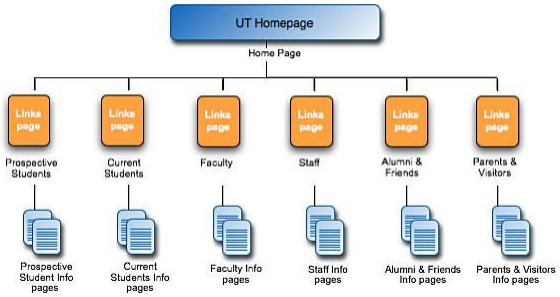
Weiterhin gibt es die Möglichkeit über Knoten (erstes Kind, letztes Kind, Elternknoten etc.) Zugriff zu erlangen.

Kapitel 2

Graphentheorie

2.1 Hypertext-Struktur

HTML-Seiten enthalten Hyperlinks zu anderen HTML-Seiten oder sich selbst. Außerdem drücken Hyperlinks eine Relation aus (Quelle $\rightarrow$ Ziel). Sitemaps geben diese Relationen/Struktur grafisch wieder.



2.2 Graphen

2.2.1 Definition von Graphen

Ein Graph $G = (V, E)$ ist eine Datenstruktur mit folgenden Eigenschaften:

- $V <> 0$ ist eine endliche, nichtleere Menge, die "Knotenmenge". Die Elemente $v \epsilon V$ werden als Knoten bezeichnet. V steht für vertices, englisch für "Eckpunkt, Knoten".
- $E \subseteq VxV$ ist eine Menge von Knotenpaaren, die "Kantenmenge". Die Elemente $(u, v) \epsilon E$ werden als Kante bezeichnet; Elemente $(v, v) \epsilon E$ als Schlinge oder (direkter) Zyklus. E steht für edges, also "Kanten"

2.2.2 Gerichtete und ungerichteter Graph

ungerichteter Graph

In einem ungerichteten Graphen ist die Anordnung der Knoten innerhalb eines Knotenpaars irrelevant: $(u, v,) = (v, u)$. Beim ungerichteten Graph gelten Knoten u, v als benachbart (adjazent), wenn $(u, v)\epsilon E \vee (v, u)\epsilon E$ gilt.

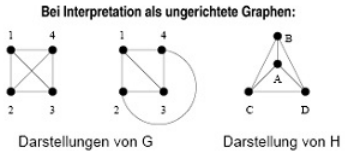
gerichteter Graph

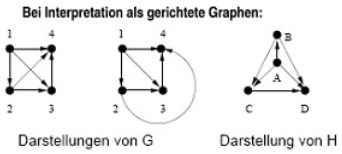
In einem gerichteten Graph hingegen gibt (u, v) an, dass eine Kante von u nach v existiert. u wird dann als die Quelle oder Startknoten, v als Endknoten der Kante (u, v) bezeichnet und auch als $u \to v$ notiert.

- Teilen sich zwei Kanten einen Endknoten, so sind diese inzident
- Der Grad $deg(a)$ eines Knotens a ist die Anzahl der Kanten, die inzident mit dem Knoten sind (Anzahl der Kanten, die vom Knoten "ausgehen")
- Einen Knoten, dessen Grad gleich 0 ist, bezeichnet man als isoliert

2.2.3 Darstellung von Graphen

$$G = (V, E)\ mit\ V_G = 1, 2, 3, 4,\ E_G = (1, 2)(1, 3), (1, 4), (2, 3)(2, 4), (3, 4)$$
$$H = (V, E)\ mit\ V_H = A, B, C, D,\ E_H = (A, B)(B, C), (C, D), (A, D)(A, C), (B, D)$$





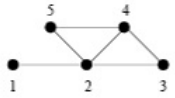
Kantenfolge, Kantenzug und Weg

Sei$G = (V, E)$ Graph $k = v_0, v_1, ..., v_n$ eine Folge von $n + 1$ Knoten von G.

- k heißt **Kantenfolge** der Länge n von v_0 nach v_n, wenn für $0 \leq i \leq n-1$ gilt: $(v_i, v_{i+1})\epsilon E$. Falls G gerichtet ist, ist v_0 der Startknoten und v_n der Endknoten, andernfalls sind beide Endknoten der Kantenfolge k.
- k ist ein **Kantenzug** der Länge n von v_0 nach v_n, falls k eine Kantenfolge der Länge n von v_0 nach v_n ist und für $0 \leq i, j \leq n - 1$ gilt: $(vi, vi + 1) \neq (vi, vi + 1)$. Jede Kante darf in einem Kantenzug also maximal einmal auftreten.
- k ist ein **Weg der Länge** n von v_0 nach v_n, wenn k eine Kantenfolge der Länge n von v_0 nach v_n ist und für $0 \leq i, j \leq n$ gilt: $i \neq jgiltv_i \neq v_j$. In einem Weg darf jeder Knoten also nur maximal einmal auftreten.
- Ein Graph ist **zusammenhängend**, wenn es für jedes Paar $i, j\epsilon V$ einen Weg von i nach j gibt.

