7.1.4 Hyperscript Structure - Technical

- If you want to produce a Hyperscript in the sense discussed in the previous sub-chapter, you need to a week-defined structure and a suitable set of rules to make it possible. If you just start writing, you will sooner or later spend most of your time searching for things and trying to find oout why nothing works the way it should. Retrospectively. I'm amazed at the powerful structure we came up with. By now it contains some 1.300 files that hold around 23.000 documents and there are no problems with interconnections, automatic functions and updating details here or there. The mastermind behind that was J. Carstensen, who also wrote all the programs needed.
  - In what follows I just give the old texts from around almost 30 years ago.
    I believ it is interesting enough to warrant archivation.
    Some links may not work.

Dateistruktur und Spielregeln

Verzeichnisstruktur

- Die Hauptverzeichnisse liegen im Server unter "amatvinz" im Ordner "hyperscripts". Im www server liegen sie direkt unter amatwww. Die Hauptverzeichnisse sind:
  - **def_en**
    Defects in Crystals
  - **elmat_en**
    Electronic Materials
  - **general_info_en**
    General Information to Hyperscripts
  - **html_en**
    HTML (for Students of Materials Science)
  - **mm_praxis**
    Multimedia in der Lehre: Hyperskripte in der Praxis
  - **mw1_ge**
    Einführung in die Materialwissenschaft I
  - **mw2_ge**
    Einführung in die Materialwissenschaft I
  - **Semiconductors**
    Semiconductors
  - **Was_heisst**
    Was heißt und zu welchem Ende studiert man Materialwissenschaft?
  - **Semiconductor_Technology**
    Part of the Lecture Course "Semiconductor Technology and Nano Electronics"
  - **Advanced Materials B**
    Part 1: Overview of Electronic, Magnetic and Optical Properties of Materials
  - **MaWi für ET IT**
    Grundlagen der Materialwissenschaft für Elektro- und Informationstechniker
  - **ISS**
    Iron, Steel and Swords
  - **Matwis_Sem**
    Materials Science Seminar
- Die Verzeichnisstruktur im Computer besteht aus einem Verzeichnis für jedes **Kapitel**, das wiederum **6** Unterverzeichnisse enthält:
  - **Basics:** Umfaßt alle Dateien die Basiswissen enthalten, d.h. Wissen das eigentlich vorausgesetzt wird. Dies mag Fachwissen sein, Mathematik oder auch Allgemeinbildung.
  - **Backbone:** Enthält die beiden "Rückgratstränge" des zu lernenden Stoffes.
  - **Illustrations:** Umfaßt alle Bilder, Grafiken, Animationen und Videos sowie die JAVA Applets die zur Bewältigung des Lernstoffes dienen.
  - **Exercise:** Enthält die Übungsaufgaben (und Lösungen) sowie e-mail Funktionen
  - **Advanced:** Umfaßt alle Dateien zu Themen, die über den Lernstoff hinausgehen
  - **Styles:** Hier liegen die Hilfsmittel zur Gestaltung von html Seiten in allen Strängen, z.B. die Hintergründe und Gliederungssymbole.

- Die **Metamodule** haben ein eigenes Verzeichnis, es enthält z.B. die Inhaltsmatrix, das Bücherverzeichnis, das Vorwort etc.. Alle für User wichtige Metamodule können vom Menu aus (über "project") abgerufen werden.
- Ein weiteres Verzeichnis namens **"check"** enthält die "Checkprogramme" und die automatisch generierten Listen.

  Für JAVA Module existiert ebenfalls ein eigenes Verzeichnis; es enthält den Quelltext und das compilierte Programm.
- Weitere Verzeichnisse nach Notwendigkeit, z.B. **"biographien"** bei **mw1_en**.

