

HTML5

Neue Konzepte in HTML5

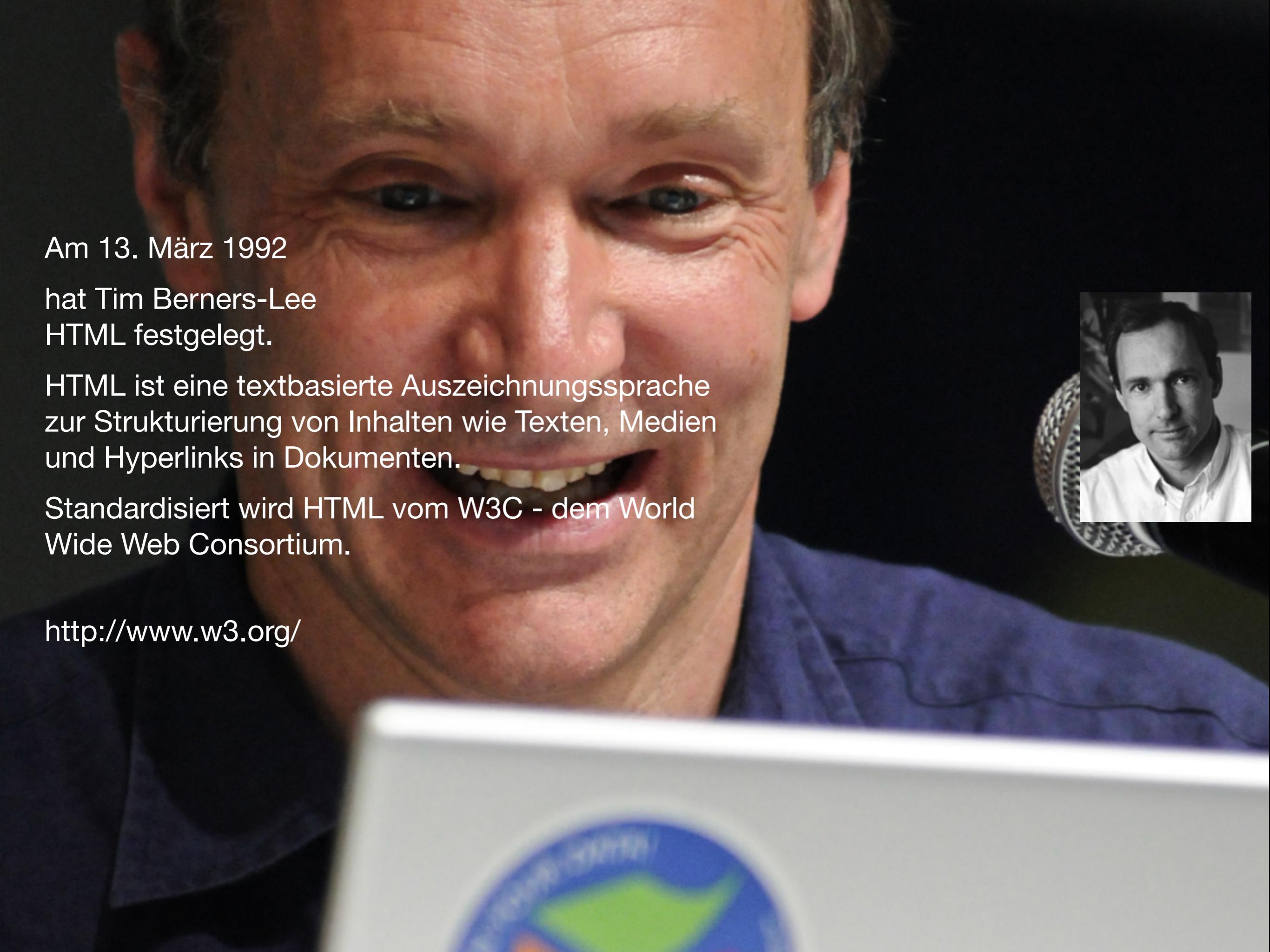
HTML5 Basisdokument

XHTML5 Basisdokument

5

Warum gibt es HTML?

Wie HTML entstanden ist, wer es erfunden hat
und wer sich heute darum kümmert.



Am 13. März 1992
hat Tim Berners-Lee
HTML festgelegt.

HTML ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalten wie Texten, Medien und Hyperlinks in Dokumenten.

Standardisiert wird HTML vom W3C - dem World Wide Web Consortium.

