Kapitel WT:III (Fortsetzung)

III. Dokumentsprachen

- Auszeichnungssprachen
- HTML
- Cascading Stylesheets CSS
- XML-Grundlagen
- XML-Schema
- Die XSL-Familie
- Parse-Paradigmen und APIs für XML

HTML

Einordnung

Q. SGML hat alle notwendigen Konzepte – warum überhaupt HTML?

A. HTML ist ein guter Kompromiss zwischen Einfachheit und Ausdrucksstärke.

HTML ermöglicht eine strikte Trennung
zwischen Dokumenteninhalt und Dokumentendarstellung,
erzwingt sie aber nicht.

Bemerkungen:

- HTML kompakt:
  1. Historie
  2. HTML Dokumentenverarbeitung
  3. Aufbau HTML-Dokument
  4. Inhaltsmodelle
  5. Universalattribute
  6. HTML-Elementtypen

HTML [W3C status, reports]

Historie

1991 Vorstellung von ersten Versionen für URL, HTTP und HTML.

1994 HTML 2.0. Basiert auf standardkonformer DTD.

1998 HTML 4.0. Führt Cascading Stylesheets CSS ein.

1999 HTML 4.01. Recommendation. [W3C REC]

HTML [W3C status, reports]

Historie

1991 Vorstellung von ersten Versionen für URL, HTTP und HTML.

1994 HTML 2.0. Basiert auf standardkonformer DTD.

1998 HTML 4.0. Führt Cascading Stylesheets CSS ein.

1999 HTML 4.01. Recommendation. [W3C REC]

2000 XHTML 1.0. Reformulierung von HTML4 in XML. [W3C REC, differences] [Wikipedia]

2010 XHTML 2.0. Working Group Note, “back to the roots”. [W3C NOTE, 1.1.3]

HTML [W3C status, reports]

Historie

1991 Vorstellung von ersten Versionen für URL, HTTP und HTML.

1994 HTML 2.0. Basiert auf standardkonformer DTD.

1998 HTML 4.0. Führt Cascading Stylesheets CSS ein.

1999 HTML 4.01. Recommendation. [W3C REC]

2000 XHTML 1.0. Reformulierung von HTML4 in XML. [W3C REC, differences] [Wikipedia]

2010 XHTML 2.0. Working Group Note, “back to the roots”. [W3C NOTE, 1.1.3]

2008–2014 HTML5. Recommendation. Loslösung von SGML, neue Struktur- und Multimedia-Elemente. [W3C REC, differences] [Wikipedia article, figure]

2015 Polyglot Markup. XML-Serialisierung von HTML5. [W3C NOTE] [Stackexchange]

2017 HTML 5.2. Recommendation. [W3C REC]

2023 HTML. Living Standard. [WHATWG living standard, developer, about developer]

Bemerkungen (HTML4) :

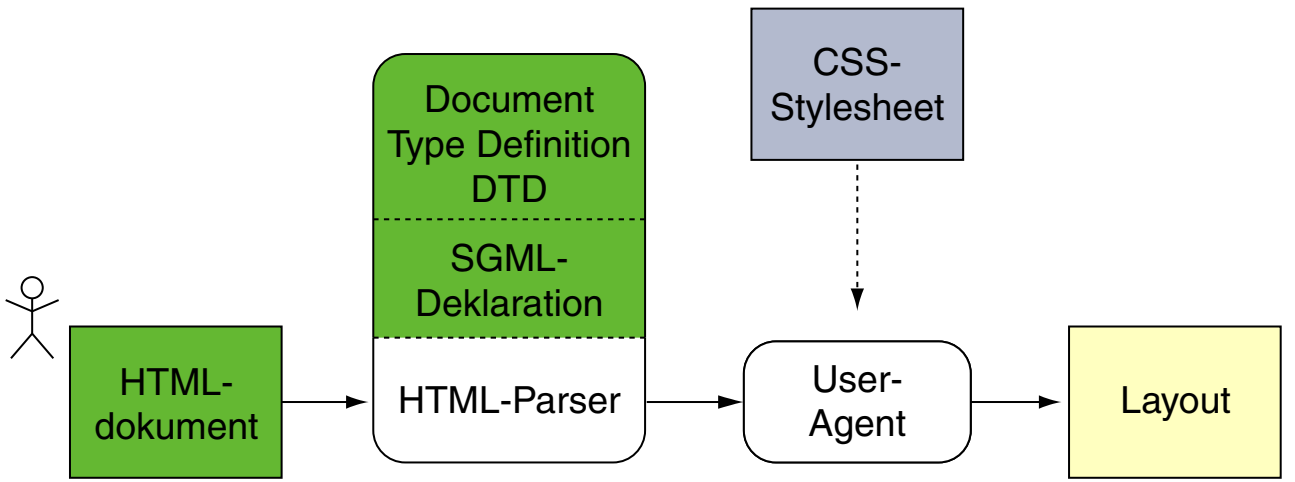
- Beispiele für die fehlende Trennung zwischen Dokumenteninhalt und Dokumentendarstellung sind Formatierungsangaben wie `<font>`, `<center>`, etc.
- Mit der Einführung von Cascading Stylesheets in HTML 4.0 existiert ein Mechanismus, um Formatierungsangaben aus dem Dokumenteninhalt auszugliedern.
- XHTML 1.0 bringt keine neue Funktionalität gegenüber HTML 4.0, enthält aber die (kleineren) syntaktischen Anpassungen für den XML-Standard.
- Bei der Weiterentwicklung von XHTML 2.0 konnte keine Einigung zwischen W3C und der Industrie (WHATWG-Konsortium) erzielt werden. Es folgte eine konkurrierende Entwicklung des HTML5-Standards. Seit Mai 2019 arbeiten das W3C und WHATWG jedoch gemeinsam an der Weiterentwicklung von HTML. [W3C Blog]
- Der Standardisierungsprozess der W3C ist formalisiert und spiegelt sich in den verschiedenen Leveln der veröffentlichten Reports wider. [W3C reports, rec. track, diagram]

Bemerkungen (HTML5) :

- HTML5 führt Strukturelemente wie `<header>`, `<footer>` oder `<nav>` ein, um die Semantik eines Elements im Dokument explizit zu machen und die Interpretation (insbesondere für Maschinen) zu erhöhen.
- HTML5 zielt in besonderem Maße darauf ab, sogenannten *Rich Content* darstellen zu können. Beispielsweise ermöglichen `<canvas>`, `<video>` und `<audio>` die native Medieneinbindung und machen damit vorherige Plugin-Technologien wie Flash überflüssig.
- HTML5 reagiert auf die große Menge nicht valider Dokumente im Web (Stichwort: *tag soup*), die bislang jeder Browser auf eigene Weise behandelt, mit einer standardisierten Fehlerbehandlung (Stichwort: *quirks mode*). [W3C wiki] [Wikipedia]
- HTML5-Logo: 
  HTML5-Schreibweise: HTML5 oder HTML 5? [WHATWG]