Automatische Funktionen und Register

- Das Programm **"lecture_check.exe"** führt mehrere Funktionen aus und generiert Listen für speziell markierte Wörter. Details dazu weiter unten. Es sind dies:
  - Das **Stichwortverzeichnis**. Jedes mit einem "Ziel"-Tag markierte **Wort** wird in das Stichwortregister aufgenommen
  - Das **Namensverzeichnis**. Jeder mit einem "Ziel"-Tag und vorausgehendem **! - Zeichen** markierter Name wird in das Namensverzeichnis aufgenommen
  - Das **Abkürzungsverzeichnis**. Jede mit einem "Ziel"-Tag und vorausgehendem **? - Zeichen** markierte Abkürzung wird in das Abkürzungsverzeichnis aufgenommen
  - Das **"Dictionary"** mit den Übersetzungen spezieller Wörter. Dazu mit "**§**" Zeichen englische Wörter markieren und deutsches Wort nach **§** eingeben.

- Textteile, die **nicht in ein Verzeichnis** aufgenommen werden sollen, aber als Sprungverweis dienen, werden mit einem vorausgehendem **_ - Zeichen** markiert.
- Das "Check" Programm verlinkt außerdem automatisch die Seiten und generiert div. Fehlerprotokolle, die im Haupmenu bei "**Project**" auftauchen.
  - Details dazu weiter unten.

Dokumentenbezeichnungen

- Ein absolutes Muß bei der Erstellung von **html** Seiten in allen Strängen, ist die definierte Benennung jeder Datei ("Speichern unter"...), d.h. jede bei der Erstellung des Hyperskripts angedachte Datei hat sofort einen eindeutigen Namen. Da es bei der Gesamtzahl der Dokumente völlig unmöglich ist, sinnvolle Abkürzungen zu finden, wurde **für HTML Dokumente** die folgende Systematik gewählt
- **1. Ziffer:** Buchstabe
  Frei wählbar (folgende Systematik ist empfohlen) außer bei "backbone" Texten: **Erster Buchstabe ist dann mmer r!** (für "Rückgrat")
  - **a** Animation
  - **b** Basisinformation wiss./technischer Art
  - **f** Faksimile
  - **g** Graphik
  - **i** Illustration (gleichzeitig Oberbegriff)
  - **j** Java Applet (Simulationen, Animationen, ...)
  - **l** Literatur (Bemerkungen zu Büchern etc.)
  - **m** Mathematik
  - **n** Name (Kurzbiographie)
  - **p** Photo (picture)
  - **r** Rückgrat
    Muß immer so sein!
  - **s** Lösung (solution)
  - **t** Text
  - **e** Übung (exercise)
  - **v** Video

- **2. Ziffer:** Kapitelnummer (**1 - 9**, **a=10, b=11**, ...)
- **3. Ziffer:** Unterstrich
- **4. Ziffer:** Unterkapitelnummer (**1 - 9**, **a=10**, **b=11**, ...)
- **5. Ziffer:** Unterstrich
- **6. - 8.** Nummer des Dokuments (**1-** beliebig)

- Beispiel: **r1_2_1** bezeichnet die **1.** Textdatei im Rückgrat des Kapitels **1.2**
- Bei den "Illustrations" können beliebige Namen für die **.gif** und **.jpg** Formate verwendet werden. Die zugehörigen **.doc** Formate enthalten i.f.R. die Wordgraphiken für die gleichnamigen **gifs** und sind in einem eigenene Ordner abgelegt, der nicht im Netz steht.
- Die Dateien im Verzeichnis **"Styles"** sind nicht an diese Systematik gebunden; hier sind Namen gewählt, die die Natur der Datei erkennen lassen. Zum Beispiel enthält das "Style" Verzeichnis in jedem Kapitel die Datei "dreieck.gif"; dies sind die in den Kapitelfarben ausgeführten Dreiecke der Absatzmarkierung.
- Die Datei "Metamodule" hat keine besondere Feinstruktur oder Regeln

