

## III. Dokumentsprachen

- ❑ Auszeichnungssprachen
- ❑ HTML
- ❑ Cascading Stylesheets CSS
- ❑ XML-Grundlagen
- ❑ XML-Schema
- ❑ Die XSL-Familie
- ❑ APIs für XML-Dokumente

# HTML

## Einordnung

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

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

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](#), [REC \(D\)](#)]

# 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](#), [REC \(D\)](#)]

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](#)]


## 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](#), [REC \(D\)](#)]
- 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 HTML5. Recommendation. Loslösung von SGML, neue Struktur- und  
2014 Multimedia-Elemente. [W3C [REC](#), [differences](#)] [\[Wikipedia \[article\]\(#\), \[figure\]\(#\)\]](#)
- 2015 XHTML5. XML-Serialisierung von HTML5. [W3C [NOTE](#)] [\[Stackexchange\]](#)
- 2017 HTML 5.2. Recommendation. [W3C [REC](#)]
- 2019 HTML. Living Standard. [WHATWG [living standard](#), [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. Mittlerweile versuchen das W3C und WHATWG, gemeinsam an HTML zu arbeiten. [W3C [REC 1.4](#)]
- ❑ Der Standardisierungsprozess der W3C ist formalisiert und spiegelt sich in den verschiedenen Leveln der veröffentlichten Reports wider. [W3C [reports](#)]

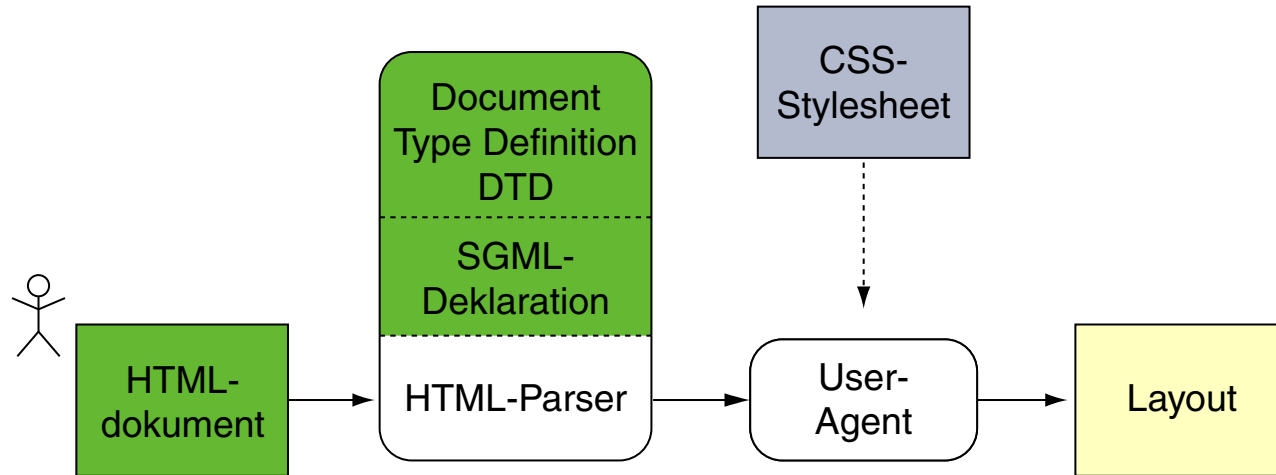
## Bemerkungen (HTML5) :