HTML

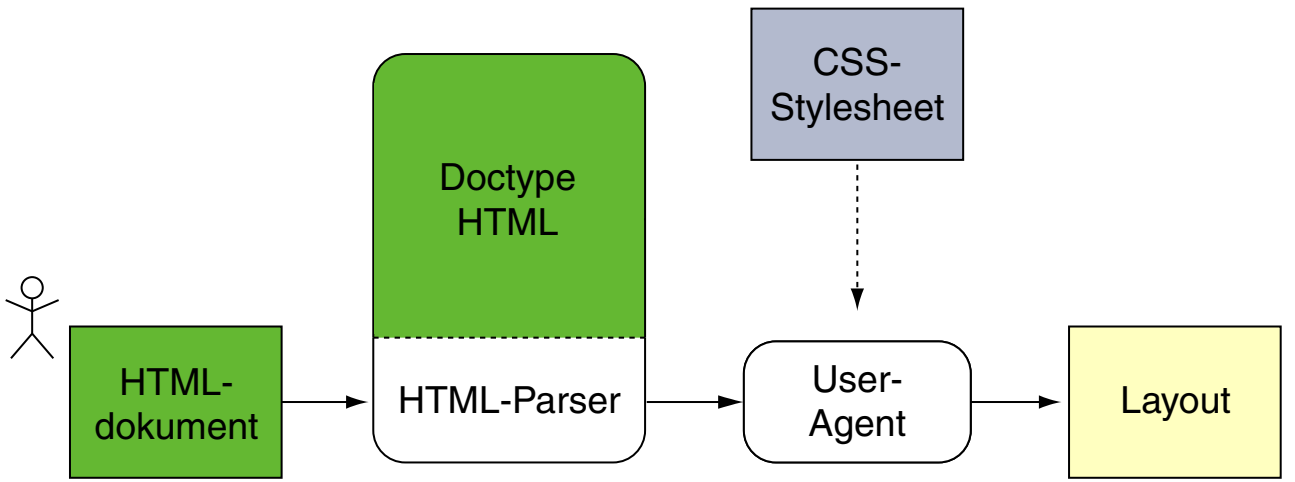
HTML Dokumentenverarbeitung (HTML4)



Vergleiche hierzu die SGML Dokumentenverarbeitung.

HTML

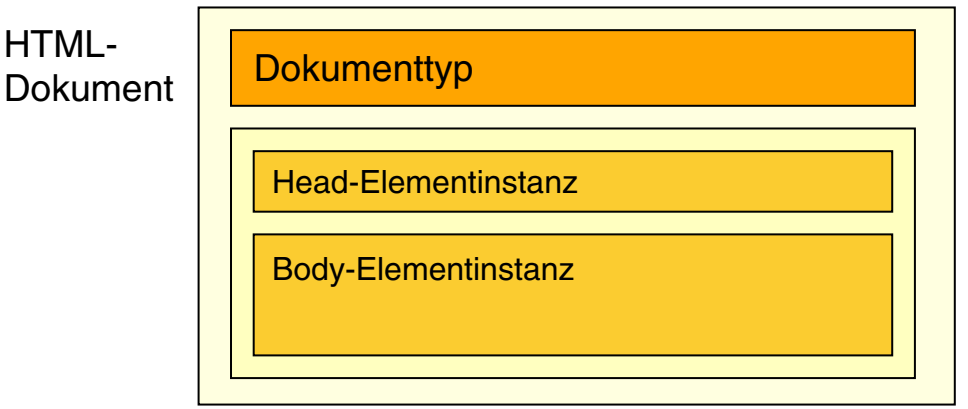
HTML Dokumentenverarbeitung (HTML5)



Vergleiche hierzu die SGML Dokumentenverarbeitung.

HTML

Aufbau HTML-Dokument

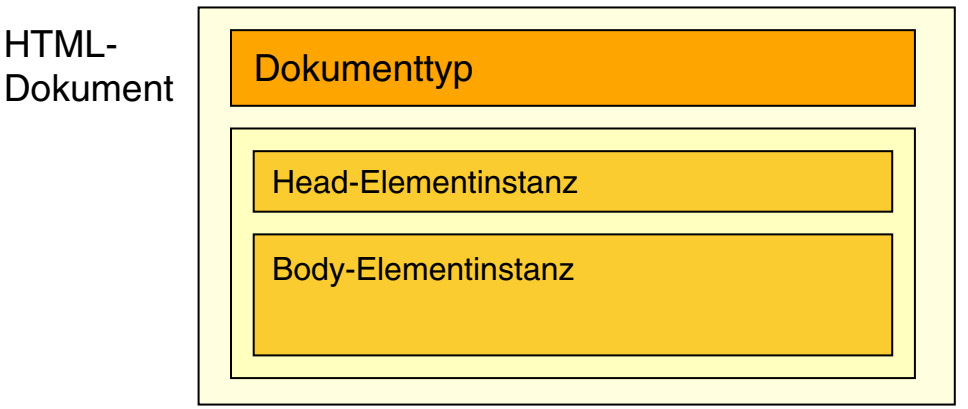



```
<!DOCTYPE html>

<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

- Das `<html>`-Element repräsentiert die Dokument-Wurzel. [WHATWG HTML 4.1]
- Das `<head>`-Element repräsentiert die Meta-Daten. [WHATWG HTML 4.2]
- Das `<body>`-Element repräsentiert den Dokumentinhalt. [WHATWG HTML 4.3]
- Vergleiche hierzu die XML-Dokumentstruktur.

HTML

Aufbau HTML-Dokument




```
<!DOCTYPE html>

<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

- Das `<html>`-Element repräsentiert die Dokument-Wurzel. [WHATWG HTML 4.1]
- Das `<head>`-Element repräsentiert die Meta-Daten. [WHATWG HTML 4.2]
- Das `<body>`-Element repräsentiert den Dokumentinhalt. [WHATWG HTML 4.3]
- Vergleiche hierzu die XML-Dokumentstruktur.

Allgemeine Form einer HTML-Elementinstanz [WT:III SGML]:

```
<elementname {attribute}*> ... </elementname>
```

HTML

Deklaration der DTD (HTML4)

HTML hat eine feste Dokumentstruktur, die unter HTML4 als DTD (*Document Type Definition*) spezifiziert ist. Unterscheidung von drei DTD-Varianten [W3C 1, 2]:

1. Strict

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
       "https://www.w3.org/TR/html4/strict.dtd">
   ```

   Trennung zwischen Inhalt und Darstellung: keine Formatierungsangaben erlaubt; strenge Verschachtelungsregeln; kein Inhalt ohne Block-Level-Auszeichnung.

2. Transitional

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "https://www.w3.org/TR/html4/loose.dtd">
   ```

   Ohne die Beschränkungen der Strict-DTD.

3. Frameset

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
       "https://www.w3.org/TR/html4/frameset.dtd">
   ```

   Für HTML-Dokumente mit Framesets.

HTML

Deklaration der DTD (HTML4)

HTML hat eine feste Dokumentstruktur, die unter HTML4 als DTD (*Document Type Definition*) spezifiziert ist. Unterscheidung von drei DTD-Varianten [W3C 1, 2]:

1. Strict

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
       "https://www.w3.org/TR/html4/strict.dtd">
   ```

   Trennung zwischen Inhalt und Darstellung: keine Formatierungsangaben erlaubt; strenge Verschachtelungsregeln; kein Inhalt ohne Block-Level-Auszeichnung.

2. Transitional

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
       "https://www.w3.org/TR/html4/loose.dtd">
   ```

   Ohne die Beschränkungen der Strict-DTD.

3. Frameset

   ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
       "https://www.w3.org/TR/html4/frameset.dtd">
   ```

   Für HTML-Dokumente mit Framesets.

