



# ***Web-basierte Anwendungen***

*Studiengänge AI (4140) & WI (4120)*



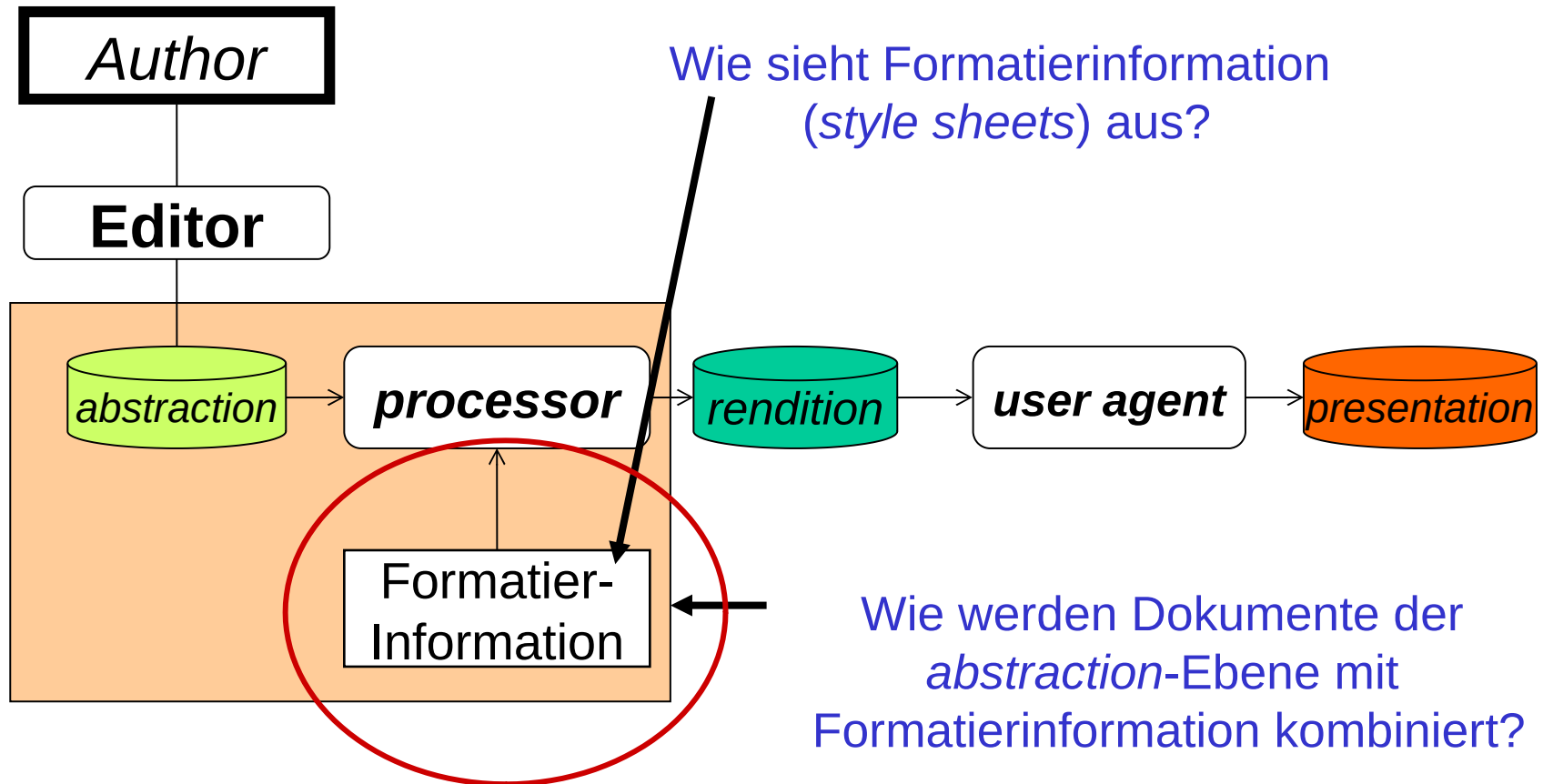
# ***Style Sheets (Teil 1)***

Einbettung von *Style Sheets*  
CSS2 - Eine kleine Einführung

# \* Abstraction – Rendition - Presentation

(Erinnerung)

Offene Fragen:





# CSS2 - Eine kleine Einführung

*Cascading Style Sheets Level 2*  
am Beispiel XML

(Einzelheiten: Siehe Originalspezifikationen!)



# CSS2: Vorgeschichte

---

- **HTML**

- HTML entstand ohne klare Trennung zwischen *abstraction* und *rendition*, aber mit Schwerpunkt auf der *abstraction*-Seite.
- Die stürmische Entwicklung des WWW außerhalb des akademischen Ursprungs rückte rasch die Frage in den Vordergrund, wie Inhalte dargestellt werden.
- Der so motivierte Bedarf nach HTML-Erweiterungen fachte einen „Browser-Krieg“ an, insb. zwischen Netscape und Microsoft. Beide Browser-Hersteller entwickelten proprietäre, zueinander inkompatible HTML-Erweiterungen.
- Das W3C reagierte mit der Spezifikation der CSS (Level 1).



# CSS2: Vorgeschichte

---

- **HTML (Forts.)**
  - CSS sollte idealerweise komplett die Darstellung übernehmen, HTML-Code sollte sich auf die *abstraction*-Ebene konzentrieren.
  - Dem waren allerdings Grenzen gesetzt durch
    - inzwischen standardisierte HTML-Erweiterungen (bis HTML 4.0)
    - die konzeptionell unklare Trennung von *abstraction* und *rendition* in HTML
    - die Beschränkungen von HTML auf der *abstraction*-Ebene, welche letztlich erst von XML überwunden werden.



# CSS2: Vorgeschichte

---

- **CSS**

- **1996: CSS1**. Dieser erste - noch relativ einfache - CSS-Level fand lange Zeit nur geringe bis mäßige Unterstützung durch die Browser-Hersteller.
- Web-Entwickler, die zur Vermeidung von Browser-Inkompatibilitäten CSS einsetzten, gerieten so vom Regen in die Traufe.
- Dies schadete dem Ruf von CSS - trotz seines bestechenden Konzepts - und erschwerte seine Verbreitung.
- **1998: CSS2**. Damit wurde bereits ein erheblich komplexerer Nachfolger als Standard spezifiziert, der weitgehend - aber nicht vollkommen - abwärtskompatibel zu CSS1 ist. Zu dieser Zeit war selbst CSS1 noch nicht hinreichend verbreitet.