<http://www.w3.org/>



Die Geschichte von HTML

1992	HTML	2004	Web Applications 1.0
1994	HTML 2 (Netscape)	2005	AJAX
1995	Livescript	2010	HTML5 Draft
1996	HTML 3.2, CSS 1, JavaScript	2014	HTML5 Recommendation
1997	HTML 4, HTML 4.01	2016	HTML 5.1 Recommendation
1998	CSS 2	2022 HTML5 ist Standard!	
2000	XHTML 1.1		
2002	Tableless Web Design		

HTML5

HTML5 erweitert die 1992 entwickelte Hypertext Markup Language um eine Vielzahl von neuen Möglichkeiten, insbesondere für die Entwicklung dynamischer und mobiler Web-Anwendungen.

Der erste Entwurf zu HTML5 stammt aus dem Jahr 2004 und hieß Web Application 1.0. Er wurde von der WHATWG, der Web Hypertext Application Technologiy Working Group entwickelt.

Vom W3C erschien 2008 erstmals ein Working Draft für HTML5 und 2010 die aus damaliger Sicht erste vollständige Beschreibung.

<http://www.w3.org/TR/2010/WD-html5-20100624/>

Ursprung von HTML5

WHATWG -
Web Hypertext Application Technology Working
Group
2004 - Web Applications 1.0
<http://whatwg.org/html5>

W3C - World Wide Web Consortium
2007 - Working Draft
<http://www.w3.org/TR/2010/WD-html5-20100624/>

Ian Hickson



Ian "Hixie" Hickson war bis 2012 Autor von HTML5 und (u.a.) von Web Application 1.0. Er setzt sich für Webstandards ein und hatte schon bei der Entwicklung von CSS eine entscheidende Mitwirkung. Er ist Coautor von CSS 2.1.

Hickson arbeitete bei Netscape, Opera und ist heute für Google tätig.

HTML5 Projektlead heute

Robin Berjon, W3C



Steve Faulkner, The Paciello Group



Travis Leithead, Microsoft



Erika Doyle Navara, Microsoft

Edward O'Connor, Apple Inc.

Silvia Pfeiffer



Struktur, Syntax und Semantik

Wie man mit HTML Dokumente baut.

Semantische Strukturierung

HTML dient als **Auszeichnungssprache** dazu, einen Text semantisch zu strukturieren.

Die semantische Struktur beschreibt
das **Layout**
und den **Inhalt**.

No Design please!
Formatiert wird mit Cascading Style Sheets (CSS).

Container - Knoten - Elemente

```
<html> ... </html>
```

Ein HTML Element besitzt Attribute

```
<html  
    xmlns="http://www.w3.org/1999/xhtml"  
    xml:lang="de" lang="de"  
> ... </html>
```

```
<a href="link_zur_datei.html">Linktext</a>
```

Zwei Pflichtelemente einer HTML-Seite

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

Verschiedene Text- und Absatzauszeichnungen

<p>Ein Textabsatz, der ein
betontes Wort enthält.</p>

Eine Textzeile,
 die hier fortgesetzt wird.

Ein validierbares HTML 4 Dokument

```
<!DOCTYPE HTML  
PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">  
<html>  
  <head>  
    <title>Titel der Webseite</title>  
    <!-- Evtl. weitere Kopfinformationen -->  
  </head>  
  <body>  
    <p>Inhalt der Webseite</p>  
  </body>  
</html>
```

HTML Struktur-Elemente

Beinhaltet

html

head, body

head

title, meta, link, object, script, style, base

body

noscript, div

noscript *inline | block*

article *inline | block*

section *inline | block*

nav *inline | block*

div *inline | block*

HTML Struktur-Elemente

Beinhaltet

div

inline | block

h1

inline

p

inline

ol, ul

li

dl

dt, dd

table

caption, colgroup, thead, tfoot, tbody

form

inline | block (außer form)

fieldset

inline | block (außer form)

HTML Struktur-Elemente

Beinhaltet

div

inline | block

a

inline, außer a

img

null

canvas

null

audio

null

video

null

map

area

object

param, *inline | block*

br

null

HTML Struktur-Elemente

Beinhaltet

<i>null</i>	Kein Content, Tag mit schließendem Slash
<i>text</i>	Unicode Text, HTML Entities
block	structural block ol, ul, dl, table, tr, thead, tfoot, tbody, colgroup, col
	multi-purpose block div, li, dd, td, th, form, noscript
	terminal block h1, p, dt, caption, address, blockquote
<i>inline</i>	inline-semantic Vermischter Text mit keinem oder mehreren Elementen (s.u.)
	inline-flow br , bdo
	inline-block Replaced Elements und Form Controls

HTML Struktur-Elemente

		Beinhaltet
inline-semantic	importance	span, em, strong
	phrase	a, cite, code, kbd, samp, var
	word	abbr, dfn, cite
	char	sup, sub
inline-flow		br, bdo
inline-block	replaced	img, obj, embed, iframe, audio, video, canvas, svg
	controls	input, textarea, select, button, label, video (mit controls)