Bemerkungen:

- Ein HTML4-Dokument ohne DTD-Deklaration wird nach den Regeln der Transitional-DTD für HTML 4.01 verarbeitet.
- Ein HTML4-Dokument darf nur *eine* DTD besitzen. Bei der Verwendung von Frames ermöglicht die Frameset-DTD für jedes Frame die Einbindung einer DTD.
- HTML5 ist weitgehend kompatibel zu HTML 4.01 und XHTML 1.0, basiert aber nicht mehr auf SGML. Folglich ist die Dokumentstruktur nicht mehr in Form einer DTD spezifiziert.

  [Wikipedia: DTD-less, FPI] [CoreLangs] [WHATWG HTML 13.1.1]

HTML

Inhaltsmodelle (*Content Models*) (HTML4)

Elementinstanzen innerhalb einer `<body>`-Elementinstanz gehören zu genau einer der folgenden zwei Kategorien [MDN block-level, inline]:

1. Block-Level-Elemente

2. Inline-Elemente

HTML

Inhaltsmodelle (*Content Models*) (HTML4)

Elementinstanzen innerhalb einer `<body>`-Elementinstanz gehören zu genau einer der folgenden zwei Kategorien [MDN block-level, inline]:

1. Block-Level-Elemente

   Instanzen von Block-Level-Elementen erzeugen immer einen Absatz im Textfluss; sie können normalen Text und Instanzen von Inline-Elementen enthalten; einige dürfen auch Instanzen anderer Block-Level-Elemente enthalten.

   Beispiele für Block-Level-Elemente:
   `<center>`, `<div>`, `<form>`, `<h1>`, `<noframes>`, `<p>`, `<table>`, `<ul>`

2. Inline-Elemente

   Instanzen von Inline-Elementen werden in derselben Zeile wie der vorhergehende Text gesetzt; sie können normalen Text und Instanzen weiterer Inline-Elemente enthalten.

   Beispiele für Inline-Elemente:
   `<a>`, `<br>`, `<cite>`, `<em>`, `<font>`, `<img>`, `<small>`, `<span>`

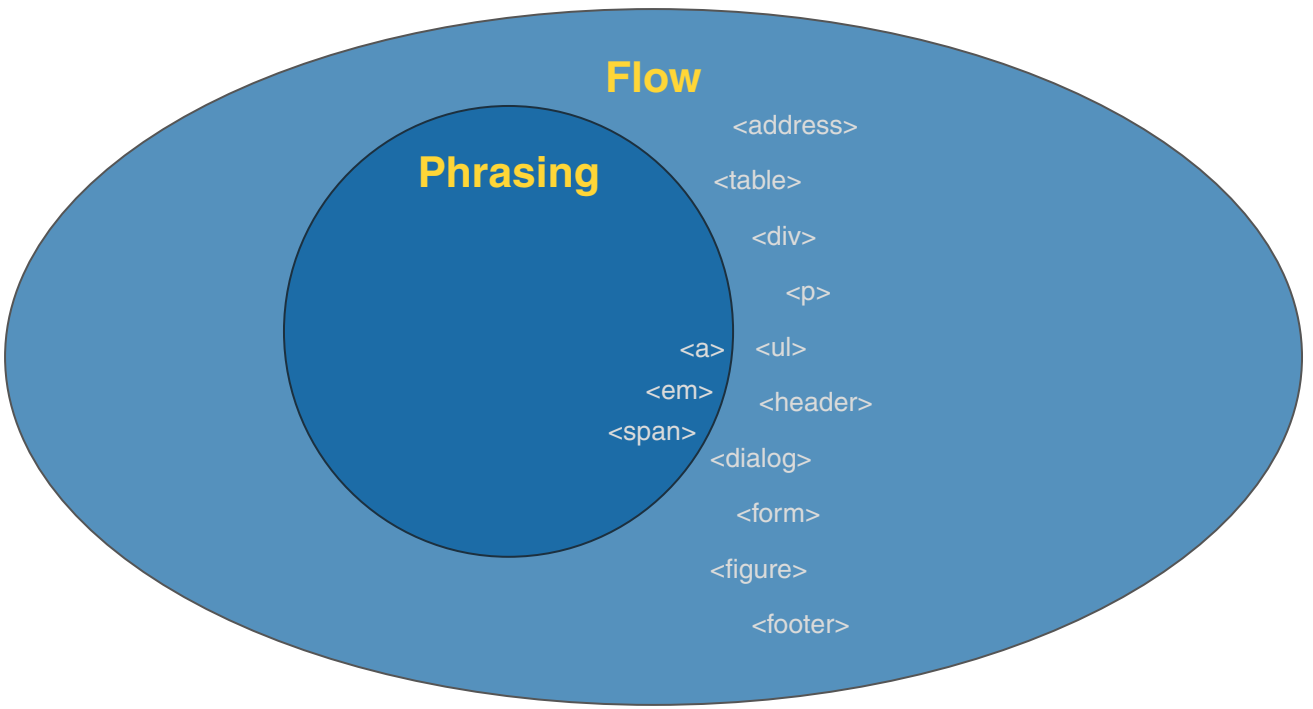
Bemerkungen:

- Die Verarbeitung von Block-Level-Elementen aus Sicht des Layout-Programms (beispielsweise mit einem Web-Browser) ist mit dem Verhalten von LaTeX im `\vmode` vergleichbar. Die Verarbeitung von Inline-Elementen ist mit dem Verhalten von LaTeX im `\hmode` vergleichbar.

HTML

Inhaltsmodelle (*Content Models*) (HTML5) [WHATWG HTML 3.2.5] [MDN]

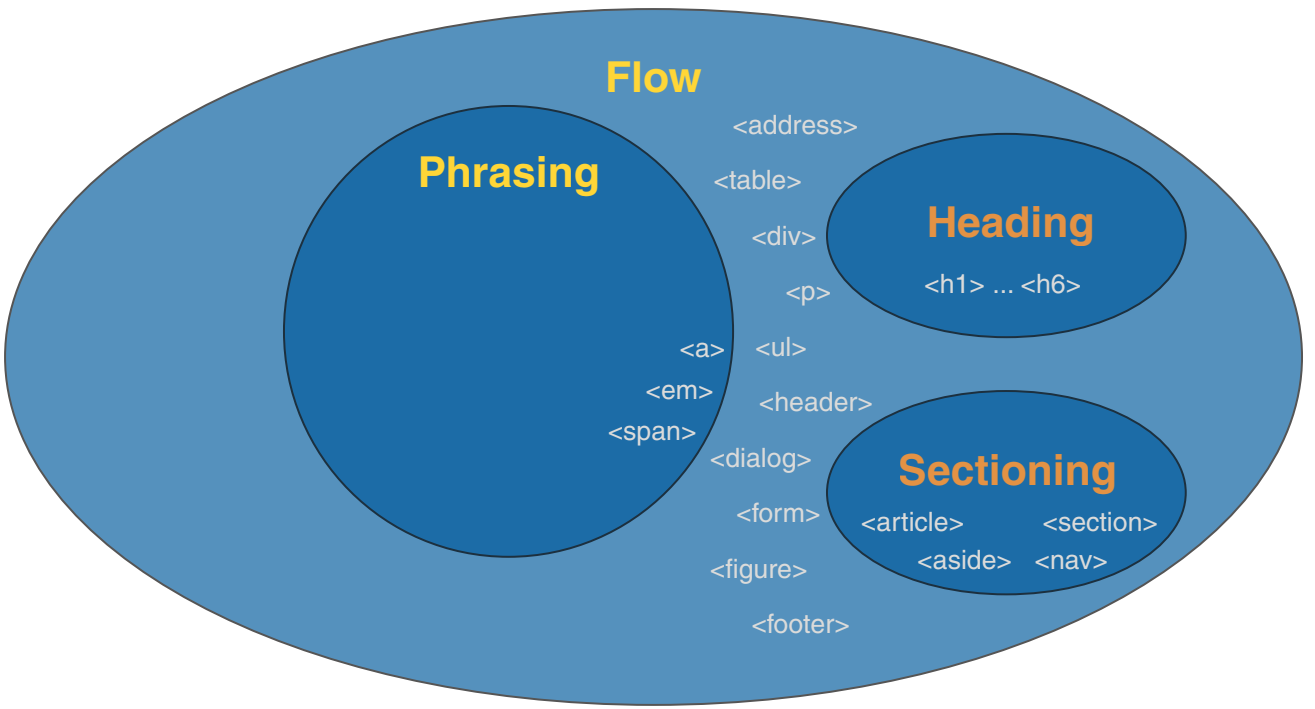
Elementinstanzen innerhalb einer `<body>`-Elementinstanz fallen in mindestens eine der folgenden sieben Inhaltskategorien [WHATWG HTML 3.2.5.2]:



HTML

Inhaltsmodelle (*Content Models*) (HTML5) [WHATWG HTML 3.2.5] [MDN]

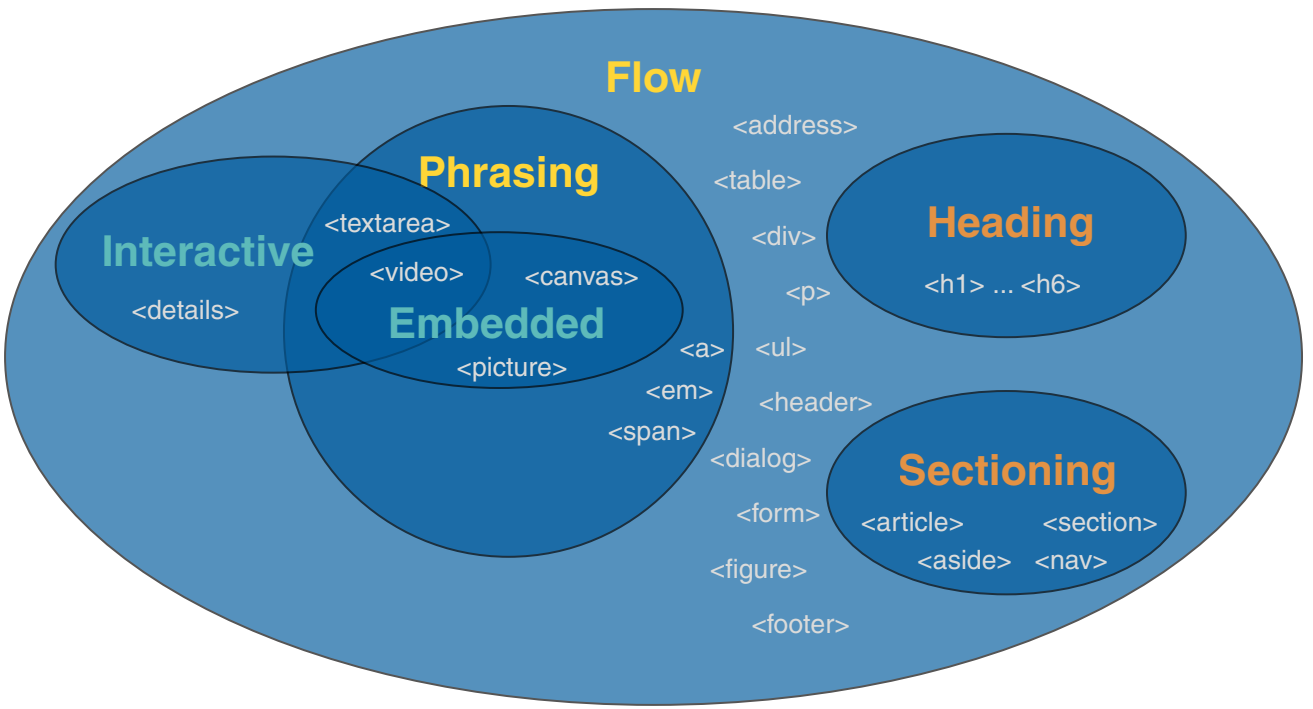
Elementinstanzen innerhalb einer `<body>`-Elementinstanz fallen in mindestens eine der folgenden sieben Inhaltskategorien [WHATWG HTML 3.2.5.2]:



HTML

Inhaltsmodelle (*Content Models*) (HTML5) [WHATWG HTML 3.2.5] [MDN]

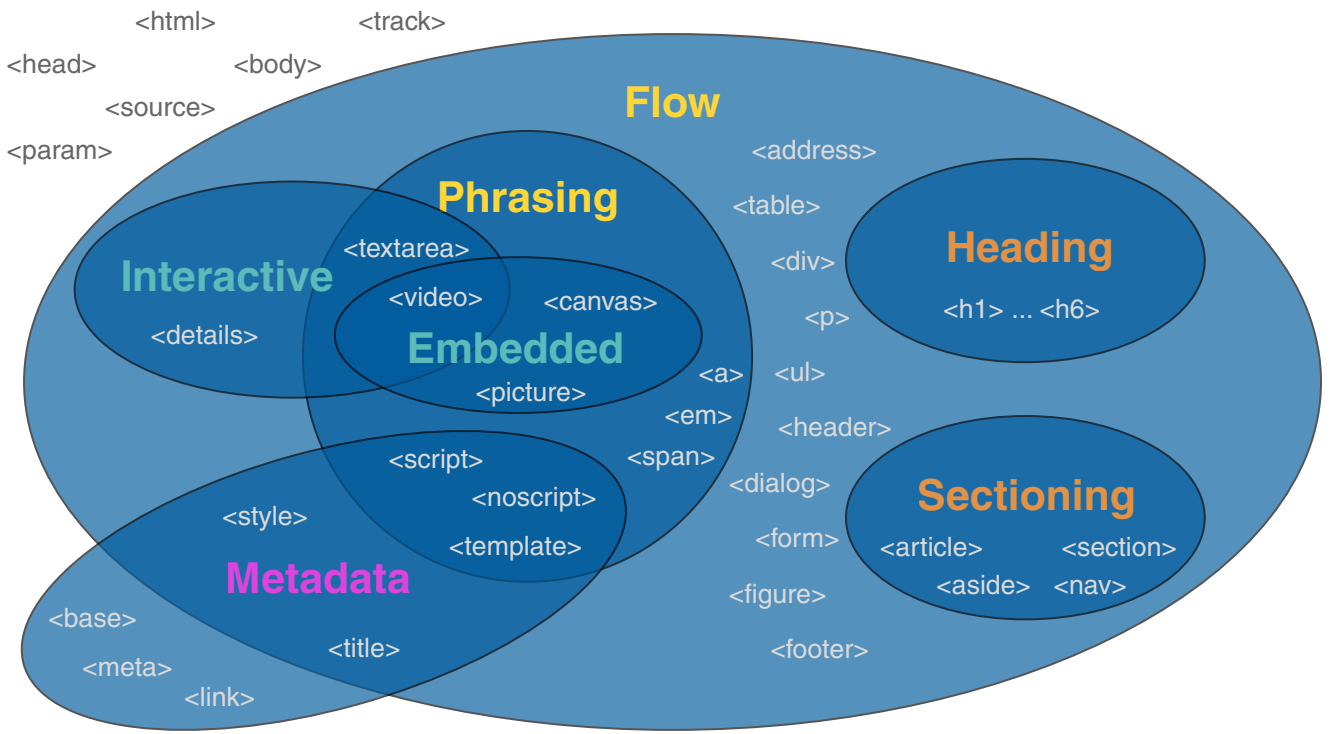
Elementinstanzen innerhalb einer `<body>`-Elementinstanz fallen in mindestens eine der folgenden sieben Inhaltskategorien [WHATWG HTML 3.2.5.2]:



HTML

Inhaltsmodelle (*Content Models*) (HTML5) [WHATWG HTML 3.2.5] [MDN]

Elementinstanzen innerhalb einer `<body>`-Elementinstanz fallen in mindestens eine der folgenden sieben Inhaltskategorien [WHATWG HTML 3.2.5.2]:



Bemerkungen:

- Bei HTML5 ist die syntaktische Aufteilung in Block-Level- und Inline-Elemente durch eine an semantischen Überlegungen orientierte Aufeilung abgelöst bzw. ergänzt worden. Aus Sicht des Layout-Programms (beispielsweise des Web-Browsers) gilt für die beiden Philosophien in etwa die folgende Entsprechung [MDN]:
  - Block-Level-Elemente (HTML4) $\sim$ Flow Content [MDN]
  - Inline-Elemente (HTML4) $\sim$ Phrasing Content [MDN]
- HTML5 verzichtet auf eine Reihe von (Block-Level-)Elementen, die unter HTML4 in erster Linie zur Layout-Gestaltung dienen [w3schools]:
  `<center>, <frame>, <frameset>, <noframes>`

HTML

Universalattribute (*Global Attributes*) [WHATWG HTML 3.2.6] [w3schools] [SELFHTML]

Universalattribute sind in allen HTML-Elementen verwendbar. Einteilung und Beispiele:

1. Allgemeine

2. Zur Internationalisierung

HTML

Universalattribute (*Global Attributes*) [WHATWG HTML 3.2.6] [w3schools] [SELFHTML]

Universalattribute sind in allen HTML-Elementen verwendbar. Einteilung und Beispiele:

1. Allgemeine

`class`	ordnet der Elementinstanz eine Stylesheet-Klasse zu
`id`	ordnet der Elementinstanz einen eindeutigen Namen zu
`style`	definiert CSS-Angaben zur Formatierung der Elementinstanz
`title`	definiert den Mouse-Over-Text

2. Zur Internationalisierung

`dir`	definiert die Schreibrichtung für Text in der Elementinstanz
`lang`	definiert Landessprache (nach RFC 1766 → ISO 639-1)
`translate`	spezifiziert, ob Inhalte bei Lokalisierung zu übersetzen sind

HTML

Universalattribute (*Global Attributes*) (Fortsetzung)

Universalattribute sind in allen HTML-Elementen verwendbar. Einteilung und Beispiele:

3. Zum Event-Handling [WHATWG HTML 8.1.7.2]

`onclick`	Ausführen von Script-Code beim Anklicken der Elementinstanz
`onkeydown`	Ausführen von Script-Code beim Herunterdrücken einer Taste
`onmouseover`	Ausführen von Script-Code beim Überfahren der Elementinstanz

4. Für eigene Daten (*Custom Data Attributes*) [WHATWG HTML 3.2.6.6] [w3schools]

`data-*`	Semantik definiert durch Programmierer der Web-Site

Attributnamen müssen mit "`data-`" beginnen, XML-kompatibel sein und dürfen keine Großbuchstaben enthalten.

Custom Data Attributes sind für benutzerdefinierte Daten, die nur für eine Seite oder Anwendung gelten, gedacht. Für generische Erweiterungen sollte eine Technologie wie "Microdata" und ein standardisiertes Vokabular verwendet werden. [WHATWG HTML 5.]

HTML

Organisation der HTML5-Spezifikation von W3C bzw. WHATWG

[W3C REC (bis Jan'21)] [WHATWG living standard, developer]

1. Introduction
2. Common infrastructure
3. Semantics, structure, and APIs
4. The elements of HTML
5. Microdata
6. User interaction
7. Loading web pages
8. Web application APIs
9. Communication
10. Web workers
11. Worklets
12. Web storage
13. The HTML syntax
14. The XML syntax
15. Rendering
16. Obsolete features
17. IANA considerations

HTML

Organisation der HTML5-Spezifikation von W3C bzw. WHATWG

[W3C REC (bis Jan'21)] [WHATWG living standard, developer]

1. Introduction
2. Common infrastructure
3. Semantics, structure, and APIs
4. The elements of HTML
5. Microdata
6. User interaction
7. Loading web pages
8. Web application APIs
9. Communication
10. Web workers
11. Worklets
12. Web storage
13. The HTML syntax
14. The XML syntax
15. Rendering
16. Obsolete features
17. IANA considerations

- 4.1 The document element
- ∗ 4.2 Document metadata
- ∗ 4.3 Sections
- ∗ 4.4 Grouping content
- ∗ 4.5 Text-level semantics
- ∗ 4.6 Links
- 4.7 Edits
- ∗ 4.8 Embedded content
- ∗ 4.9 Tabular data
- ∗ 4.10 Forms
- 4.11 Interactive elements
- ∗ 4.12 Scripting
- 4.13 Custom elements
- 4.14 Common idioms
- 4.15 Disabled elements
- 4.16 Matching HTML elements

HTML

Markup- versus Interface-Sicht

Die HTML5-Spezifikation von W3C bzw. WHATWG definiert die HTML-Elemente nicht mehr mittels einer DTD (*Document Type Definition*) sondern mittels einer IDL (*Interface Definition Language*).

1. Die Definition via DTD fokussiert auf die Inhalts- bzw. Markup-Sicht eines HTML-Elements.

2. Die Definition via IDL fokussiert auf die Objekt-, Interface- bzw. Programmiersicht und ermöglicht die Integration von Eigenschaften und Verhalten (vgl. Objektparadigma der Programmierung).