# CSS2: Vorgeschichte

---

- **CSS (Forts.)**
  - CSS 2.1 (Rec, 07. Juni 2011, Update 12.4.2016)
    - Quelle: <http://www.w3.org/TR/CSS21/> (Kern-Modul, s.u.)
    - Korrektur einiger Fehler in CSS2
    - Aufnahme einiger ursprünglich für CSS3 geplanter Möglichkeiten
    - **Noch aktueller Stand und Basis für weitere Entwicklungen!**
    - Weitere Module: „CSS Style Attributes“, „CSS Namespaces“, „Media Queries Level 3“, „Selectors Level 3“, „CSS Color Level 3“
      - Modul-Level entwickeln sich unabhängig voneinander weiter!
  - CSS3, CSS4 ?
    - Gibt es streng genommen nicht – die o.g. Module entwickeln sich nun unabhängig voneinander weiter
    - Etliche Module haben bereits „Level 3“ erreicht, einige bereits „Level 4“
  - CSS-Profile: *Mobile 2.0, Print 1.0, TV 1.0*
    - Nicht alle Anwendungen benötigen alle CSS-Fähigkeiten!
  - Zu CSS allgemein (aktueller Stand)
    - <http://www.w3.org/TR/CSS/>





# CSS2: Vorgeschichte

---

- Status
  - Seit ca. 2009 hat sich die Situation stark verbessert:
    - Aktuelle Browser mit XML-Unterstützung implementieren nun auch CSS2.1 praktisch komplett, auch viele CSS3-Eigenschaften
    - Standard-Tests sind verfügbar → Acid-Tests:
      - <http://acid2.acidtests.org/>
      - [http://www.philippbauer.de/info/webdesign/webstandards/#browser\\_im\\_acid2\\_test](http://www.philippbauer.de/info/webdesign/webstandards/#browser_im_acid2_test)
      - <http://acid3.acidtests.org/>
  - Allerdings gibt es auch heute (2021) noch Lücken:
    - Firefox, aktuelle Version 88.0: Acid2-Test bestanden, Acid3: 97%
    - Chromium 90.0.x: Acid2-Test nicht bestanden, Acid3: 97%
    - Literatur: c't 9/2008, S. 140 ff.



# CSS: Einbindung

---

- **Inline**

- Beispiel: `<p style="align: center"> ... </p>`
- In seltenen Fällen und stets für kurze Anweisungen berechtigt
- **In der Regel vermeiden** – erschwert die Arbeitsteilung zwischen HTML-Autor und Layouter/Designer

- **Im HTML-Kopf**

`<head>`

`<style type="text/css" media="all">`

```
@import "style/all.css"; /* CSS-Einbindung auf CSS-Ebene */
body {background-color: lightgray;} /* + lokale Regeln */
h1 {align: center;}
```

`</style>`

`</head>`

- Spart eine externe CSS-Datei, aber **schlecht für einheitliches Design**

- **In externen CSS-Dateien**, importiert im HTML-Kopf

`<link rel="stylesheet" type="text/css" href="all.css" media="all">`

- **Der Regelfall!**

# \* CSS2 – mehr als nur ein HTML-Zusatz!

- CSS kann nicht nur HTML-Elemente gestalten, sondern auch die Elemente jeder XML-basierten Auszeichnungssprache!
- Prinzip
  - Element(e) selektieren
  - Eigenschaften (der selektieren Elemente) deklarieren
- Beispiel SVG:

```
line {  
  stroke: black;  
  stroke-width: 2px;  
}  
rect {  
  stroke: black;  
  fill: lightblue;  
}
```



```
<?xml version="1.0" encoding="UTF-8"?>  
<svg xmlns="http://www.w3.org/2000/svg"  
  version="1.1" width="800" height="600">  
  <line x1="0" y1="10" x2="799" y2="10"/>  
  <rect x="0" y="299" width="800" height="300"/>  
</svg>
```

# CSS2 – mehr als nur ein HTML-Zusatz!

---

- CSS diene auch als Vorbild für andere Web-Technologien
- Beispiel JavaScript
  - Die weit verbreitete JS-Bibliothek **jQuery** verwendet CSS-artige Selektoren statt des umständlicheren DOM-APIs
- Beispiel HAML
  - Kurzschreibweisen von CSS für id- und class-Attribute wurden übernommen
- Beispiel XSL-FO
  - (FO=„Formatting Objects“, eine Seitenbeschreibungssprache)
  - Viele Modell-Annahmen zum Seitenaufbau sowie Eigenschaftswerte sind gleich, CSS und FO wurden aufeinander abgestimmt entwickelt



# CSS2: Ein XML-Beispiel

---

- XML-Fragment (aus: Tutorial der CSS2-Spezifikation, Kap. 2.2):

```
<?xml-stylesheet type="text/css" href="bach.css"?>
```

```
<ARTICLE>
```

```
<HEADLINE>Fredrick the Great meets Bach</HEADLINE>
```

```
<AUTHOR>Johann Nikolaus Forkel</AUTHOR>
```

```
<PARA>
```

```
  One evening, just as he was getting his
```

```
  <INSTRUMENT>flute</INSTRUMENT> ready and his musicians were  
  assembled, an officer brought him a list of the strangers who  
  had arrived.
```

```
</PARA>
```

```
</ARTICLE>
```

- Ein passendes CSS-Dokument dazu:

```
INSTRUMENT          { display: inline }
```

```
ARTICLE, HEADLINE, AUTHOR, PARA { display: block }
```

```
HEADLINE            { font-size: 1.3em }
```

```
AUTHOR              { font-style: italic }
```

```
ARTICLE, HEADLINE, AUTHOR, PARA { margin: 0.5em }
```