Beispiele für Texte und Auszeichnungen

< p > Inhalt der Webseite </ p >
Text mit Zeilen-

umbruch
< strong > wichtiger Text </ strong >
< em > betonter Text </ em >
< u > Unterstrichener Text </ u >
< h1 > Titel 1 </ h1 > ... < h6 > Titel 6 </ h6 >

Das Anchor Tag - a

```
<a href="link_zur_datei.html">Linktext</a>
```

```
<a href="link_zur_datei.html" target="_blank">  
  Link in ein neues Fenster  

```

```
<a href="#">Linkdummy ohne Funktion</a>
```

```
<a href="mailto:mail@domain.de">  
  Link ins Mailprogramm  

```

```
<a href="javascript:void();"  
  onclick="doFunction();">Linktext</a>
```

Der benannte Anchor

```
<a href="#top">nach oben</a>  
<a name="top">Erster Absatz</a>
```

HTML ist komplex - Ausgelagerte Spezifikationen

Viele der ursprünglich ein HTML5 enthaltenen Technologien wurden in eigene Spezifikationen ausgelagert. Entweder in der HTML5 Working Group oder auch in anderen Arbeitsgruppen:

HTML Microdata - HTML WG

HTML Canvas 2D Context - HTML WG

HTML5 Web Messaging - Web Apps WG

Web Workers - Web Apps WG

Web Storage - Web Apps WG

The WebSocket API - Web Apps WG

The WebSocket Protocol - IETF HyBi WG

Server-Sent Events - Web Apps WG

WebRTC - WebRTC WG

WebVTT - W3C Web Media Text Tracks CG

Spezifikationen zur Erweiterung von HTML5

HTML+RDFa - RDFa WG

DOM Parsing and Serialization - Web Apps WG

Shadow DOM - Web Apps WG

Web Intents - Web Apps WG / Device APIs WG

Polyglot Markup: HTML-Compatible XHTML Documents -
HTML WG

HTML5: Techniques for providing useful text alternatives -
HTML WG

HTML Editing APIs - HTML Editing APIs CG

HTML Media Capture - Device APIs WG

Media Capture and Streams - Device APIs WG / WebRTC WG

Media Fragments URI - Media Fragments WG

Encrypted Media Extensions - HTML WG

Media Source Extensions - HTML WG

many rel value specifications registered at the link type
registry - Microformats

In HTML5 integrierte externe Spezifikationen.