   Stichwort: DOM (*Document Object Model*) -Interface

Bemerkungen:

- Markup- versus Interface-Sicht am Beispiel des `<a>`-Elements.
  - Verwendung von `<a>...</a>` im serialisierten HTML-Dokument:

```
<ul>
  <li> <a href="https://html.spec.whatwg.org/">HTML</a> </li>
  <li> <a id="example3">Example 3</a> </li>
</ul>
```

  - HTML4- und HTML5-Spezifikation für das `<a>`-Element:

	Attribut-Semantik	
	Inhalts- bzw. Markup-Sicht	Interface-Sicht
HTML4	[W3C DTD]	–
HTML5	[W3C content]	[W3C DOM interface]
	[WHATWG content]	[WHATWG DOM interface]
	[WHATWG developer]	–

Bemerkungen: (Fortsetzung)

- Entsprechend der unterschiedlichen Beschreibungsparadigmen (Inhalts- bzw. Markup-Sicht versus Interface-Sicht) werden die Attribute eines HTML-Elements aus Inhalts- bzw. Markup-Sicht als *Content Attributes* und aus Interface-Sicht als *IDL Attributes* oder *JavaScript Property* bezeichnet. [MDN] [WHATWG HTML 3.0]
- Einige IDL-Attribute (nicht alle) „spiegeln“ ein bestimmtes Inhaltsattribut wider: Beim Abfragen gibt das IDL-Attribut den aktuellen Wert des Inhaltsattributs zurück, beim Setzen des IDL-Attributs ändert sich der Wert des Inhaltsattributs entsprechend. [WHATWG HTML 2.6.1]
- Eine HTML-Datei kann als Serialisierung der dazu gehörigen Instanz des Document Object Models (DOM) verstanden werden. Beim Einlesen dieser HTML-Datei durch den Browser wird das zugehörige DOM im Speicher aufgebaut.
- Die Developer’s Edition der WHATWG stellt die Markup-Sicht in den Vordergrund und ist besonders für die Autoren von Web-Seiten geeignet. [WHATWG about developer]
- Eine Interface Definition Language, IDL, ist eine Sprache zur Beschreibung der API einer Software-Komponente. Mittlerweile gibt es eine dedizierte Web IDL, die vom W3C entwickelt wird und auf die Besonderheiten der Web-Plattform (typische Web-Hardware + Browser) zugeschnitten ist. [Wikipedia: IDL, Web IDL] [W3C REC, status]

HTML

4.2 Document Metadata [WHATWG HTML 4.2]

- Titel [SELFHTML]

```
<head>
  <title>Lemmy Caution's Strange Adventures</title>
</head>
```

Bei HTML4 ist der Titel obligatorisch, bei HTML5 kann er fehlen, falls er ableitbar ist. Der Titel erscheint nicht im dargestellten HTML-Dokument, wird aber als Fenstertitel, Lesezeichen, von Robots etc. ausgewertet.

- Meta-Tags [SELFHTML]

```
<head>
  <title>...</title>

  <meta charset="utf-8">
  <meta name="author" content="Judea Pearl">
  <meta name="keywords" content="Heuristics, Search, Bayes">
  <meta http-equiv="refresh" content="60">

</head>
```

Meta-Tags haben meist zwei Attribute „Eigenschaft = Wert“ (`name` bzw. `http-equiv` = `content`); sie dienen zur Information von Web-Browsern, Robots und Web-Servern.

HTML

4.2 Document Metadata [WHATWG HTML 4.2] (Fortsetzung)

- **Adressbasis** [SELFHTML]

```
<head>
  <title>...</title>

  <base href="https://www.my-webserver.de/absolute/path">

</head>
```

Definiert einen absoluten Bezugspfad und ermöglicht so die Verwendung von relativen Pfaden im Dokument.

- **Links** [SELFHTML]

```
<head>
  <title>...</title>

  <link rel="stylesheet" href="../share/bib.css" type="text/css">

</head>
```

Ermöglicht die Referenzierung (keine Hyperlinks) von Dokumenten; wird meist zur Angabe von externen Stylesheets verwendet.

Bemerkungen:

- Zur Standardisierung von Meta-Tags hat das W3C die Sprache RDF (*Resource Description Framework*) entworfen.
- Meta-Tags, die mit `http-equiv` definiert sind, werden vom Client-Programm wie ein HTTP-Entity-Header einer HTTP-Response-Message interpretiert. Ein gleichnamiger HTTP-Header in der Response-Message hat Vorrang gegenüber einer Metadata-Angabe im HTML-Dokument. [SELFHTML]

HTML

4.3 Sections [WHATWG HTML 4.3, summary]

- Strukturelemente (HTML5) [MDN]

`<article>`	eigenständiger Inhalt, ggf. mit eigenem `<header>` und `<footer>`
`<section>`	(1) Gruppierung verschiedener `<article>` in Themen, (2) Einteilung *eines* `<article>` in Abschnitte, typisch mit Überschrift.
`<nav>`	Navigationsmenü oder andere Navigationsmöglichkeiten
`<aside>`	Gruppierung von verwandter Information mit Bezug zum Hauptinhalt
`<header>`	Kopfinformationen einer Website oder eines Artikels
`<footer>`	Fußzeile einer Website oder eines Artikels

HTML

4.3 Sections [WHATWG HTML 4.3, summary]

- Strukturelemente (HTML5) [MDN]

`<article>`	eigenständiger Inhalt, ggf. mit eigenem `<header>` und `<footer>`
`<section>`	(1) Gruppierung verschiedener `<article>` in Themen, (2) Einteilung *eines* `<article>` in Abschnitte, typisch mit Überschrift.
`<nav>`	Navigationsmenü oder andere Navigationsmöglichkeiten
`<aside>`	Gruppierung von verwandter Information mit Bezug zum Hauptinhalt
`<header>`	Kopfinformationen einer Website oder eines Artikels
`<footer>`	Fußzeile einer Website oder eines Artikels

- Überschriftselemente [WHATWG HTML 4.3.6] [SELFHTML]

```
<h1>Überschrift 1. Ordnung</h1>
...
<h6>Überschrift 6. Ordnung</h6>
```

HTML

4.4 Grouping Content [WHATWG HTML 4.4]

- Zusammenhängende Abschnitte

`<p>`	ohne Semantik	[MDN]
`<pre>`	vorformattierter Text	[MDN]
`<blockquote>`	zitierter Text	[MDN]

HTML

4.4 Grouping Content [WHATWG HTML 4.4]

- Zusammenhängende Abschnitte

`<p>`	ohne Semantik	[MDN]
`<pre>`	vorformattierter Text	[MDN]
`<blockquote>`	zitierter Text	[MDN]

- Listen

`<ol>`	geordnete Liste	[SELFHTML]
`<ul>`	ungeordnete Liste	
`<li>`	Listeneintrag	
`<dl>`	Definitionsliste	[SELFHTML]
`<dt>`	Definitionsüberschrift	
`<dd>`	Definitionseintrag	

HTML

4.5 Text-Level Semantics [WHATWG HTML 4.5]

Unterscheidung von Textauszeichnungen hinsichtlich ihrer Konkretheit [MDN]:

1. Physische Auszeichnungen (HTML4)

2. Logische Auszeichnungen (HTML4 und HTML5)

HTML

4.5 Text-Level Semantics [WHATWG HTML 4.5]

Unterscheidung von Textauszeichnungen hinsichtlich ihrer Konkretheit [MDN]:

1. Physische Auszeichnungen (HTML4)

`<i>`	zeichnet einen Text als kursiv aus
`<b>`	zeichnet einen Text als fett aus
`<u>`	zeichnet einen Text als unterstrichen aus
`<strike>`	zeichnet einen Text als durchgestrichen aus
`<tt>`	zeichnet einen Text in Schreibmaschinenschrift aus

2. Logische Auszeichnungen (HTML4 und HTML5)

`<em>`	zeichnet einen Text als betonten, wichtigen Text aus
`<strong>`	zeichnet einen Text als stark betont aus (Steigerung von `<em>`)
`<cite>`	zeichnet einen Text als Zitat aus
`<code>`	zeichnet einen Text als Quelltext aus
`<samp>`	zeichnet einen Text als Beispiel aus

Bemerkungen:

- *“[...]; strong is a logical state, and bold is a physical state. Logical states separate presentation from the content, and by doing so allow for it to be expressed in many different ways.”* [MDN]

- HTML5 verzichtet auf eine Reihe von Elementen, die unter HTML4 vordringlich zur physischen Auszeichnungen dienen [w3schools]: `<basefont>`, `<big>`, `<dir>`, `<strike>`, `<tt>`

  Die weiteren HTML4-Elemente zur physischen Auszeichnung haben unter HTML5 explizite Verwendungshinweise erhalten. Beispiele: `<b>`, `<i>`, `<u>` [w3schools b, i, u]

- Bei HTML5 dienen die Elemente `<ins>`, `<del>` zur Auszeichnung von sogenannten *Edits*. [WHATWG HTML 4.7] [w3schools]

- Beispiele für physische Auszeichnungen in LaTeX sind die Schriftschnitte und -gewichte:
  `\itshape`, `\bfseries`, `\fontfamily{phv}\fontsize{8}{0}\selectfont`

  Beispiele für logische Auszeichnungen in LaTeX:
  `\em`, `\begin{quote} ... \end{quote}`

HTML

4.6 Links [WHATWG HTML 4.6] [SELFHTML]

Zur Definition von Hyperlinks dient das `<a>`-Element (*Anchor*). Als Inline-Element (HTML4) kann es keine Instanzen von Block-Level-Elementen auszeichnen; der erlaubte Kontext (HTML5) ist Phrasing Content [WHATWG HTML 4.5.1].

- Hyperlink

- Hyperlink-Ziel

HTML

4.6 Links [WHATWG HTML 4.6] [SELFHTML]

Zur Definition von Hyperlinks dient das `<a>`-Element (*Anchor*). Als Inline-Element (HTML4) kann es keine Instanzen von Block-Level-Elementen auszeichnen; der erlaubte Kontext (HTML5) ist Phrasing Content [WHATWG HTML 4.5.1].

- Hyperlink

  `<a` **`href`**`="`*URL*`">` — Ziel ist durch *URL* definiert [WHATWG HTML 4.3]

  Beispiel:

  `<a` **`href`**`="#`*Identifier*`">` — URL besteht nur aus Fragment-String

  Optionale Attribute des Anchor-Elements:

`title`	definiert den Mouse-Over-Text
`type`	MIME-Type des Zieldokuments
`download`	spezifiziert, dass lokal gespeichert werden soll

- Hyperlink-Ziel

  `<... id="`*Identifier*`">` — Zieldefinition im selben Dokument

Bemerkungen:

- Die Syntax von Hyperlinks ist unabhängig von dem angegebenen Ziel.
- URLs, die mit einem Dokumentanker #*Identifier* abschließen, werden auch als *Fragment-Identifier* bezeichnet, weil sie ein Dokument nicht als Ganzes, sondern abschnittsgenau adressieren.

HTML

4.8 Embedded Content [WHATWG HTML 4.8]

Wichtige Elemente:

- `<img>, <picture>, <source>` [MDN img, picture, source]
- `<iframe src="https://www.w3c.org"></iframe>` [MDN]
- `<embed>, <object>` [MDN]
- `<video data="introduction-video.mp4" controls></video>` [MDN]
- `<audio src="sample.mp3" controls></audio>` [MDN]
- `<map>, <area>` [MDN] [WHATWG]
- `<math>` [WHATWG]
- `<svg>` [WHATWG] [SELFHTML SVG vs. Canvas]

HTML

4.9 Tabular Data [WHATWG HTML 4.9] [SELFHTML]

- Elemente

`<table>`	Tabelle
`<caption>`	Tabellenüberschrift
`<colgroup>`	Spaltengruppe
`<col>`	Tabellenspalte
`<tbody>`	Tabellenkörper
`<thead>`	Tabellenkopf
`<tfoot>`	Tabellenfuß
`<tr>`	Tabellenzeile
`<td>`	einzelne Zelle
`<th>`	Zelle mit Überschrift

Spalte 1	Spalte 2	Spalte 3
Zelle 1.1	Zelle 1.2	Zelle 1.3
Zelle 2.1	Zelle 2.2	Zelle 2.3

[html-table.html]

HTML

4.10 Forms [WHATWG HTML 4.10] [SELFHTML]

Zum Formular gehört alles, was zwischen den `<form>`-Tags steht.

- Attribute des `<form>`-Elements [WHATWG HTML 4.10.3]

- Kindelemente des `<form>`-Elements

HTML

4.10 Forms [WHATWG HTML 4.10] [SELFHTML]

Zum Formular gehört alles, was zwischen den `<form>`-Tags steht.

- Attribute des `<form>`-Elements [WHATWG HTML 4.10.3]

`action`	definiert URL vom Server-Anwendungsprogramm
`enctype`	Angabe eines MIME-Typs
`method`	spezifiziert die `get` oder `post`-Methode des HTTP-Protokolls

- Kindelemente des `<form>`-Elements

`<label>`	Beschreibungstext zu Eingabefeld	[WHATWG HTML 4.10.4]
`<input>`	Definition von Eingabefeld	[WHATWG HTML 4.10.5]
`<fieldset>`	Gruppierung von Formularelementen	[WHATWG HTML 4.10.15]

HTML

4.10 Forms [WHATWG HTML 4.10] [SELFHTML]

Zum Formular gehört alles, was zwischen den `<form>`-Tags steht.

- Attribute des `<form>`-Elements [WHATWG HTML 4.10.3]

`action`	definiert URL vom Server-Anwendungsprogramm
`enctype`	Angabe eines MIME-Typs
`method`	spezifiziert die `get` oder `post`-Methode des HTTP-Protokolls

- Kindelemente des `<form>`-Elements

`<label>`	Beschreibungstext zu Eingabefeld	[WHATWG HTML 4.10.4]
`<input>`	Definition von Eingabefeld	[WHATWG HTML 4.10.5]
`<fieldset>`	Gruppierung von Formularelementen	[WHATWG HTML 4.10.15]

Attribute des **`<input>`**-Elements

`name`	definiert Variablennamen im `<form>`-Element	
`size`	definiert die Zeichenanzahl des Eingabefelds	
`type`	Typ des Eingabefelds: `text, radio, submit`	[SELFHTML]
`value`	definiert einen Default-Wert	

Bemerkungen:

- `<input>` ist ein „leeres Element“ (standalone tag), das keinen Inhalt hat und nur aus einem Tag besteht.
- HTML5 erweitert die Attribute des `<input>`-Elements. So ermöglicht `type` zusätzliche Datentypen mit den passenden Eingaben, `placeholder` eine adäquatere Gestaltung und `autofocus, pattern, required` eine leistungsfähigere Validierung. [MDN]