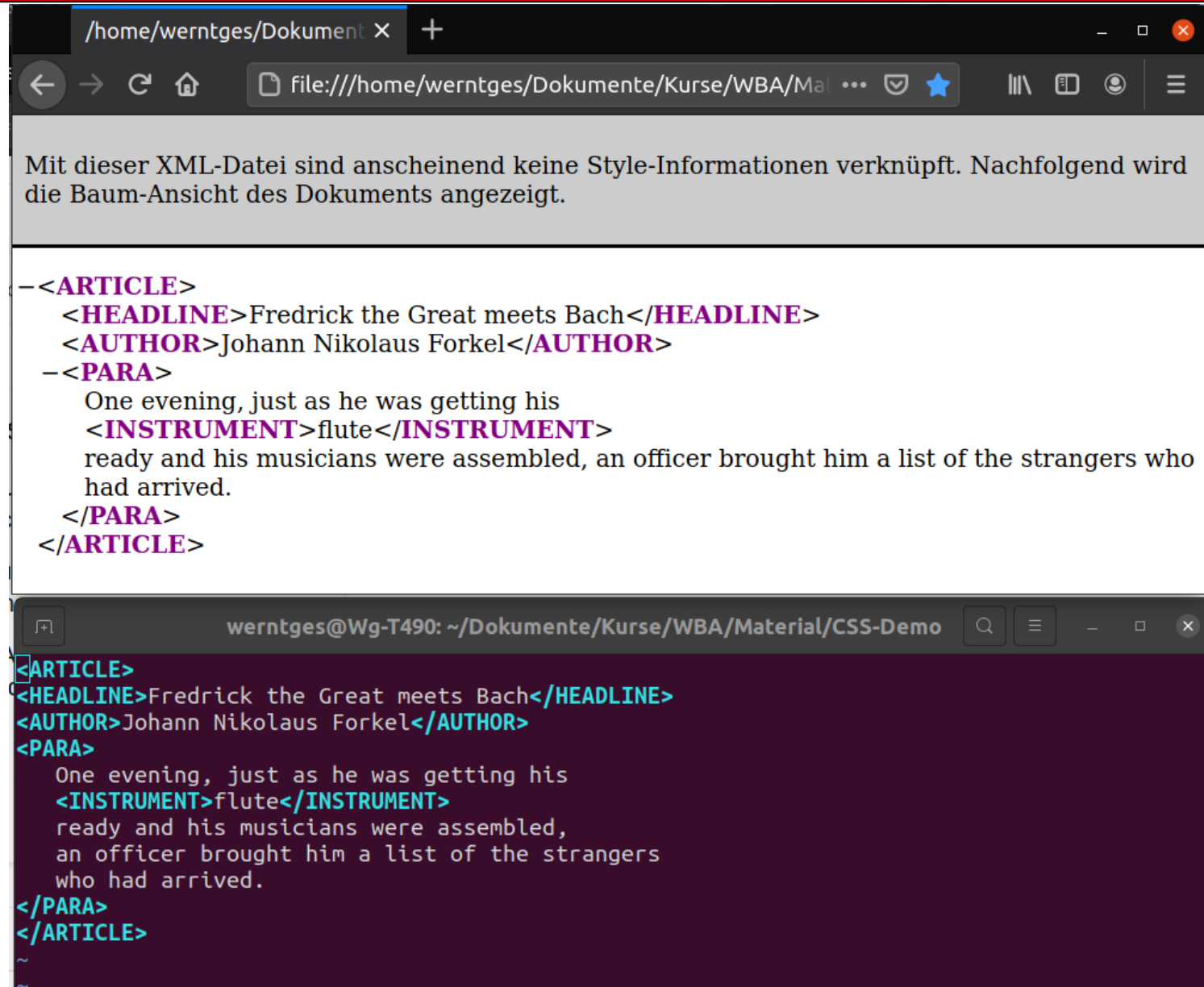


# CSS2-Demo

---

- bach0 - XML ohne CSS:
  - Browserspezifische Darstellung, bei IE & Firefox baumartig.
- bach1 - XML mit leerer CSS-Datei:
  - Nur die unformatierten Nutzdaten, „*inline style*“
- bach2 - XML mit CSS-Datei gemäß Beispiel:
  - Brauchbar, aber:  
Unterschiede im Detail zur Wiedergabe in den Spezifikationen
- bach3 - Variationen in CSS:
  - Hintergrundfarbe
  - Font: Helvetica, sans-serif
  - Breite fest vorgegeben, Titel und Autor zentriert
  - 3 Textattribute für „Instrument“.
- Browser-Unterschiede Mozilla Firefox – andere ?

# \* CSS2-Demo: Fall "bach0"



The image shows two windows side-by-side. The top window is a web browser with the address bar showing a file path. The main content area displays a message about missing style information and an XML document tree. The bottom window is a terminal with a dark background, showing the same XML content with syntax highlighting.

Browser address bar: `/home/werntges/Dokument`

Browser address bar: `file:///home/werntges/Dokumente/Kurse/WBA/Ma`

Message: Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft. Nachfolgend wird die Baum-Ansicht des Dokuments angezeigt.

XML Document Tree:

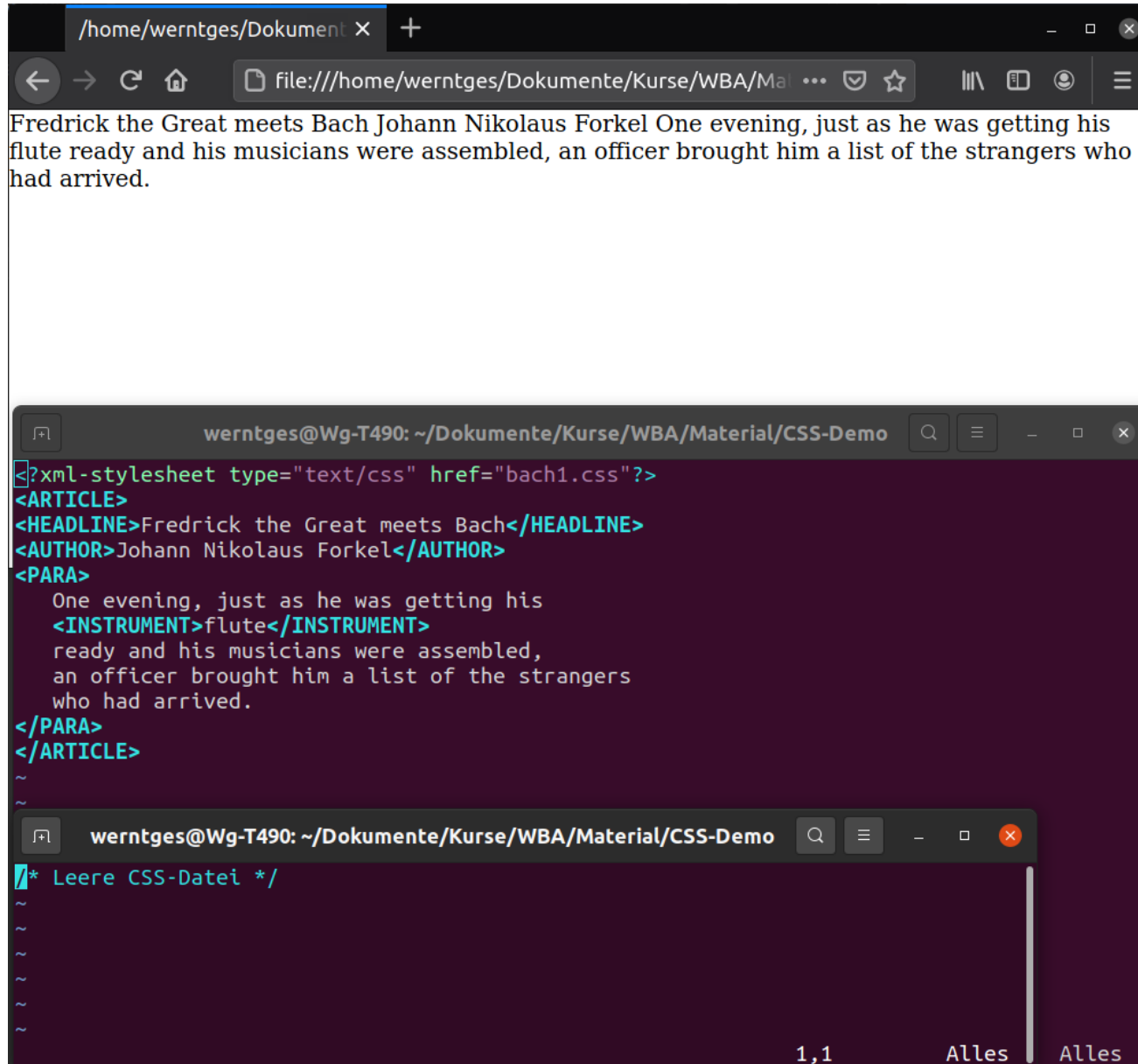
- <ARTICLE>
  - <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
  - <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
  - <PARA>
    - One evening, just as he was getting his
    - <INSTRUMENT>flute</INSTRUMENT>
    - ready and his musicians were assembled, an officer brought him a list of the strangers who had arrived.
- </PARA>
- </ARTICLE>

Terminal window title: `werntges@Wg-T490: ~/Dokumente/Kurse/WBA/Material/CSS-Demo`

Terminal content:

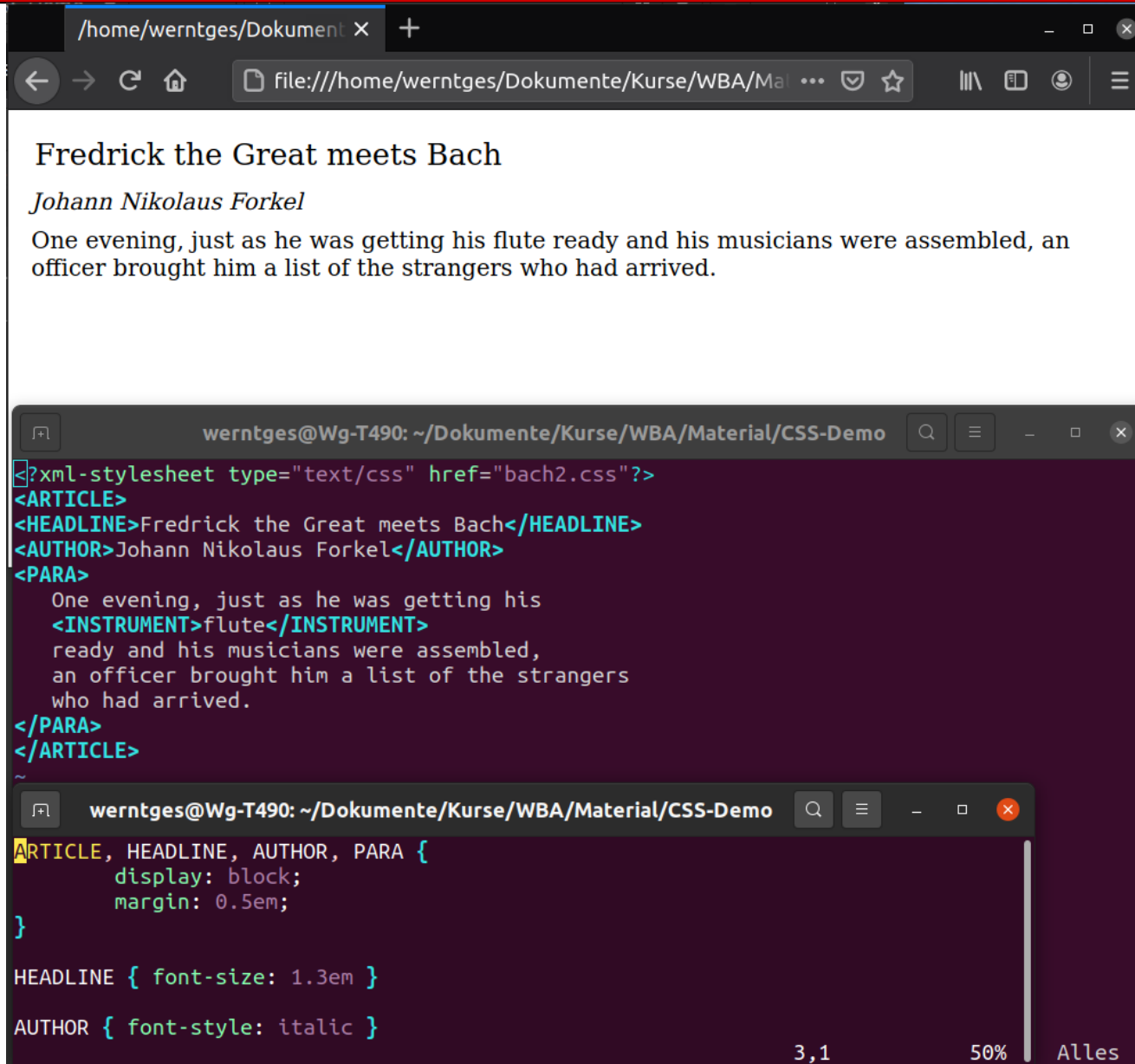
```
<ARTICLE>
<HEADLINE>Fredrick the Great meets Bach</HEADLINE>
<AUTHOR>Johann Nikolaus Forkel</AUTHOR>
<PARA>
  One evening, just as he was getting his
  <INSTRUMENT>flute</INSTRUMENT>
  ready and his musicians were assembled,
  an officer brought him a list of the strangers
  who had arrived.
</PARA>
</ARTICLE>
~
~
```