SVG - SVG WG

MathML - Math WG

WAI-ARIA - PFWG

Literatur

<http://w3.org>

<http://www.w3.org/TR/2010/WD-html5-20100624/>

<http://www.w3.org/TR/2012/WD-html51-20121217/>

<http://dev.w3.org/html5/decision-policy/html5-2014-plan.html>

<http://whatwg.org>

<http://developers.whatwg.org/>

<http://www.whatwg.org/specs/web-apps/current-work/multipage/>

Das Boxmodel

Die Darstellungsgrösse
von HTML-Elementen
richtig berechnen

Das Content Modell von HTML 4

FLOW

BLOCK

```
<p><em>Betonung</em></p>
```

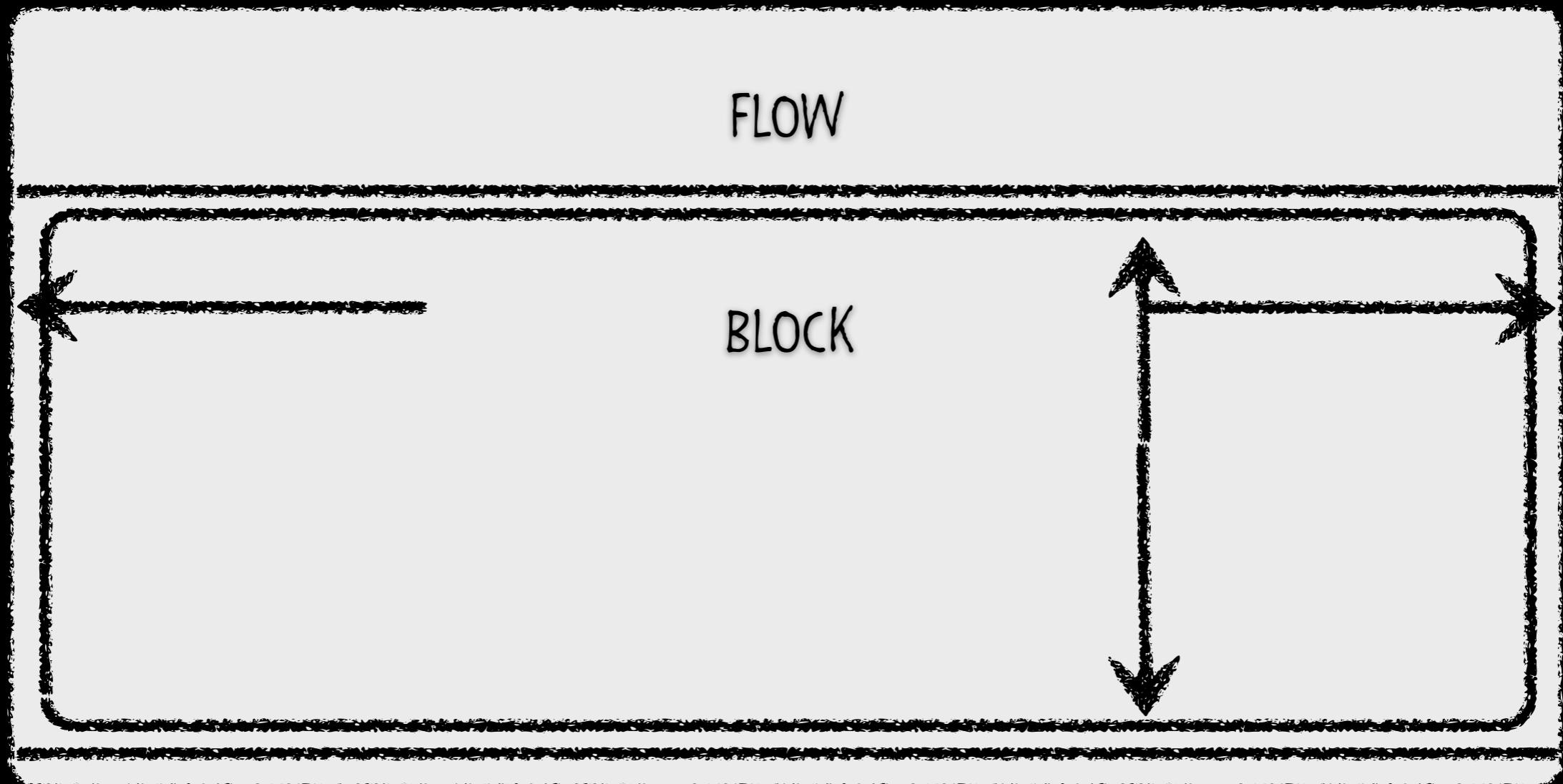
INLINE

Die Inhalte im html-Element "fließen" von oben nach unten

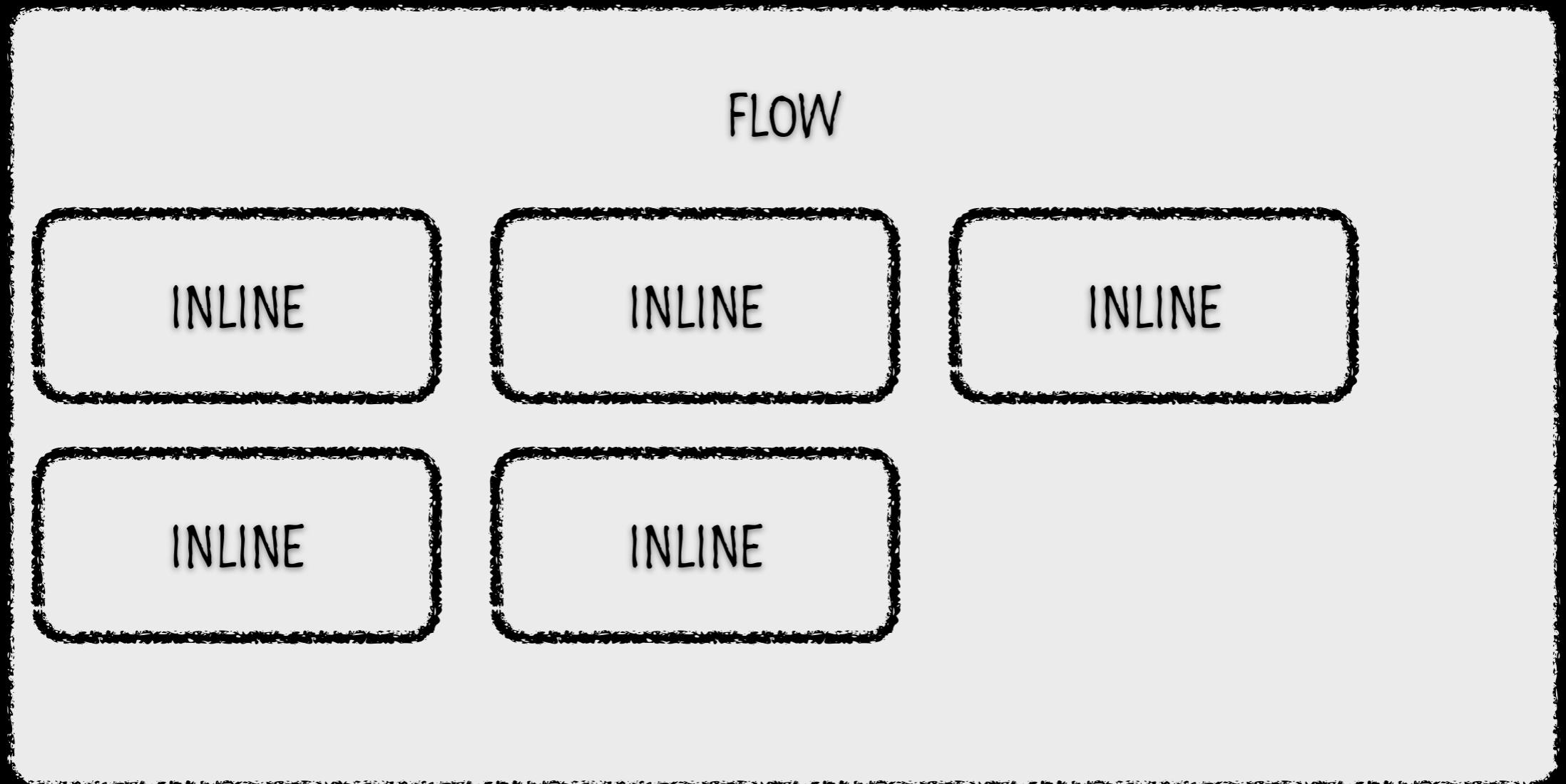
FLOW



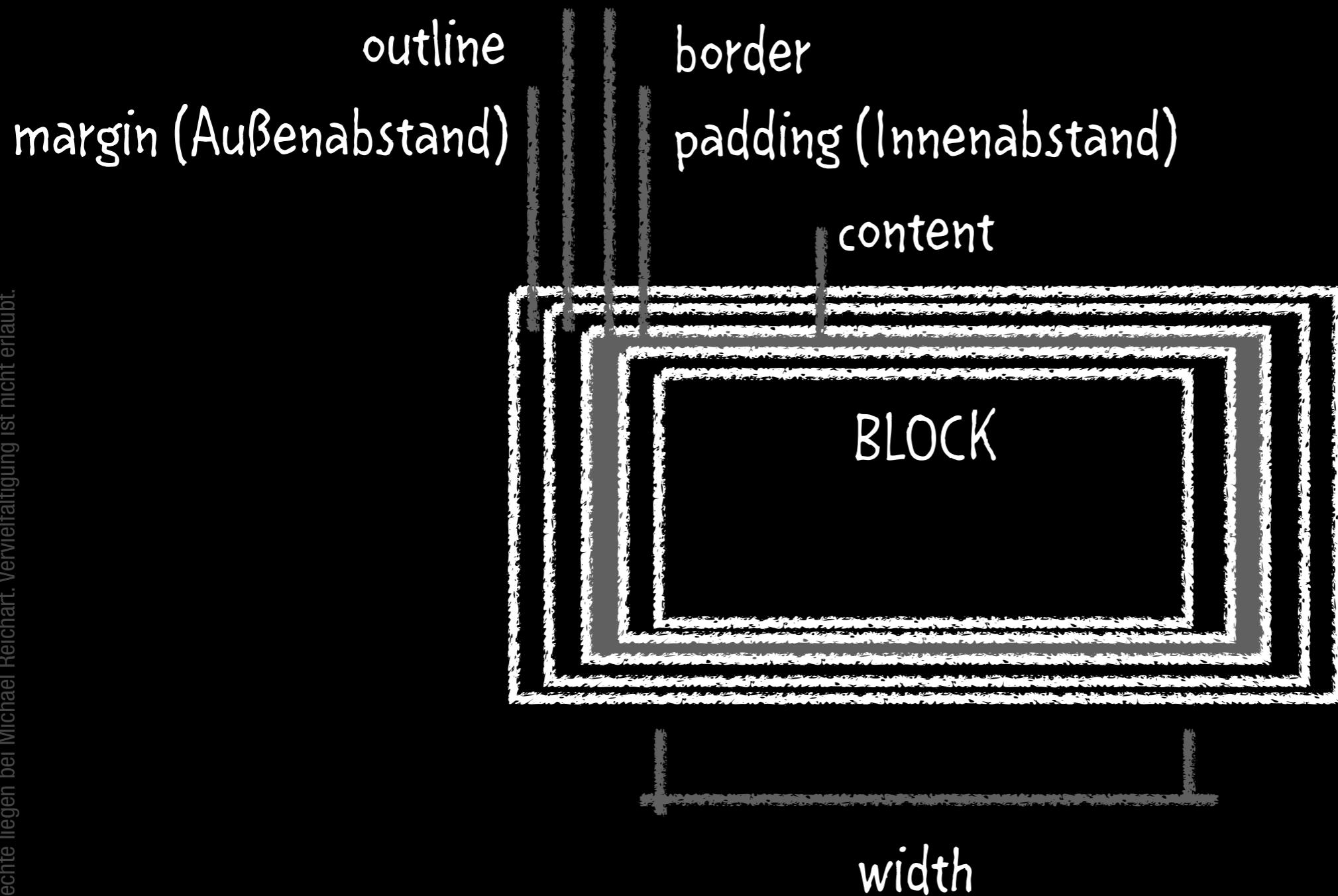
Blockelemente übernehmen die volle Breite des Parentelements und besitzen einen Umbruch nach oben und unten. Sie besitzen eine Höhe und eine Breite und Abstände.