Stilelemente

- **Textstrukturierung**
  - Überschriften sind in drei Ebenen ausgeführt und in Schriftgröße und Farbe automatische abgehoben (**H1 - H3**). Weiter Überschriften (**4.** Ebene), rot/bold/zentriert wo nötig.
  - Das Dreieck wird grundsätzlich als Absatzmarkierung verwendet. Farbe etc. ist in "styles" definiert.
  - Ein weitere Untergliederungsebene innerhalb eines Absatzes wird mit Kugeln markiert. Farbe etc. ist in "styles" definiert.
  - Wörter die "**Index**" auftauchen sind schwarz und fett, bei "**Namen**" rot und fett. Wörter im ***Dictionary*** sind rot, kursiv und fett; die Übersetzung wird eingeblendet falls der Kursor auf das Wort zeigt. Rote Fonts werden häufig zur *Betonung* verwendet; i.d.R. dann gekoppelt mit kursiv.
  - *Redaktionelle Hinweise* (z. B. zur Schreibweise von Formeln) sind oft in blau und kursiv.
  - Weitere Strukturierungen, z.B. Aufzählungen, Hervorhebungen etc. sind nach Belieben gestaltet. Merksätze, Links zu Übungen etc. sind häufig eingerahmt und farbig hinterlegt.
  - Die Merkpunkte folgen einem eigenen zweispaltigen Schema.
- **Formeln**
  - Formeln sind in **html** nicht bequem darstellbar. Die übliche Einbindung als **gif**-Bild ist zeitaufwendig und nur schwer zu ändern. In diesem Hyperskript werden deshalb praktisch alle Formel so gut es geht in **html** erstellt, unter extensiver Nutzung des "FONT=SYMBOL" Attribute. Dass das ganz gut funktioniert (mit dem richtigen Browser) demonstriert der Link.
- **Bilder und Grafiken in Text**
  - Die in den Text eingebundenen Bilder und Grafiken sind abgesetzt und mit Stärke **2** umrandet, sie sind fallweise als Bestandteil des Textes zu verstehen und tragen dann keine Bildunterschrift oder stärker hervorgehoben und eigenen Unterschriften oder Kommentaren. Die Einbindung einer Vielzahl von Illustrationen bedingt allerdings einen Verzicht an graphischer Perfektion.
- **Farben und Hintergründe**
  - Jedes Kapitel hat eine eigene Farbe; sie findet sich (pastell) im Hintergrund, in den Randleisten aller Stränge /außer Rückgrat **I** und Rückgrat **II**), in den Farben der Dreiecke und in den Farben der Markierungskugeln. Alle Farben werden über die Style sheets definiert.
- **Menu**
  - Alle Hyperskripte haben alle eine (abschaltbare) Menuleiste mit direktem Zugriff auf Kapitel und Unterkapitel und Umblätterfunktionen.

Automatisches Erstellen von Index-Listen

Es gibt zwei Varianten des Check-Programms:

1. **Lecture_Check**
2. **Mawint_Check**

Die erste Variante ist für das automatische Bearbeiten von Vorlesungs-Skripten geeignet. Sie verbindet die Dokumente des "Backbones" miteinander und stellt Verbindungen mit Sprungmöglichkeiten zwischen den Dokumenten her. Sie erstellt die unten beschriebenen Listen, wandelt alle relativen Referenzen und Dateinamen in Kleinbuchstaben um (wichtig für die Kompatibilität mit UNIX-Servern), überprüft Bilder und Links und nimmt automatisch verschiedene stilistische Anpassungen vor.

Die zweite Variante ist zum automatische Bearbeiten nicht so streng strukturierter HTML-Projekte geeignet. Sie kann somit nicht automatisch (sinnvolle) Verknüpfungen zwischen Dokumenten herstellen. Da solche Projekte auch aktuelle Daten enthalten können, die kurzfristig geändert werden müssen, bietet diese Variante deshalb die Möglichkeit zur automatischen Organisation solcher Referenzen (siehe unten). Ansonsten werden die gleichen stilistischen Checks wie im obigen Programm durchgeführt und dieselben Listen generiert.

1. Automatisches Erstellen einer Liste aktueller Änderungen

Ziel

Erleichtertes Projektmanagement und Benutzung durch auffällige Anzeige (Blinken) von wichtigen oder aktuellen Änderungen eines Textes.