# \* CSS2-Demo: Fall “bach1”

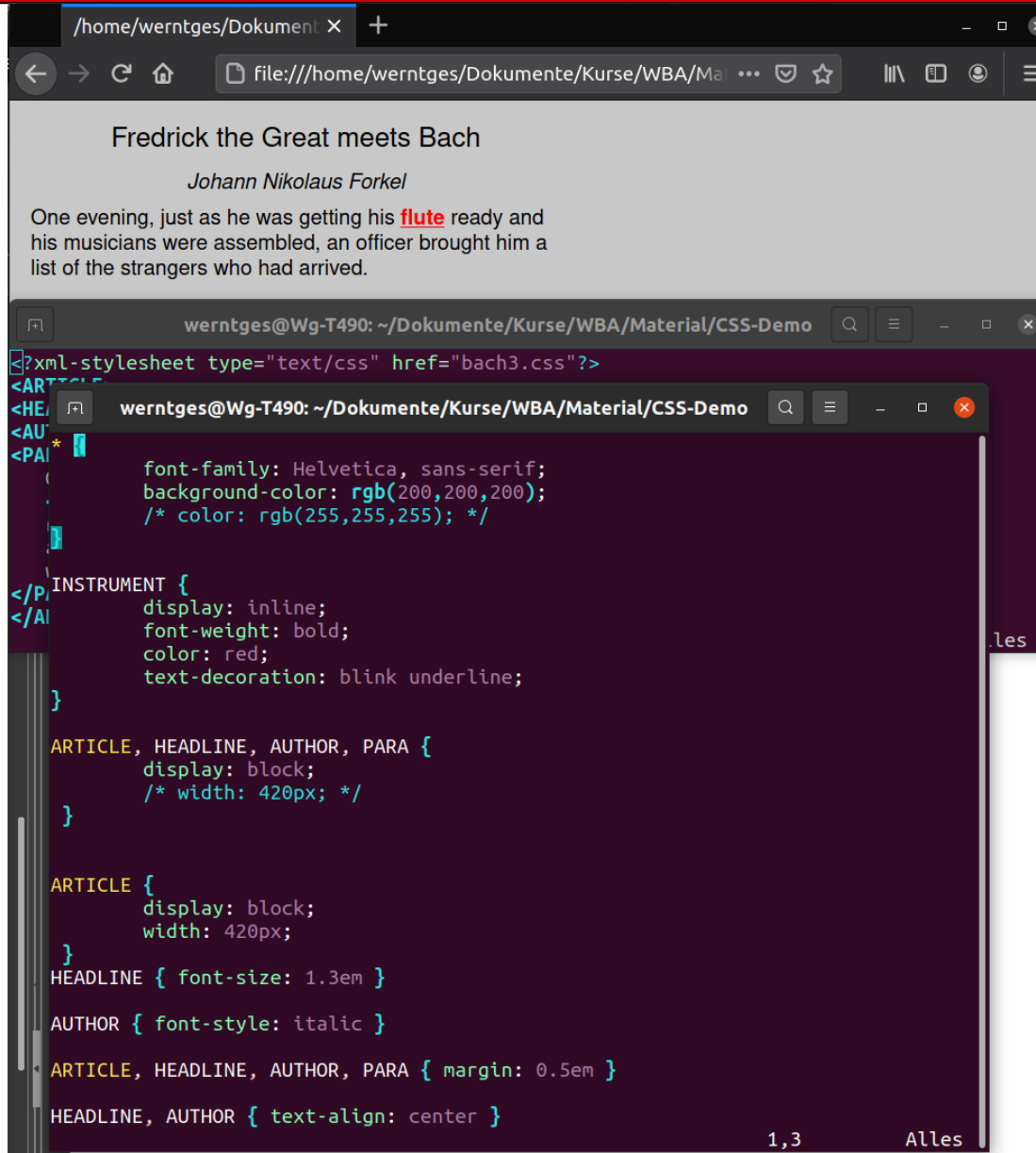




# \* CSS2-Demo: Fall “bach2”



# \* CSS2-Demo: Fall “bach3”





# CSS2: Ein XML-Beispiel

---

- Die Anzeige der XML-Datei im Browser laut W3C:
  - Code laut Beispiel “bach2”
  - Tatsächliche Darstellung erforderte mehr Code, vgl. „bach3“

## Fredrick the Great meets Bach

*Johann Nikolaus Forkel*

One evening, just as he was getting his flute ready and his musicians were assembled, an officer brought him a list of the strangers who had arrived.

# CSS2: Aufbau

---

- Ein CSS *style sheet* besteht aus einer Folge von *statements*
  - Es gibt zwei Arten von *statements* (Anweisungen):
    - *at-rules* (@-Regeln)
    - *rule sets* bzw. *rules* (Regelmengen, Regeln)
- *at-rules*
  - Grammatik:
    - `at-rule ::= '@' identifier S* ( [^;] ';' | block )`
  - Beispiele:
    - `@import "subs.css"`
    - `@media print { BODY { font-size: 10pt; } }`
- *rule sets / rules*
  - Grammatik:
    - `rule ::= selector block`
  - Beispiel:
    - `H1, H2 { color: green; }`

# \* CSS2: *charset*

---

- Situation
  - CSS2-Dateien werden analog zu XML-Dateien mit Editoren erzeugt und von UAs gelesen, sodass Missverständnisse in Bezug auf den verwendeten Zeichensatz bzw. dessen Codierung entstehen können
- Frage
  - Was entspricht dem „encoding“-Teil der XML-Deklaration in CSS-Dateien?
- Antwort
  - Die **@-Regel „charset“**, die ganz am Anfang stehen muss. Beispiel:  
`@charset "ISO-8859-15";`
- Hinweise
  - Es gibt weitere Möglichkeiten, das *Encoding* zu bestimmen, Einzelheiten dazu in Kap. 4.4 der CSS2.1-Spezifikation
  - **Wenn nichts anderes gilt, geht ein UA von UTF-8 aus!**

# \* CSS2: *media types, media groups*

---

- Mittels der in CSS2 aufgenommenen **at-rule** `@media` ist es möglich, Regeln für verschiedene Medien parallel und sauber getrennt in einer CSS-Datei zu pflegen.
- *media types*
  - *all, braille, embossed, handheld, print, projection, screen, speech, tty, tv*
- *media groups*
  - *continuous/paged, visual/audio/speech/tactile, grid/bitmap, interactive/static*
- Matrix der Zuordnungen zwischen *media types* und *media groups*
  - Siehe 7.3.1

- Beispiel (aus: 7.2.1):

```
@media print { body { font-size: 10pt } }
```

```
@media screen { body { font-size: 12pt } }
```

```
@media screen, print { body { line-height: 1.2 } }
```

# \* CSS2: *selector*

---

- *selector*
  - Im einfachsten Fall der Elementname, dem bestimmte Eigenschaften im folgenden *declaration block* zugewiesen werden sollen.

- Grammatik

```
selector ::= ( type_selector | universal_selector )  
( attribute_selector | ID selector | pseudo_class )*
```

- Beispiele

**\*** passt zu jedem Element („*universal selector*“)

**E** wählt jedes Element E aus

**E F** wählt F aus, wenn es von einem E abstammt (*descendant*)

**E > F** ..., wenn F ein direktes Unterelement (*child*) von E ist

**E + F** ..., wenn F direkt auf ein Element E folgt

**E[foo]** wählt E aus, wenn es ein gesetztes Attribut *foo* besitzt

**E[foo="val"]** ..., wenn sein Attribut *foo* den Wert "val" besitzt

**E:first-child** ..., wenn es das erste *child element* seines *parent* ist

Beispiel: **li:first-child** zur Gestaltung des ersten Elements einer Liste

# \* CSS2: *selector*

---

- Weitere Beispiele
  - Besuch des [Kapitels 5](#) der (gut lesbaren!) CSS2.1-Spezifikation
- *selector grouping*
  - Mehreren *selectors* kann auf einfache Weise derselbe *declaration block* zugewiesen werden. Dazu listet man sie einfach komma-separiert auf.
  - Beispiel: **H1, H2, H3, P { ... }**
- Bemerkungen
  - Es gibt noch zahlreiche Details allein zu *selectors* zu beachten, die aber nichts grundsätzlich Neues hergeben und daher keinen Eingang in die Vorlesungsfolien finden.
  - Einzelheiten lesen Sie bitte direkt in Kapitel 5 („Selectors“) der CSS2-Spezifikation nach!





# CSS2: Vererbung

---

- Vererbung:
  - „Normalerweise“ gemäß Dokumentenbaum-Struktur
  - Jede Eigenschaft definiert, ob sie vererbt wird oder nicht
  - Eigenschaftswert „inherit“. Beispiel: CSS2-6.2.1
- Beispiel:
  - Die Eigenschaft „color“ wird von einem Block-Element an abhängige inline-Blockelemente vererbt.
    - `body { color: blue }` /\* auch Text in „p“ betroffen \*/
  - Die Eigenschaften „border-color“, „border-width“, „border-style“ werden nicht geerbt.
    - `body { border-color: black }` /\* „p“ nicht betroffen, Demo \*/
  - Vererbung erreichbar durch Eigenschaftswert „inherit“ im Ziel-Element:
    - `p { border-color: inherit }` /\* „p“ nun auch betroffen! \*/



# CSS2: „cascading“

---

- *Cascading*:
  - Drei Quellen konkurrieren um Auswirkung:  
*user agent defaults, user settings/style sheet, author style sheet*
  - Die „Kaskade“ der Wichtigkeit, in steigender Reihenfolge:
    - 1. *User agent settings*
    - 2. *User settings, normal*
    - 3. *Author style sheets, normal*
    - 4. *Author style sheets, important*
    - 5. *User settings, important*

Nur innerhalb von *author style sheets* sinnvoll.
  - Steuerung mittels Schlüsselwort „!important“:
    - **p { font-size: 18pt !important }** im Stylesheet des Autors der Website kennzeichnet wichtige Eigenschaften und hat Vorrang vor normalen Benutzereinstellungen für Zeichensatzgrößen im Browser. Der Benutzer kann dies notfalls seinerseits überschreiben (wenn der *user agent* dies unterstützt...) !