- ❑ 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

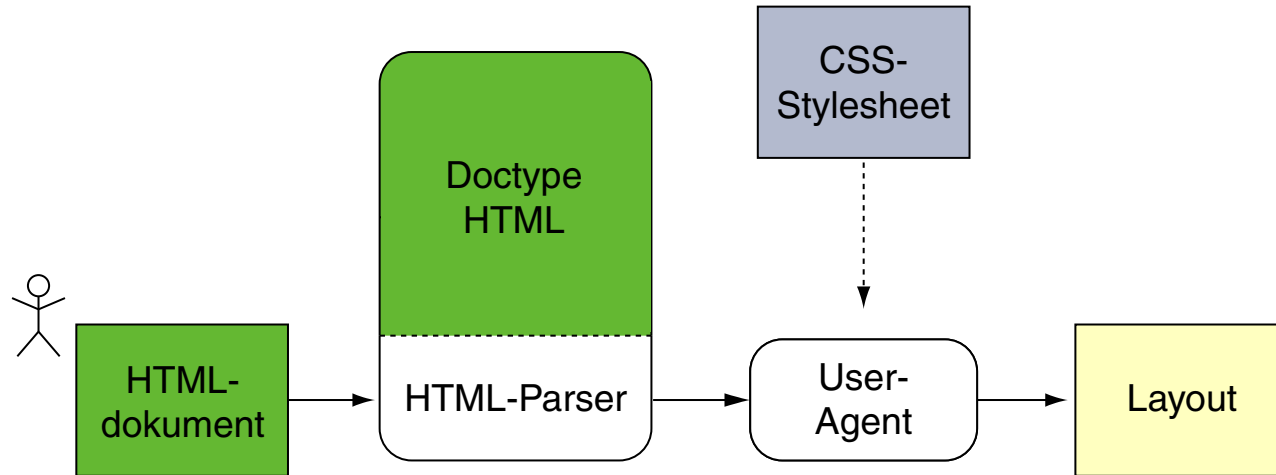
## HTML Dokumentenverarbeitung (HTML4)



Vergleiche hierzu die SGML Dokumentenverarbeitung.

# HTML

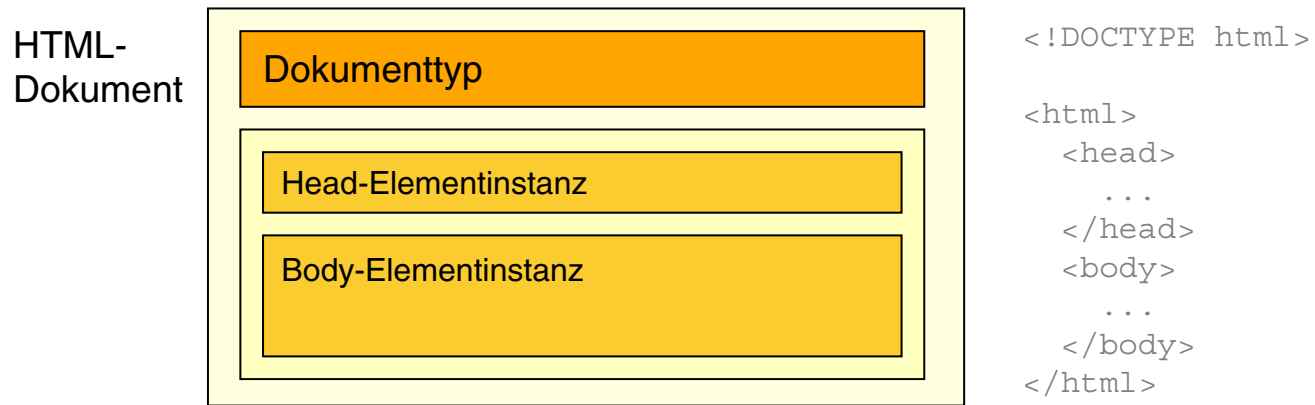
## HTML Dokumentenverarbeitung (HTML5)



Vergleiche hierzu die SGML Dokumentenverarbeitung.