Durchführung

1. Markieren als Anker (Link, F6, Class eintragen)
2. Setzen der Class "Bli1"
3. Definition eines Namens nach nachfolgender Namenskonvention:

- Beginn mit $-Zeichen
- 6 Ziffern für aktuelles Datum: Beispiel 050799
- Nummerierung innerhalb eines Textes durch einen Buchstaben (dies sollte reichen für eine Datei pro Tag)
- Optional: Nachfolgende Buchstaben (*nicht durch Unterstrich getrennt*) werden als Kommentar interpretiert
- Optional: Weiterer Text (mehr als ein Wort) durch Unterstriche trennen.

Beispiel: $050799aKommentar_NormalerName bedeutet: Änderung am 5.7.99 durchgeführt; Anderung Nr. 1 an diesem Tag; Als Kommentar erscheint "Kommentar Normaler Name"

Automatische Bearbeitung des Ankers durch das Check-Programm

1. Alle so markierten Anker werden in eine Liste aufgenommen (mit Datumsanzeige und Dateianzeige), von der direkt auf einzelne Änderungen gesprungen werden kann
2. Der Kommentar wird in die 3. Spalte eingetragen.
3. Nach einer *vom Benutzer vorgegebenen Zeit* wird die Hervorhebung wieder entfernt. Dies geschieht unterschiedlich, je nach Wahl des Anker-Namens
4. Besteht der Anker nur als *Kurzversion* (z.B. $050799a oder $050799aKommentar, so wird der Anker mit Class und Name gelöscht. Der ursprünglich markierte Text erscheint nun normal.
5. Handelte es sich dagegen um einen *langen Anker-Namen* (z.B. $050799aKommentar_NormalerName), dann wird dieser Name gekürzt auf den restlichen Text (z.B. hier NormalerName) und bleibt als An- und Absprungmarke erhalten
6. Da auch in allen Referenzen die Namensänderung automatisch zur gleichen Zeit erfolgt, bleiben alle Anker gültig
7. Der entsprechende Anker erscheint dann nicht mehr in der Änderungs-Liste. Falls der restliche Name jedoch mit einem der geschützten Zeichen beginnt, wird er dann in die entsprechende Liste aufgenommen.

2. Automatisches Erstellen einer Index-Liste

Ziel

Alle Anker innerhalb des Projektes, die mit dem "NAME"-Attribut versehen sind, werden alphabetisch sortiert in einer Liste dargestellt. Existiert ein "HREF"-Attribut, so wird dieses als URL für den Listeneintrag verwendet

Durchführung

Ein beliebiger Name mit wenigen Ausnahmen (siehe unten). Soll ein Anker in keiner Liste erscheinen, so muß er mit einem Unterstrich beginnen (z.B. _Zitat1)

3. Automatisches Erstellen einer Namens-Liste

Ziel

- Namen, die innerhalb eines Projektes vorkommen, können in einer Liste alphabetisch sortiert dargestellt werden. Die Dateien, in denen diese Namen vorkommen, werden durchnummeriert und können angesprungen werden. Existiert zu einem Namen ein Link (z.B. auf eine Homepage) so kann über den Namen auf diese URL zugegriffen werden

Durchführung

- Der Name des Ankers muß mit einem "!" beginnen (z.B. !Föll)

4. Automatisches Erstellen einer Abkürzungsliste

Ziel

- Abkürzungen, die im Projekt vorkommen, werden in einer Liste alphabetisch sortiert und mit einer Erläuterung versehen. Existiert zu einer Abkürzung ein Link so kann über die Abkürzung auf diese URL zugegriffen werden

Durchführung

- Der durch den Anker markierte Text wird als Abkürzung interpretiert (z.B. RLZ=Raumladungszone" . Der Name des Ankers auf RLZ muß mit einem "?" beginnen; Eintrag also ?Raumladungszone . Der nach ? folgende Text stellt dann die Erläuterung zur Abkürzung dar.