# \* CSS2: „*cascading*“

---

- *Cascading* (Forts.):
  - Sortierung nach Spezifität:  
Spezifischere Selektoren haben Vorrang vor allgemeinen.
  - Beispiele:
    - E** > **F**      hat Vorrang vor **F**,
    - F**              hat Vorrang vor \*
  - Sortierung nach Reihenfolge
    - Bei ansonsten gleichen Bedingungen „gewinnt“ die letzte Angabe
    - Importierte Angaben sind grundsätzlich nachrangig zu lokalen.

# \* CSS2: *declaration blocks*

---

- Eigenschaften
  - Jede Eigenschaft enthält in der Spezifikation eine einheitliche Beschreibung mit folgenden Angaben:
    - *Value*: Liste der zulässigen Werte oder „inherit“.
    - *Initial*: Angenommener Wert, wenn Eigenschaft nicht explizit definiert
    - *Applies to*: Beschreibung der betroffenen Elemente
    - *Inherited*: Wurde diese Eigenschaft vom Eltern-Element geerbt?
    - *Percentages*: Ggf. Angabe, auf was sich Prozent-Angaben beziehen
    - *Media*: Von dieser Eigenschaft betroffene Medientypen
    - *Computed value*: Ggf. Regeln zur Berechnung des Eigenschaftswertes
  - Beispiel '**color**'
    - *Value*: `<color> | inherit`
    - *Initial*: `depends on user agent`
    - *Applies to*: `all elements`
    - *Inherited*: `yes`
    - *Percentages*: `N/A`
    - *Media*: `visual`
    - *Computed value*: `as specified`

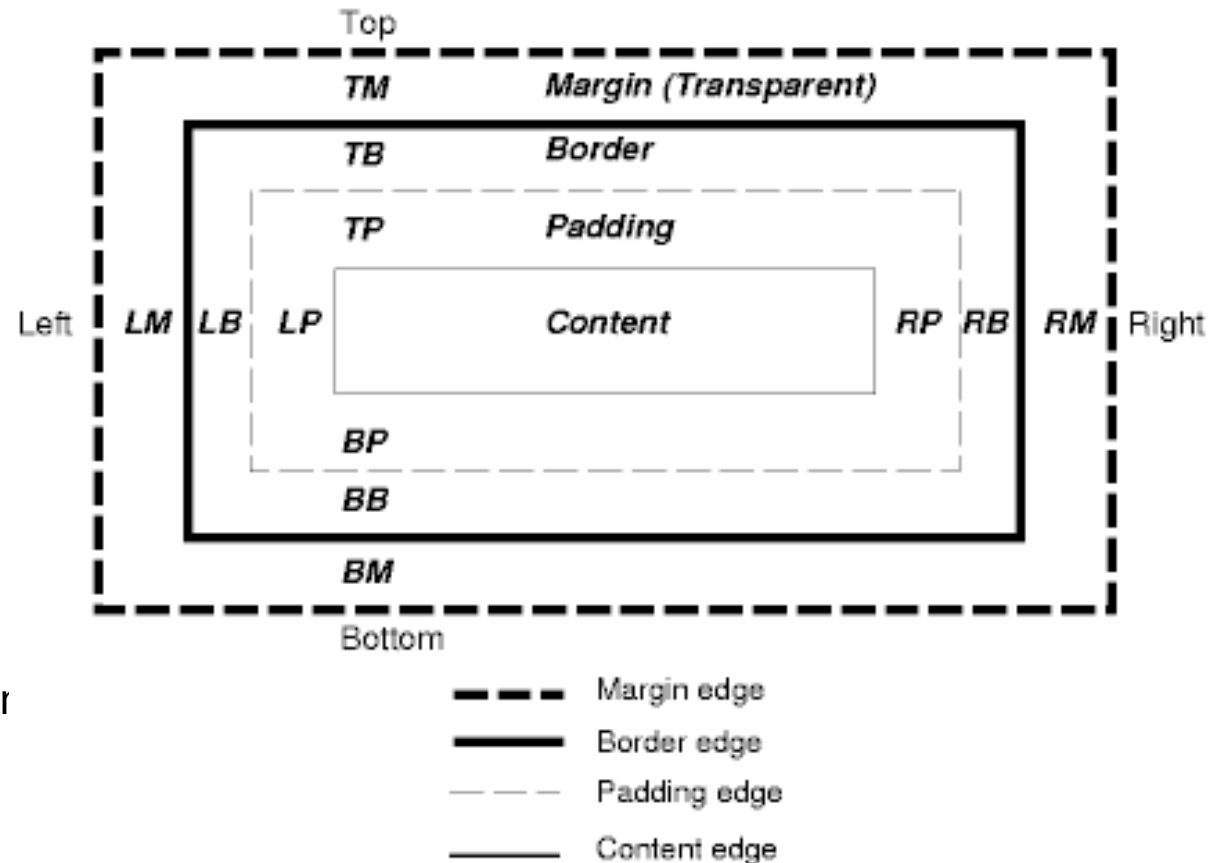
# **CSS2: *declaration blocks***

---

- (mündliche Kommentare)
- (Basiswissen erschließt sich leicht aus den Beispielen)
- (die zahlreichen Beispiele ggf. in den Spezifikationen nachlesen)
- Falls Zeit: Demo (W3Schools)

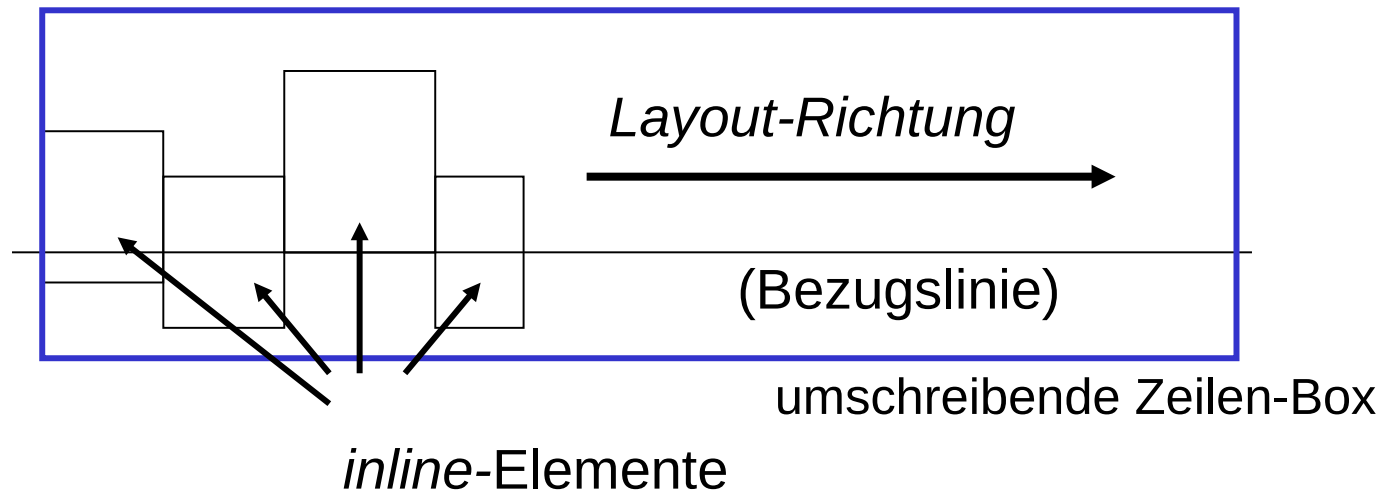
# \* CSS2: Box-Modell

- Elemente des Dokumentenbaums werden in rechteckige Kästen (*boxes*) umgewandelt
- Jeder Kasten besitzt einen Inhalt (*content*) sowie optionale Umgebung wie folgt:
  - Polsterung (*padding*),
  - Rahmen (*border*),
  - Randbereiche (*margin*)
- *Block vs. Inline*
  - Vertikale vs. horizontale Aneinanderreihung der Kästen!
  - Richtung u.U. sprachabhär Vgl. XSL-FO