# HTML

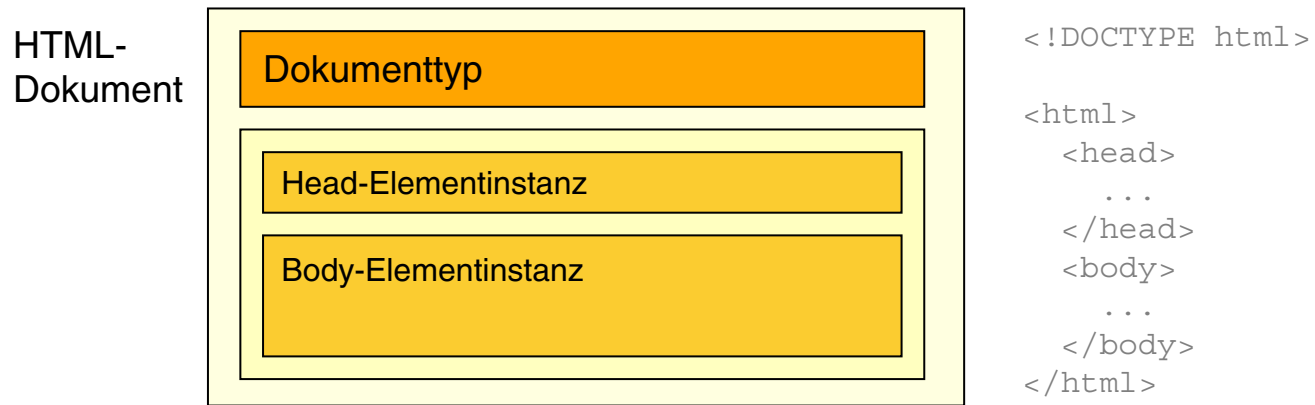
## Aufbau HTML-Dokument



- ❑ Das `<html>`-Element repräsentiert die Dokument-Wurzel. [w3c [REC 4.1](#)]
- ❑ Das `<head>`-Element repräsentiert die Meta-Daten. [w3c [REC 4.2](#)]
- ❑ Das `<body>`-Element repräsentiert den Dokumentinhalt. [w3c [REC 4.3](#)]
- ❑ Vergleiche hierzu die [XML-Dokumentstruktur](#).

# HTML

## Aufbau HTML-Dokument



- ❑ Das `<html>`-Element repräsentiert die Dokument-Wurzel. [w3c [REC 4.1](#)]
- ❑ Das `<head>`-Element repräsentiert die Meta-Daten. [w3c [REC 4.2](#)]
- ❑ Das `<body>`-Element repräsentiert den Dokumentinhalt. [w3c [REC 4.3](#)]
- ❑ Vergleiche hierzu die [XML-Dokumentstruktur](#).

Allgemeine Form einer HTML-Element**instanz** [WT:III [SGML](#)] :

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

# HTML

## Deklaration der DTD (HTML4)

HTML hat eine feste Dokumentstruktur, die bei 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-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 bei 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-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](#)] [W3C [REC 8.1](#)]

# HTML

## Inhaltsmodelle (*Content Models*) (HTML4)

Elementinstanzen innerhalb einer `<body>`-Elementinstanz gehören zu genau einer der folgenden zwei Kategorien [MDN [1](#), [2](#)] :

1. Block-Elemente

2. Inline-Elemente



# HTML

## Inhaltsmodelle (*Content Models*) (HTML4)

Elementinstanzen innerhalb einer `<body>`-Elementinstanz gehören zu genau einer der folgenden zwei Kategorien [MDN [1](#), [2](#)] :

### 1. Block-Elemente

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

Beispiele für Block-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>`

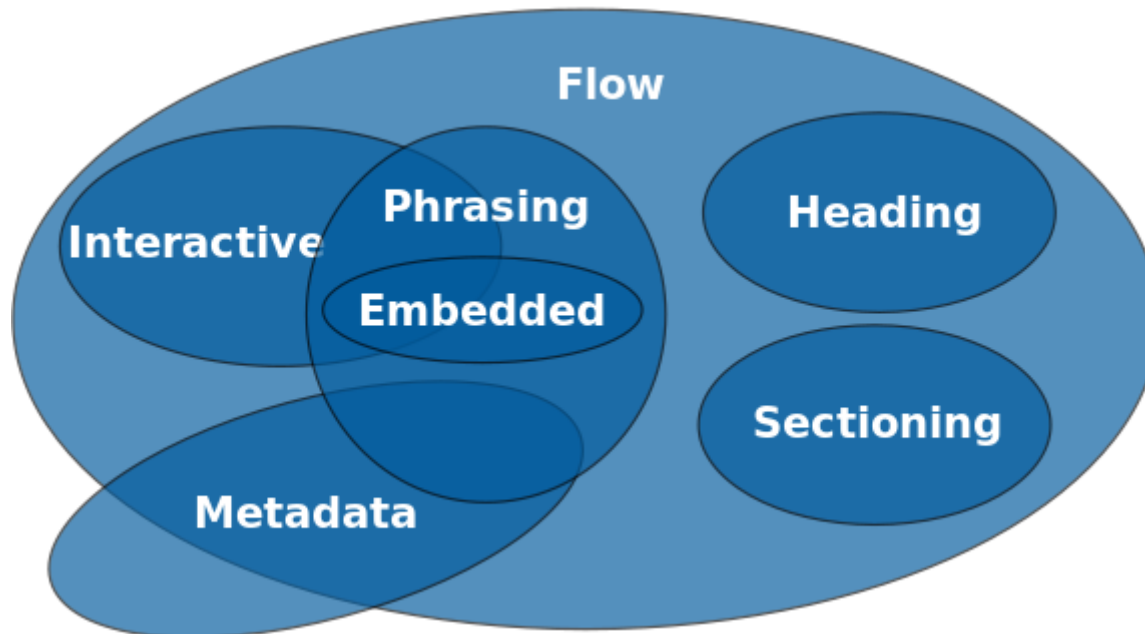
## Bemerkungen:

- ❑ Die Verarbeitung von Block-Elementen aus Sicht des Layout-Programms (beispielsweise mit einem Web-Browser) ist mit dem Verhalten von  $\text{\LaTeX}$  im  $\text{\textbackslash vmode}$  vergleichbar. Die Verarbeitung von Inline-Elementen ist mit dem Verhalten von  $\text{\LaTeX}$  im  $\text{\textbackslash hmode}$  vergleichbar.

# HTML

Inhaltsmodelle (*Content Models*) (HTML5) [W3C [REC 3.2.4](#)] [MDN]

Elementinstanzen innerhalb einer `<body>`-Elementinstanz fallen in mindestens eine der folgenden sieben Inhaltskategorien [W3C [REC 3.2.4.1](#)] :



## Bemerkungen:

- ❑ Bei HTML5 ist die syntaktische Aufteilung in Block- und Inline-Elemente durch eine an semantischen Überlegungen orientierte Aufteilung 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-Elemente (HTML4) ~ Flow Content [\[MDN\]](#)
  - Inline-Elemente (HTML4) ~ Phrasing Content [\[MDN\]](#)
- ❑ HTML5 verzichtet auf eine Reihe von (Block-)Elementen, die bei HTML4 in erster Linie zur Layout-Gestaltung dienen [\[w3schools\]](#) : `<center>`, `<frame>`, `<frameset>`, `<noframes>`

# HTML

Universalattribute (*Global Attributes*) [[W3C REC 3.2.5](#)] [[w3schools](#)] [[SELFHTML](#)]

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

1. Allgemeine

2. Zur Internationalisierung

# HTML

Universalattribute (*Global Attributes*) [[W3C REC 3.2.5](#)] [[w3schools](#)] [[SELFHTML](#)]

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

## 1. Allgemeine

---

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

---

## 2. Zur Internationalisierung

---

<code>dir</code>	gibt die Schreibrichtung für Text in der Elementinstanz an
<code>lang</code>	gibt die verwendete Landessprache (nach <a href="#">RFC 1766</a> ) an
<code>translate</code>	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 [W3C [REC 7.1.5.2](#)]

---

<code>onclick</code>	Ausführen von Script-Code beim Anklicken der Elementinstanz
<code>onkeydown</code>	Ausführen von Script-Code beim Herunterdrücken einer Taste
<code>onmouseover</code>	Ausführen von Script-Code beim Überfahren der Elementinstanz

---

### 4. Für eigene Daten (*Custom Data Attributes*) [W3C [REC 3.2.5.9](#)] [[w3schools](#)]

---

<code>data-*</code>	Semantik definiert durch Programmierer der Web-Site
---------------------	---

---

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

# HTML

## Organisation der HTML5-Spezifikation von W3C bzw. WHATWG

[W3C [REC](#)]

[ $\Delta$ WHATWG [living standard](#)]

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



# HTML

## Organisation der HTML5-Spezifikation von W3C bzw. WHATWG

[W3C [REC](#)]

[ $\Delta$ WHATWG [living standard](#)]