Zyklus, Länge

- k heißt **Zyklus** der Länge n, wenn k eine geschlossene Kantenfolge der Länge v_0 von nach v_n ist (d.h.: $v_0 = v_n$) und $k' = (v0, ..., vn - 1)$ ein Weg ist. Graphen ohne Zyklen heißen azyklisch.
- $|k|$ gibt die **Länge** von k an.
- Ist k eine Kantenfolge von v nach w, so gibt es immer auch einen Weg von w nach v



(1, 2, 3, 4, 5, 2, 3) ist *Kantenfolge* der Länge 6 von 1 nach 3
(1, 2, 5, 4, 2, 3) ist *Kantenzug* der Länge 5 von 1 nach 3
(1, 2, 5, 4, 3) ist *Weg* der Länge 4 von 1 nach 3
(2, 3, 4, 5, 2) ist *Zyklus* der Länge 4

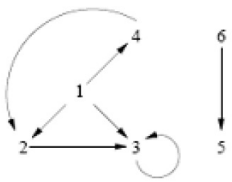
Speicherung von Graphen

Ein Graph $G = (V, E)$ mit $|V| = n$ wird üblicherweise auf eine der folgenden Arten gespeichert:

- Speicherung in einer Adjazenzmatrix
- Speicherung in einer Adjazenzliste
- Speicherung in einer doppelt verketteten Kantenliste

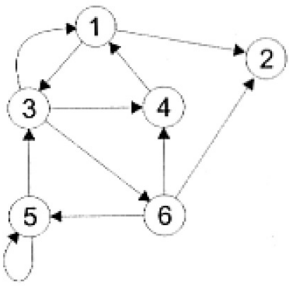
Speicherung in einer Adjazenzmatrix

- Die Matrix hat die Größe nxn. Der Eintrag an Position a_{ij} ist dabei genau dann $true$ bzw. 1, wenn die Kante (i, j) in E enthalten ist.
- Der Speicherplatzbedarf einer Adjazenzmatrix hängt nur von der Anzahl Knoten ab, nicht von der Anzahl Kanten.
- Es ergibt sich immer ein Platzbedarf von n^2.
- Bei ungerichteten Graphen ist die Matrix symmetrisch, d.h.: die Werte an Position (i, j) und (j, i) sind identisch.
- Statt einer quadratischen Matrix könnte also auch eine (untere) Dreiecksmatrix verwendet werden $\frac{n^2}{2}$ Einträgen.



	1	2	3	4	5	6
1	0	1	1	1	0	0
2	0	0	1	0	0	0
3	0	0	1	0	0	0
4	0	1	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	1	0

Graph G



Adjazenzmatrix von G

$$G_y = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Gewichtung von Kanten

Bislang war die Anordnung der Knoten beliebig wählbar. In einigen Fällen ist dies aber nicht zulässig, z.B. in den folgenden Fällen:

- Modellierung von geometrischen Punkten als Koordinaten Hier wird jedem Punkt eine feste Koordinate zugeordnet; die Anordnung der Knoten ist damit fest vorgegeben; es kann höchstens eine Skalierung der Knoten vorgenommen werden. Beispiel: Darstellung von CAD-Zeichnungen als Graph.
- Gewichtete Kanten Hierbei geht es nicht nur um die Frage "existiert eine Kante", sondern um das Gewicht der Kante (in der Regel > 0). Beispiel: Kostenfunktionen für die Verfolgung einer Kante; Verzögerungszeiten

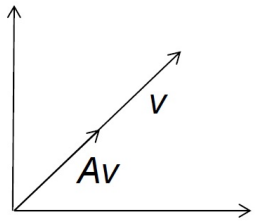
2.2.4 Eigenwerte und Eigenvektoren

Gegeben sei eine nxn-Matrix A. Ein Vektor $v \neq 0$ heißt Eigenvektor zu A, wenn A_v die gleiche oder die gegengleiche Richtung zu v hat. Als Gleichung ausgedrückt: Es gibt eine Zahl:

λ, so dass gilt

$Av = \lambda v$

Die Zahl λ nennt man den Eigenwert zu A.



2.3 Google Page Rank

- Google PageRank ermittelt die "Wichtigkeit" von Hyperlinks und ist patentiert (U.S. Patent 6,285,999)
- Patentinhaber ist die Stanford University
- PageRank ist Ergebnis des Dissertationsprojekts von Larry Page und Sergej Brin
- Standford University erhielt 1,8 Mio. Google-Aktien für die Nutzung des Patents durch Google
- Sie verkaufte die Google-Aktien im Jahr 2005 für 336 Mio. USD
- Wert am 03.12.2013: 1,4 Mrd. USD
- Grundidee: Die Websites nach Bedeutung sortieren
- Was ist "bedeutend"? Eine Website v ist umso bedeutender, je mehr bedeutende Websites u auf sie zeigen
- Henne-Ei-Problem: Um die Bedeutung von v zu berechnen, müssen wir bereits die Bedeutung von u kennen
- Formulieren wir genauer (vereinfachter PageRank-Algorithmus):
  - x_u : Bedeutung der Website n
  - N_u : Anzahl der Websites, auf die n zeigt (sog. "Outlinks" von n).
  - Jede Website u verteilt ihre Bedeutung x_n auf ihre Outlinks, und zwar zu gleichen Teilen