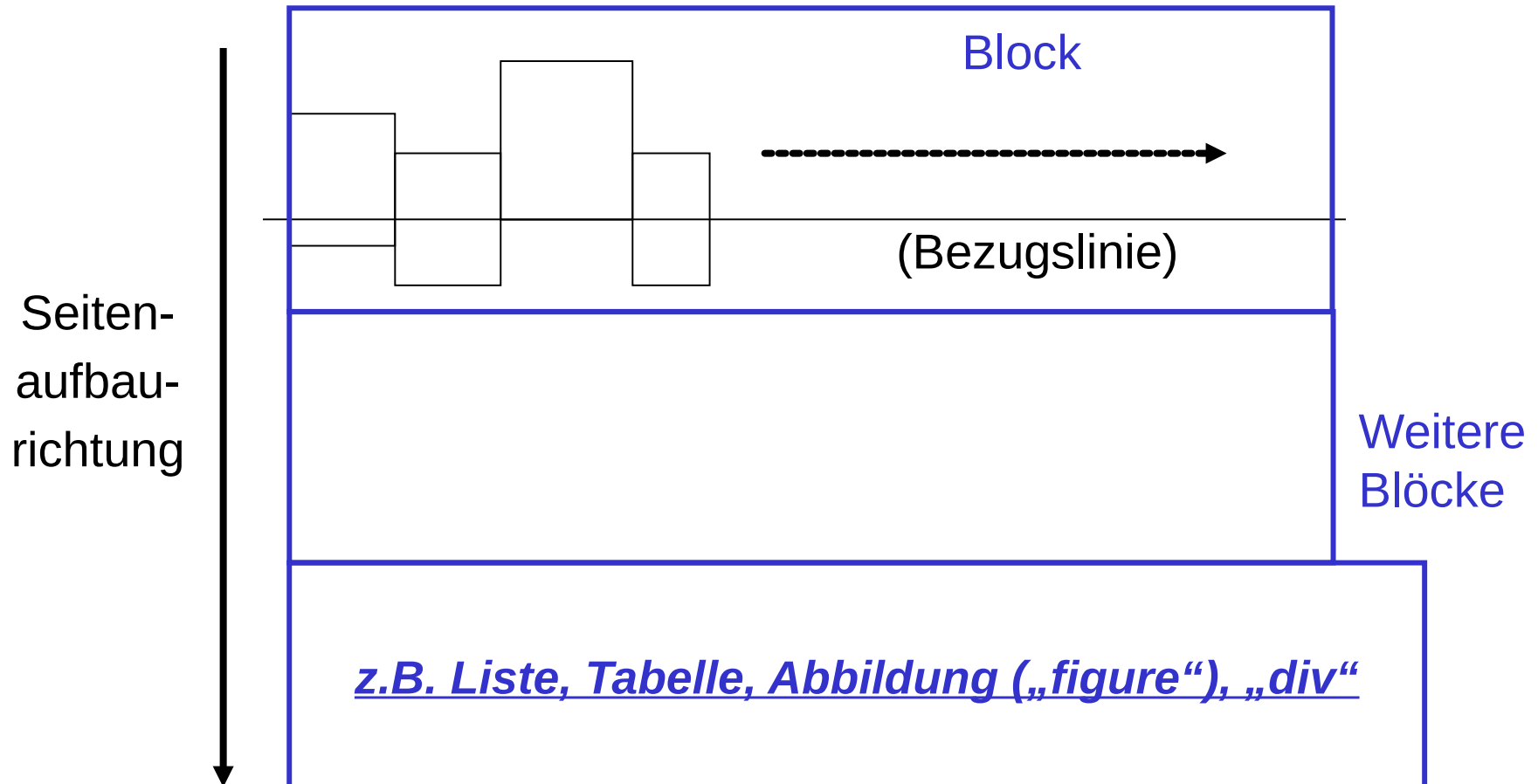
# \* CSS2: Layouts

- *Inline formatting context*
  - Zeichen u.a. Einheiten werden als „Kästchen“ (Blöcke) aneinandergereiht, meist von links nach rechts („ltr“), wie Typensetzer beim Buchdruck eine Zeile von Lettern bilden
  - Relative Positionierung: Einzelne Blöcke können mit den Eigenschaften **left**, **right**, **top**, **bottom** verschoben werden. Ihre Lücken werden nicht aufgefüllt. Sie können durch die Verschiebung andere Seitenteile verdecken.



# \* CSS2: Layouts

- *Block formatting context*
  - Paragraphen, Tabellen, Listen bilden Blöcke, die in diesem Kontext „montiert“ werden – normalerweise linksbündig von oben nach unten



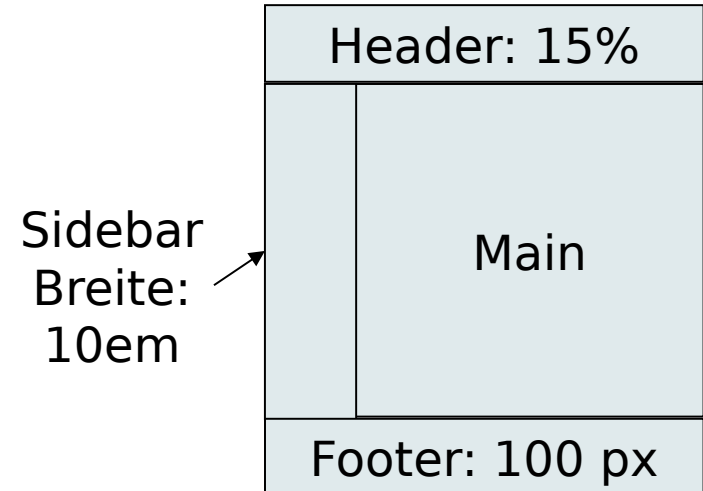


# \* CSS2: Layouts

---

- *Block formatting context*

- Absolute Positionierung: Sie können in diesem Kontext einzelne Blöcke an bestimmte Stellen der Seite „festnageln“. Sie überdecken dann eventuell andere!
- Anwendungsbeispiele:
  - Seiten-Layout, [siehe](#) CSS-Kap. 9.6
  - Popup-Fenster



- Floats: (Größere) Rechtecke aus einem Inline-Kontext, die nach links oder rechts verschoben werden, und um die herum der Text fließt. Sie verlassen u.U. ihre umschreibende Box und überlappen mit folgenden Blöcken, wo sie ebenfalls Text veranlassen, um sie herum zu fließen.
  - Spezial-Thema, hier nicht weiter verfolgt.