**Inlineelemente fließen im Text mit.
Sie haben keine Breite und Höhe.**



Ein Blockelement besitzt Dimensionen und Abstände



Ein Inlineelement besitzt nichts besonderes.

Ein Inline-Block - Element

Ein Inlineblock - Element verhält sich wie ein Inlineelement, es fliesst im Text mit.

Aber hat ein Höhen- und ein Breitenattribut.

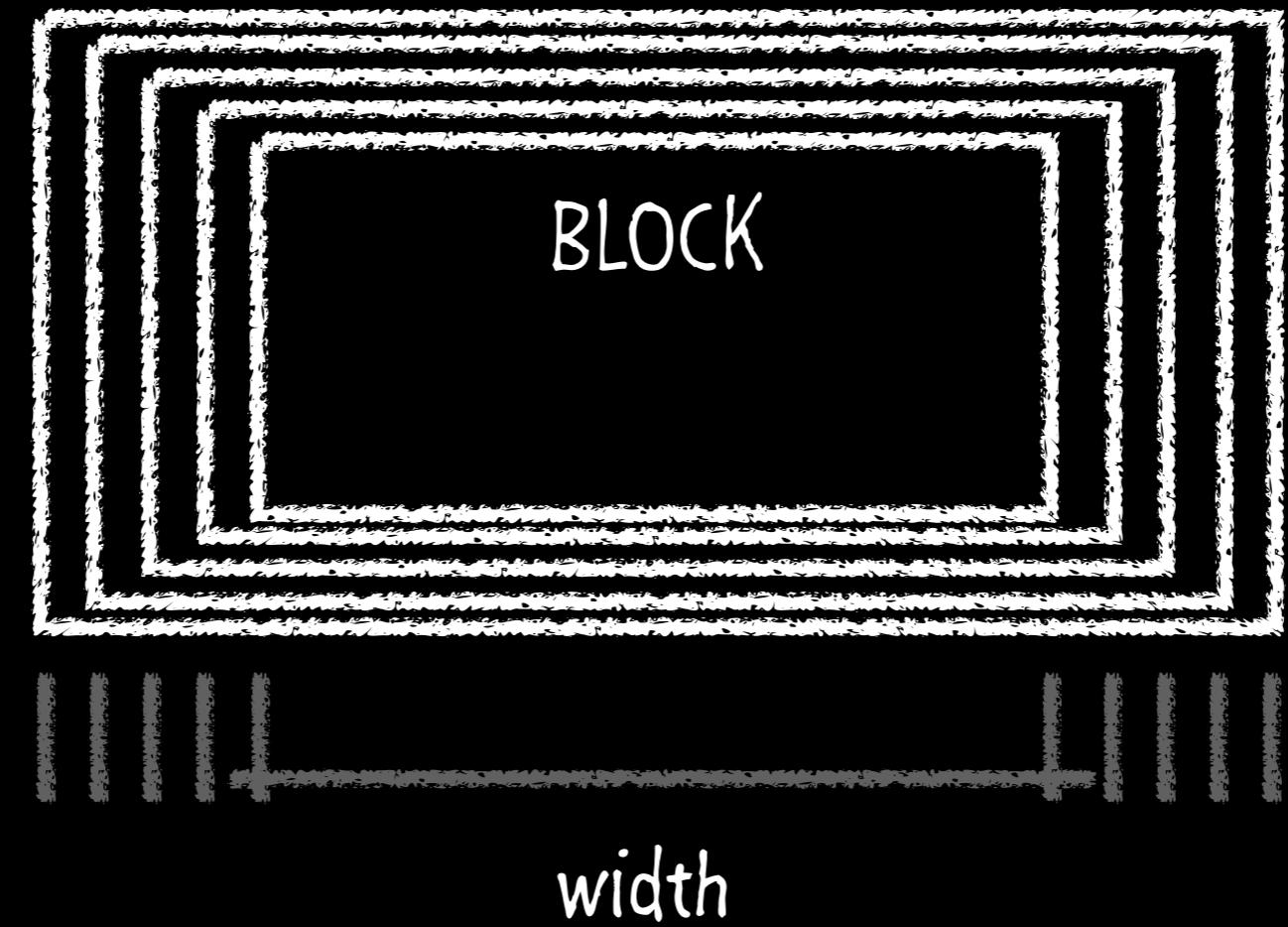
In HTML5 gibt es auch ein **Table** - Element.
Es besitzt ein bereits gesetztes Höhenattribut.