1. Introduction
2. Common infrastructure
3. Semantics, structure, and APIs

### 4. The elements of HTML

#### 5. Microdata

5. User interaction
6. Loading Web pages
7. Web application APIs

#### 9. Communication

#### 10. Web workers

#### 11. Web storage

8. The HTML syntax
9. The XML syntax
10. Rendering
11. Obsolete features
12. 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 Edits
- \* 4.7 Embedded content
- \* 4.8 Links
- \* 4.9 Tabular data
- \* 4.10 Forms
- 4.11 Interactive elements
- \* 4.12 Scripting
- 4.13 Common idioms
- 4.14 Disabled elements
- 4.15 Matching HTML elements

## Bemerkungen:

- ❑ 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*).
- ❑ Die Definition via DTD fokussierte auf die Inhalts- bzw. Markup-Sicht eines HTML-Elements; 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
- ❑ HTML4- versus HTML5-Spezifikation am Beispiel des <a>-Elements:

	Attribut-Semantik	
	Inhalts- bzw. Markup-Sicht	Interface-Sicht
HTML4	[W3C <a href="#">DTD</a> ]	—
HTML5	[W3C <a href="#">content</a> ]	[W3C <a href="#">DOM interface</a> ]
	[WHATWG <a href="#">content</a> ]	[WHATWG <a href="#">DOM interface</a> ]
	[WHATWG <a href="#">developer</a> ]	—

## 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](#)]
- ❑ Die Developer's Edition der WHATWG stellt die Markup-Sicht in den Vordergrund und ist deshalb besonders für die Autoren von Web-Seiten geeignet. [[WHATWG](#)]
- ❑ 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 [W3C [REC 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 [W3C [REC 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 Meta-Angabe im HTML-Dokument.

# HTML

## 4.3 Sections [W3C [REC 4.3](#), [summary](#)]

### □ Strukturelemente (HTML5) [[MDN](#)]

---

<code>&lt;article&gt;</code>	eigenständiger Inhalt, ggf. mit eigenem <code>&lt;header&gt;</code> und <code>&lt;footer&gt;</code>
<code>&lt;section&gt;</code>	(1) Gruppierung verschiedener <code>&lt;article&gt;</code> in Themen, (2) Einteilung <i>eines</i> <code>&lt;article&gt;</code> in Abschnitte. Typisch mit Überschrift für Inhalte
<code>&lt;nav&gt;</code>	Navigationsmenü oder andere Navigationsmöglichkeiten
<code>&lt;aside&gt;</code>	Gruppierung von verwandter Information mit Bezug zum Hauptinhalt
<code>&lt;header&gt;</code>	Kopfinformationen einer Website oder eines Artikels
<code>&lt;footer&gt;</code>	Fußzeile einer Website oder eines Artikels

---

# HTML

## 4.3 Sections [W3C [REC 4.3](#), [summary](#)]

### □ Strukturelemente (HTML5) [[MDN](#)]

---

<code>&lt;article&gt;</code>	eigenständiger Inhalt, ggf. mit eigenem <code>&lt;header&gt;</code> und <code>&lt;footer&gt;</code>
<code>&lt;section&gt;</code>	(1) Gruppierung verschiedener <code>&lt;article&gt;</code> in Themen, (2) Einteilung <i>eines</i> <code>&lt;article&gt;</code> in Abschnitte. Typisch mit Überschrift für Inhalte
<code>&lt;nav&gt;</code>	Navigationsmenü oder andere Navigationsmöglichkeiten
<code>&lt;aside&gt;</code>	Gruppierung von verwandter Information mit Bezug zum Hauptinhalt
<code>&lt;header&gt;</code>	Kopfinformationen einer Website oder eines Artikels
<code>&lt;footer&gt;</code>	Fußzeile einer Website oder eines Artikels

---

### □ Überschriftselemente [W3C [REC 4.3.6](#)] [[SELFHTML](#)]

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



# HTML

## 4.4 Grouping Content [W3C [REC 4.4](#)]

### □ Listen [[SELFHTML](#)]

---

<code>&lt;ol&gt;</code>	geordnete Liste
<code>&lt;ul&gt;</code>	ungeordnete Liste
<code>&lt;li&gt;</code>	Listeneintrag
<code>&lt;dl&gt;</code>	Definitionsliste
<code>&lt;dt&gt;</code>	Definitionsüberschrift
<code>&lt;dd&gt;</code>	Definitionseintrag