# \* CSS2 vs. *frames* und/oder Tabellen

---

- HTML-Gestaltung verwendete oft *frames* oder Tabellen zur Layoutkontrolle
  - Beide Ansätze haben Nachteile und werden heute vermieden!
- CSS2 bietet inzwischen mehrere Alternativen:
  - Absolute Positionierung von Elementen / Boxen!
  - Flexboxen
  - **Grid-Layout (neu!)**
- Beispiel zu mehrspaltigem Layout mit CSS:
  - <https://wiki.selfhtml.org/wiki/CSS/Tutorials>  
(Unterpunkte zu "Layouts")
- Zur Diskussion von Vor- und Nachteilen sowie Alternativen siehe auch:
  - S. Mintert, CSS-Tutorial, Teil 2: Frames, Tabellen und XML. iX 4/2003, pp. 138-143.
  - H. Braun, Stilspaltereien – Mehrspaltige Webseiten-Layouts mit CSS. c't 20/2006, pp. 191-191.
  - H. Braun, Neue Web-Layouts – CSS-Gestaltung mit **Flexboxen**. c't 11/2015, pp. 150ff-191.
  - <http://webkrauts.de/artikel/2012/css3-flexbox-abloesung-fuer-float-layouts>
  - H. Braun, Hinter Gittern. Layout-**Grids** mit CSS. c't 6/2017, pp. 170-172.
  - <https://css-tricks.com/snippets/css/complete-guide-grid/>

# \* Layoutkontrolle mit Grid: Beispiel

- HAML-Code (nur body-Teil eines Layouts)

Entsprechender HTML-Code

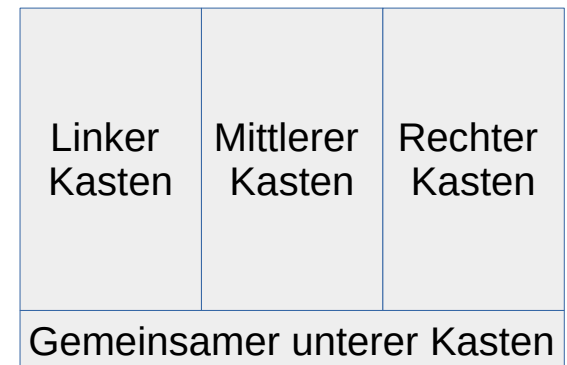
**%body**

```
#k_links Linker Kasten
#k_mitte Mittlerer Kasten
#k_rechts Rechter Kasten
#k_unten Gemeinsamer unterer Kasten
```

```
<body>
  <div id="k_links">Linker Kasten</div>
  <div id="k_mitte">Mittlerer Kasten</div>
  <div id="k_rechts">Rechter Kasten</div>
  <div id="k_unten">
    Gemeinsamer unterer Kasten
  </div>
</body>
```

- (S)CSS-Code

```
body {
  display: grid;
  grid:
    "links mitte rechts" 1fr
    "unten unten unten" 30px
    / 1fr 1fr 1fr;
}
#k_links { grid-area: links; }
/* ... andere analog ... */
#k_unten { grid-area: unten; }
```



# \* CSS2: *boxes*, „*display*“ property

---

- Box-Arten (Werte für die Eigenschaft „*display*“)
  - *block-level elements*, *block boxes*
    - *block*, *list-item*
  - *in-line level elements*, *in-line boxes*
    - *inline*, *inline-table*
  - Spezialfälle
    - *marker*: Erzeugt Nummern u.a. Randzeichen neben *list-item elements*
    - *none*: Erzeugt keine *box* für diesen *selector*.
    - Die *\*table\**-Familie - siehe Seite „CSS2.1, Kap. 17.2“
- Beliebte Effekt (meist im Zusammenspiel mit JavaScript):
  - Änderung des *display*-Werts auf „*none*“ → Ausblenden
  - Setzen von *display* auf früheren Wert → Wieder einblenden
  - Statisch auf „*none*“, später überschreiben → Aktivieren

# \* CSS2: *generated content*

---

- Überraschung (?!):
  - CSS gestaltet nicht nur vorhandene Inhalte, sondern kann u.U. auch selbst die Quelle von Inhalten sein!
  - Diskussion: Eine Verwässerung der klaren Trennung zwischen abstraktem Inhalt (HTML-Dokument) und Gestaltungsdaten (CSS)?
  - Urteilen Sie selbst anhand der Beispiele aus CSS-Kapitel 12
- Von CSS erzeugte Inhalte ([Quelle](#))
  - Positionierung: Pseudo-Attribute **:before** und **:after**
  - Inhalt: Die Eigenschaft **content**
  - Anwendungskontext Anführungszeichen im internat. Umfeld (12.3)
  - Automatische Zähler, z.B. für Kapitel (12.4)
  - Listen und ihre Marker und Aufzählungszeichen

# \* CSS2: Fehlerbehandlung

---

- **Kein Abbruch, keine Fehlermeldung!**
  - Im Gegensatz zu XML herrscht bei CSS die „tolerante“ HTML-Tradition vor.
  - **Grundregel:**
    - **Teile einer CSS-Datei, die syntaktisch nicht korrekt sind, werden ignoriert.**
  - Gleiches trifft auf Fehler zu, deren Ursache aus unbekannten Eigenschaften bzw. Schlüsselwörtern besteht.
  - Werden bestimmte Reihenfolgen nicht eingehalten, ignoriert der Browser (eigentlich: *User Agent*, UA) auch derartig falsch platzierte Anweisungen.
    - Beispiel: Ein `@include` erst *nach* Angabe des ersten *ruleset*
- **Einzelheiten:**
  - Siehe Abschnitt 4.2 der CSS2-Spezifikationen
- **Tipp:**
  - Validieren Sie Ihre CSS-Quellen! → <http://jigsaw.w3.org/css-validator/>



# CSS2: Nachwort

---

- Über den Umgang mit CSS2
  - Das hier präsentierte Material soll nur einen ersten Eindruck von den Möglichkeiten von CSS2 verschaffen.
    - Wer CSS ernsthaft einsetzen will, sollte sich mit den Spezifikationen selbst beschäftigen.
    - Diese sind umfangreich (19 Kapitel und 8 Anhänge). Es gibt zahlreiche Eigenschaften zu entdecken, aber wenig Neues zu lernen - man nutze die Spezifikationen einfach als Referenz.
    - Praxisnahe, kochbuchartige CSS/XML-Einführung in: „XML for the Word Wide Web“ von E. Castro, sowie „SelfHTML“
  - Vorsicht beim Produktionseinsatz von CSS2!
    - Es gibt Implementierungslücken und Unterschiede selbst in aktuellen Browsern. Ausgiebig testen mit allen eingesetzten Browsern!
    - *Quirks mode* vs. *strict mode* unterscheiden, s.u.
    - Tipp: Vergleichslisten suchen, nur Eigenschaften verwenden, die bereits hinreichende Unterstützung erfahren - etwa über Google mit Stichwörtern: „CSS compatibility“