Dimensionsberechnung nach dem Contentbox Model

Breite =
width (content)
+ padding
+ border
+ outline
+ margin
= Elementbreite

width : 100px;
border: 1px;
padding: 1em;

Elementebreite: 134px

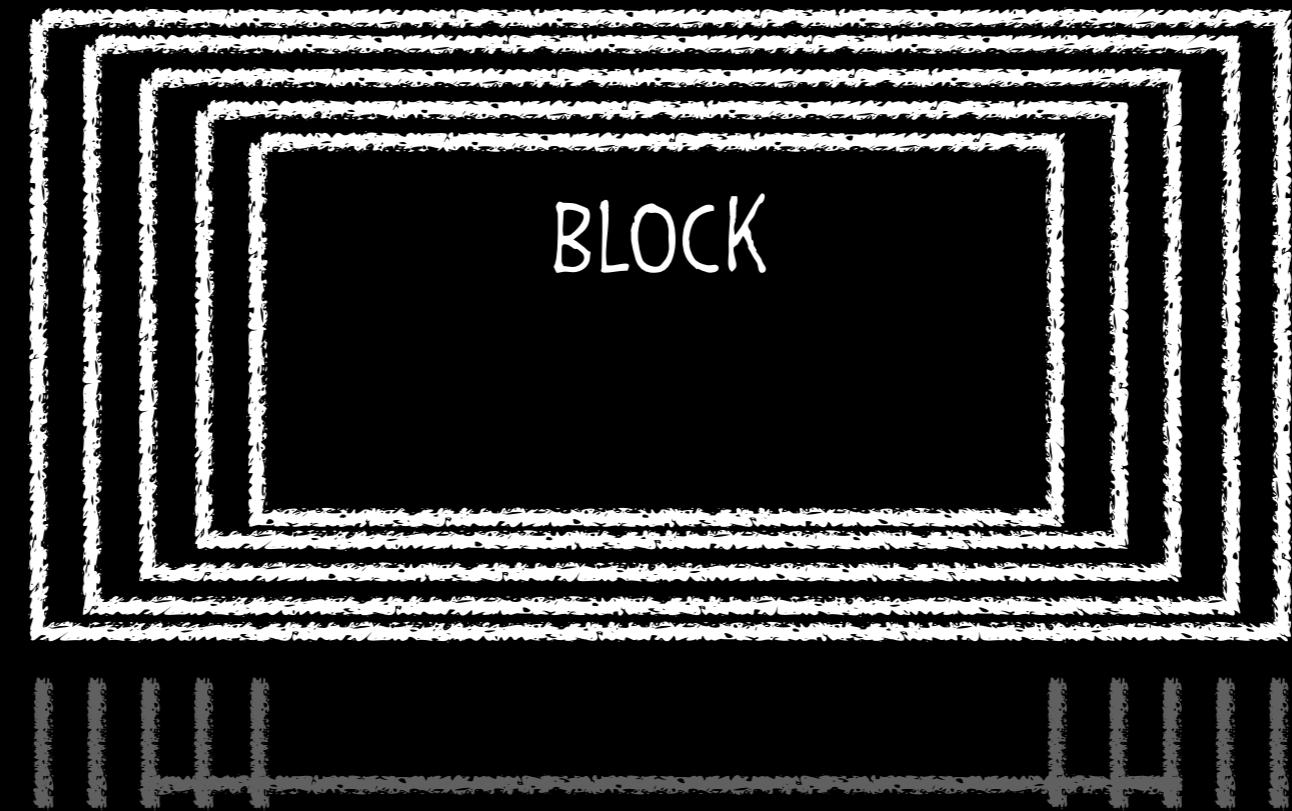


Dimensionsberechnung nach dem Borderbox Model

Breite =
width (content, padding, border)
+ outline
+ margin
= Elementbreite

width : 100px;
border: 1px;
padding: 1em;

Elementbreite: 100px



width

HTML Doctypes

Kein DOCTYPE, Quirksmode und Boxmodel

Ohne Angabe eines DOCTYPES verwenden Browser einen eigenen Defaultwert. IE verfällt dabei in den Quirksmode, der sich der Ausgabe von anderen Browsern deutlich unterscheidet.