---

# HTML

## 4.5 Text-Level Semantics [W3C [REC 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 [W3C [REC 4.5](#)]

Unterscheidung von Textauszeichnungen hinsichtlich ihrer Konkretheit [[MDN](#)] :

### 1. Physische Auszeichnungen ([HTML4](#))

---

<code>&lt;i&gt;</code>	zeichnet einen Text als kursiv aus
<code>&lt;b&gt;</code>	zeichnet einen Text als fett aus
<code>&lt;u&gt;</code>	zeichnet einen Text als unterstrichen aus
<code>&lt;strike&gt;</code>	zeichnet einen Text als durchgestrichen aus
<code>&lt;tt&gt;</code>	zeichnet einen Text in Schreibmaschinenschrift aus

---

### 2. Logische Auszeichnungen (HTML4 und HTML5)

---

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

---

## Bemerkungen:

- ❑ “*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 bei HTML4 vordringlich zur physischen Auszeichnungen dienen [\[w3schools\]](#): `<basefont>`, `<big>`, `<dir>`, `<strike>`, `<tt>`  
Die weiteren HTML4-Elemente zur physischen Auszeichnung haben bei HTML5 eine explizite Semantik erhalten. Beispiele: `<i>`, `<b>`, `<u>` [\[W3C REC 4.5\]](#)
- ❑ Bei HTML5 dienen die Elemente `<ins>`, `<del>` zur Auszeichnung von sogenannten *Edits*.  
[\[W3C REC 4.6\]](#) [\[w3schools\]](#)
- ❑ Beispiele für physische Auszeichnungen in  $\text{\LaTeX}$  sind die Schriftschnitte und -gewichte:  
`\itshape`, `\bfseries`, `\fontfamily{phv}` `\fontsize{8}{0}` `\selectfont`  
Beispiele für logische Auszeichnungen in  $\text{\LaTeX}$  :  
`\em`, `\begin{quote} ... \end{quote}`

# HTML

## 4.7 Embedded Content [W3C [REC 4.7](#)]

### Wichtige Elemente:

❑ ``

[\[SELFHTML\]](#)

❑ `<iframe src="https://www.w3c.org"></iframe>`

[\[SELFHTML\]](#)

❑ `<audio src="sample.mp3" controls></audio>`

[\[SELFHTML\]](#)

❑ `<video data="introduction-video.mp4" controls></video>`

[\[SELFHTML\]](#)

❑ `<canvas id="painting" height="400" width="300"></canvas>`

[\[SELFHTML\]](#)

# HTML

## 4.8 Links [W3C [REC 4.8](#)] [[SELFHTML](#)]

Zur Definition von Hyperlinks dient das `<a>`-Element (*Anchor*). Als Inline-Element (HTML4) kann es keine Instanzen von Block-Elementen auszeichnen; der erlaubte Kontext (HTML5) ist Phrasing Content [W3C [REC 4.5.1](#)].

- Hyperlink

- Hyperlink-Ziel

# HTML

## 4.8 Links [W3C [REC 4.8](#)] [[SELFHTML](#)]

Zur Definition von Hyperlinks dient das `<a>`-Element (*Anchor*). Als Inline-Element (HTML4) kann es keine Instanzen von Block-Elementen auszeichnen; der erlaubte Kontext (HTML5) ist Phrasing Content [W3C [REC 4.5.1](#)].

### □ Hyperlink

`<a href="URI">`

Ziel ist durch *URI* definiert

Bsp.: `<a href="#Identifier">`

Ziel ist benannter Anker im aktuellen Dokument

Bsp.: `<a href="Path#Identifier">`

Ziel ist Anker in relativ referenziertem Dokument

Optionale Attribute des Anchor-Elements:

---

<code>title</code>	definiert den Mouse-Over-Text
<code>type</code>	MIME-Type des Zieldokuments
<code>download</code>	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.
- ❑ URIs, die mit einem Dokumentanker #*Identifier* abschließen, werden auch als *Fragment-Identifier* bezeichnet, weil sie ein Dokument nicht als Ganzes, sondern abschnittsgenau adressieren.
- ❑ Bei HTML4 kann das a-Element in Kombination mit dem name-Attribut als Hyperlink-Ziel verwendet werden. Bei HTML5 ist im a-Element das name-Attribut nicht mehr erlaubt; das id-Attribut kann (wie bei jedem Element) verwendet werden. [w3schools [a](#), [name](#)]