HTML

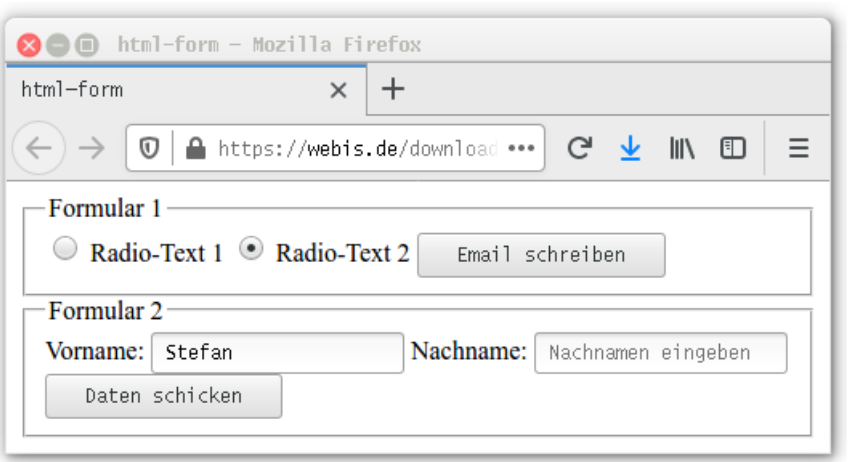
4.10 Forms: Beispiel

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>html-form</title>
  </head>
  <body>
    <form name="Webis" action="mailto:benno.stein@uni-weimar.de"
          method="post" enctype="text/plain">
      <fieldset>
        <legend>Formular 1</legend>
        <label><input type="radio" name="x" value="1">Radio-Text 1</label>
        <label><input type="radio" name="x" value="2" checked="checked">Radio-Text 2</label>
        <input type="submit" name="z" value="Email schreiben">
      </fieldset>
    </form>









  </body>
</html>
```

HTML

4.10 Forms: Beispiel

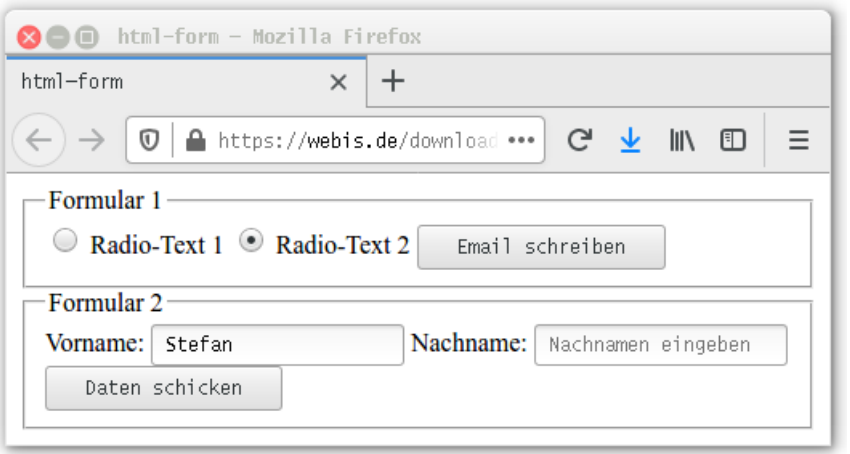
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>html-form</title>
  </head>
  <body>
    <form name="Webis" action="mailto:benno.stein@uni-weimar.de"
          method="post" enctype="text/plain">
      <fieldset>
        <legend>Formular 1</legend>
        <label><input type="radio" name="x" value="1">Radio-Text 1</label>
        <label><input type="radio" name="x" value="2" checked="checked">Radio-Text 2</label>
        <input type="submit" name="z" value="Email schreiben">
      </fieldset>
    </form>

  </body>
</html>
```



HTML

4.10 Forms: Beispiel [HTML-Form: html-Datei, Aufruf]

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>html-form</title>
  </head>
  <body>
    <form name="Webis" action="mailto:benno.stei
          method="post" enctype="text/plain">
      <fieldset>
        <legend>Formular 1</legend>
        <label><input type="radio" name="x" valu
        <label><input type="radio" name="x" valu
        <input type="submit" name="z" value="Ema
      </fieldset>
    </form>

    <form action="https://webtec.webis.de/cgi-bin/cgi-sample1.cgi">
      <fieldset>
        <legend>Formular 2</legend>
        <label for="field1">Vorname:</label>
        <input id="field1" name="vorname" type="text" value="Stefan">
        <label for="field2">Nachname:</label>
        <input id="field2" name="nachname" type="text" placeholder="Nachnamen eingeben">
        <input type="submit" name="z" value="Daten schicken">
      </fieldset>
    </form>
  </body>
</html>
```



4.10 Forms: Beispiel (Fortsetzung)

Erzeugung der HTTP-Response-Message mit dem Shell-Script `cgi-sample1.cgi`:

```
#!/bin/bash
echo "content-type: text/html"
echo ""   #Leerzeile gemäß HTTP-Protokoll.
echo "<!DOCTYPE html>"
echo "<html>"
echo "<head>"
echo "<meta http-equiv=\"content-type\" content=\"text/html; ...\">"
echo "<title>cgi-sample1</title>"
echo "</head>"
echo "<body>"
echo "<h3>Werte einiger CGI-Variablen</h3>"
echo "Installierte Server-Software: " $SERVER_SOFTWARE "<br>"
echo "Aufrufender Web-Browser: " $HTTP_USER_AGENT "<br>"
echo "Anfragemethode: " $REQUEST_METHOD "<br>"
echo "Query-String: " $QUERY_STRING "<br>"
echo "</body>"
echo "</html>"
```

4.10 Forms: Beispiel (Fortsetzung)

Erzeugung der HTTP-Response-Message mit dem Shell-Script `cgi-sample1.cgi`:

```
#!/bin/bash
echo "content-type: text/html"
echo ""   #Leerzeile gemäß HTTP-Protokoll.
echo "<!DOCTYPE html>"
echo "<html>"
echo "<head>"
echo "<meta http-equiv=\"content-type\" content=\"text/html; ...\">"
echo "<title>cgi-sample1</title>"
echo "</head>"
echo "<body>"
echo "<h3>Werte einiger CGI-Variable
echo "Installierte Server-Software:
echo "Aufrufender Web-Browser: " $HT
echo "Anfragemethode: " $REQUEST_ME
echo "Query-String: " $QUERY_STRING
echo "</body>"
echo "</html>"
```

[CGI: Script, Aufruf]

HTML

4.12 Scripting [WHATWG HTML 4.12]

Wichtige Elemente:

- `<script>` [SELFHTML]

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

- `<noscript>` [SELFHTML]

```
<noscript>
  Browser does not support JavaScript.
</noscript>
```

- `<canvas id="Demo"></canvas>` [SELFHTML 1, 2, SVG vs. Canvas] [w3schools]

```
<canvas id="Demo"></canvas>
<script>
  var canvas = document.getElementById("Demo");
  var canvasCtxt = canvas.getContext("2d");
  ...
</script>
```

HTML

Quellen zum Nachlernen und Nachschlagen im Web

- MDN. *HTML.*
  developer.mozilla.org/en-US/docs/Web/HTML
- SELFHTML e.V. *SELFHTML.*
  wiki.selfhtml.org
- WHATWG. *HTML: Living Standard, Developer's Edition.*
  html.spec.whatwg.org, html.spec.whatwg.org/dev
- W3C. *HTML5.2, Recommendation.*
  www.w3.org/TR/html52
- W3C. *HTML Wiki.*
  www.w3.org/wiki/Category:HTML
- W3 Schools. *HTML Reference.*
  www.w3schools.com/tags

HTML

Quellen zum Nachlernen und Nachschlagen im Web: Werkzeuge

- Flanders. *Web Pages That Suck.* (Web design flaws 2005-2015)
  www.webpagesthatsuck.com
- HTACG. *Tidy.* (Standardisieren und Säubern von HTML-Code)
  www.html-tidy.org
- W3C. *Markup Validation Service.*
  validator.w3.org