# HTML

## 4.9 Tabular Data [W3C [REC 4.9](#)] [[SELFHTML](#)]

### □ Elemente

---

<code>&lt;table&gt;</code>	Tabelle
<code>&lt;caption&gt;</code>	Tabellenüberschrift
<code>&lt;colgroup&gt;</code>	Spaltengruppe
<code>&lt;col&gt;</code>	Tabellenspalte
<code>&lt;tbody&gt;</code>	Tabellenkörper
<code>&lt;thead&gt;</code>	Tabellenkopf
<code>&lt;tfoot&gt;</code>	Tabellenfuß
<code>&lt;tr&gt;</code>	Tabellenzeile
<code>&lt;td&gt;</code>	einzelne Zelle
<code>&lt;th&gt;</code>	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](#)]

## Bemerkungen:

- ❑ Die Attribute zur Tabellengestaltung wie `align`, `bgcolor`, etc., werden bei HTML5 nicht mehr unterstützt. [\[w3schools\]](#)

# HTML

## 4.10 Forms [W3C [REC 4.10](#)] [[SELFHTML](#)]

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

- ❑ **Attribute** des `<form>`-Elements [W3C [REC 4.10.3](#)]

- ❑ **Kindelemente** des `<form>`-Elements

# HTML

## 4.10 Forms [W3C [REC 4.10](#)] [[SELFHTML](#)]

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

### ❑ **Attribute** des `<form>`-Elements [W3C [REC 4.10.3](#)]

---

<code>action</code>	definiert URI vom Server-Anwendungsprogramm
<code>enctype</code>	Angabe eines MIME-Typs
<code>method</code>	spezifiziert die <code>get</code> oder <code>post</code> -Methode des HTTP-Protokolls

---

### ❑ **Kindelemente** des `<form>`-Elements

---

<code>&lt;fieldset&gt;</code>	Gruppierung von Formularelementen
<code>&lt;input&gt;</code>	Definition von Eingabefeld
<code>&lt;label&gt;</code>	Beschreibungstext zu Eingabefeld

---

# HTML

## 4.10 Forms [W3C [REC 4.10](#)] [[SELFHTML](#)]

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

### ❑ **Attribute** des `<form>`-Elements [W3C [REC 4.10.3](#)]

---

<code>action</code>	definiert URI vom Server-Anwendungsprogramm
<code>enctype</code>	Angabe eines MIME-Typs
<code>method</code>	spezifiziert die <code>get</code> oder <code>post</code> -Methode des HTTP-Protokolls

---

### ❑ **Kindelemente** des `<form>`-Elements

---

<code>&lt;fieldset&gt;</code>	Gruppierung von Formularelementen
<code>&lt;input&gt;</code>	Definition von Eingabefeld
<code>&lt;label&gt;</code>	Beschreibungstext zu Eingabefeld

---

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

---

<code>name</code>	definiert Variablennamen im <code>&lt;form&gt;</code> -Element
<code>size</code>	definiert die Zeichenanzahl des Eingabefelds
<code>type</code>	Typ des Eingabefelds: <code>text</code> , <code>radio</code> , <code>submit</code> [ <a href="#">SELFHTML</a> ]
<code>value</code>	definiert einen Default-Wert

---

## Bemerkungen:

- ❑ 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 [\[Ausführung\]](#)

```
<!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>

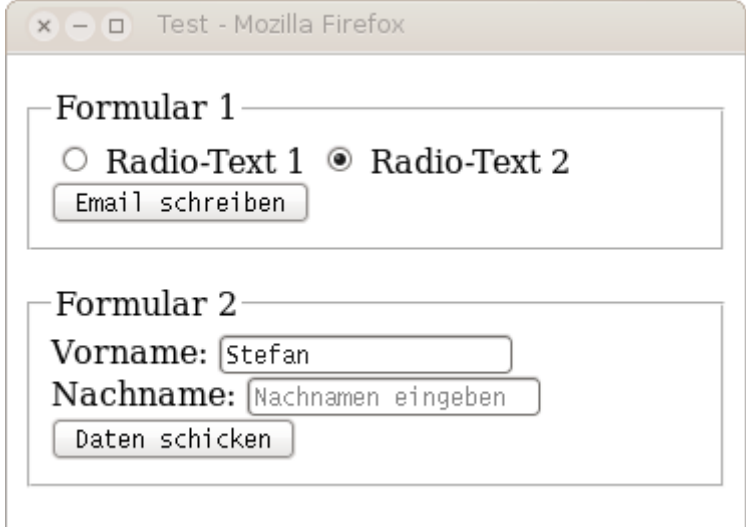
  </body>
</html>
```

# HTML

## 4.10 Forms: Beispiel [\[Ausführung\]](#)

```
<!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>
```



The screenshot shows a Mozilla Firefox browser window titled "Test - Mozilla Firefox". The rendered HTML form is displayed, consisting of two sections. The first section, titled "Formular 1", contains two radio buttons: "Radio-Text 1" (unselected) and "Radio-Text 2" (selected). Below the radio buttons is a submit button labeled "Email schreiben". The second section, titled "Formular 2", contains two text input fields: "Vorname:" with the value "Stefan" and "Nachname:" with the placeholder text "Nachnamen eingeben". Below these fields is a submit button labeled "Daten schicken".



# HTML

## 4.10 Forms: Beispiel [\[html-form.html\]](#)

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

    <form action="http://webis16.medien.uni-weimar.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>
```

Test - Mozilla Firefox

Formular 1

☐ Radio-Text 1 ☒ Radio-Text 2

Email schreiben

Formular 2

Vorname:

Nachname:

Daten schicken

# HTML

## 4.10 Forms: Beispiel (Fortsetzung)

Erzeugung der HTTP-Response-Message mit einem Shell-Script:

```
#!/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>"
```

# HTML

## 4.10 Forms: Beispiel (Fortsetzung)

Erzeugung der HTTP-Response-Message mit einem Shell-Script:

```
#!/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>"
```

cgi-sample1 - Mozilla Firefox

### Werte einiger CGI-Variablen

Installierte Server-Software: Apache/2.2.22 (Ubuntu)

Aufrufender Web-Browser: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:37.0) Gecko/20100101 Firefox/37.0

Anfragemethode: GET

Query-String: vorname=Stefan&nachname=&z=Daten+schicken

[cgi-sample1.cgi [Source](#), [Ausführung](#)]

# HTML

## 4.12 Scripting [W3C [REC 4.12](#)]

### Wichtige Elemente:

- ❑ `<script>`  
    `document.write("Hello world.")`  
    `</script>`
- ❑ `<noscript>`  
    Browser does not support JavaScript.  
    `</noscript>`
- ❑ `<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](https://developer.mozilla.org/en-US/docs/Web/HTML)
- ❑ SELFHTML e.V. *SELFHTML*.  
[wiki.selfhtml.org](https://wiki.selfhtml.org)
- ❑ WHATWG. *HTML: Living Standard, Developer's Edition*.  
[html.spec.whatwg.org](https://html.spec.whatwg.org), [html.spec.whatwg.org/dev](https://html.spec.whatwg.org/dev)
- ❑ W3C. *HTML5.2, Recommendation*.  
[www.w3.org/TR/html52](https://www.w3.org/TR/html52)
- ❑ W3C. *HTML Wiki*.  
[www.w3.org/wiki/Category:HTML](https://www.w3.org/wiki/Category:HTML)
- ❑ W3 Schools. *HTML Reference*.  
[www.w3schools.com/tags](https://www.w3schools.com/tags)

# HTML

## Quellen zum Nachlernen und Nachschlagen im Web: Werkzeuge

- ❑ Flanders. *Web Pages That Suck*. (Web-Design)  
[www.webpagesthatsuck.com](http://www.webpagesthatsuck.com)
- ❑ W3C. *HTML Tidy*. (standardisieren und säubern von HTML-Code)  
[www.html-tidy.org](http://www.html-tidy.org), [www.w3.org/People/Raggett/tidy](http://www.w3.org/People/Raggett/tidy)
- ❑ W3C. *Markup Validation Service*.  
[validator.w3.org](http://validator.w3.org)