Ursache hierfür ist beispielsweise ein anderes Boxmodel, das der IE verwendet.

Doctypes

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

>

Noch vor dem HTML Tag wird ein Dokumententyp angegeben. So kann der Browser entscheiden, welches Validierungs- und Verarbeitungsmodell er verwenden soll. Die DTD's sind beim W3C hinterlegt.

Wird kein DOCTYPE angegeben, so entscheiden die Browser selbst.

Im Internet Explorer < 8 wird dann im sogenannten Quirksmode gerendert.

Die !DOCTYPEs Transitional und Loose

Sie wurden geschaffen, um alten Seiten (in der Regel vor 2002) eine Überlebenschance zu geben, ohne auf eine Validierung zu verzichten.

Beide enthalten ein reduziertes Regelwerk, das Freiräume für applikationsspezifisches HTML lässt.

Für neue HTML 4.01 Seiten sollten die beiden DOCTYPEs nicht ohne triftigen Grund verwendet werden.

Microsoft HTML:

```
<IMG SRC=pfad/bild.JPG ALT="">
```

Der !DOCTYPE Strict

Strict enthält die meisten HTML Regeln und liefert ein einheitliches HTML!

Unter HTML 4.01 sollte strict für neue Seiten verwendet werden.

Die Regeln für HTML 4.01 beinhalten NICHT, das Tags geschlossen werden müssen, das bleibt XHTML vorbehalten.

 statt

Ebenso dürfen Attributnamen auch ohne Werteangabe verwendet werden.

<input type="checkbox" checked>

statt <input type="checkbox" checked="checked" />

Der leere DOCTYPE HTML aus HTML5

<!DOCTYPE HTML>

HTML5 verzichtet auf derlei Unterscheidungen und geht offenbar wieder einen Schritt zurück, in dem es lediglich einen leeren DOCTYPE verwendet.

<INPUT TYPE=CHECKBOX CHECKED /> wäre eine (mehr oder weniger) gültige Schreibweise in HTML5. Es ist in Zukunft den Browsern überantwortet, aus dem Quelltext etwas sinnvolles zu machen.

Aus Gründen der Sprachstandardisierung werden aber die HTML Strict oder XHTML Regeln beibehalten.

Der XHTML 1.0 DOCTYPE

```
<!DOCTYPE HTML PUBLIC  
"-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-  
strict.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">
```

Für XHTML wird ausserdem die Namespace
Definition angegeben.
Auch hier gibt es den DOCTYPE transitional.

XHTML 1.1 DOCTYPE

```
<!DOCTYPE HTML PUBLIC  
"-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/  
xhtml11.dtd"  
>  
<html xmlns="http://www.w3.org/1999/xhtml">
```

Er enthält keine Untertypen mehr und wird voraussichtlich der letzte XHTML DOCTYPE sein. XHTML 2.0 wird derzeit nicht mehr weiterverfolgt.

XHTML ist well-formed.

Tags müssen geschlossen sein, Elemente ohne Inhalt werden mit einem abschließenden / vermerkt.

<meta ... />,

Attribute dürfen nur als name="wert" - Paar geschrieben sein, Kurzschreibweisen gibt es nicht
checked="checked"

Tags und Attribute werden einheitlich klein geschrieben,

Die Einbindung oder Kombination mit anderen XML Sprachen, zum Beispiel MathXML oder SVG in ein und demselben Dokument ist möglich.

HTML5 und XML -> XHTML5

XHTML kann weiterhin verwendet werden, dazu wird lediglich kein DOCTYPE mehr angegeben und stattdessen ein Namensraum angegeben. „<http://www.w3.org/1999/xhtml>“.

Der Medientyp z. B. „application/xhtml+xml“ oder „application/xml“ wird umgangssprachlich als XHTML5 bezeichnet.

HTML5 vs. XHTML5?

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

Der Doctype verweist darauf, dass keine Version mehr ist, sondern schlicht ein HTML Dokument.

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head> ... </head>
  <body> ... </body>
</html>
```

HTML5 lässt aber auch XML-Schreibweisen zu!

Da allerdings HTML5 definiert, dass auch HTML-Elemente dem Namensraum „<http://www.w3.org/1999/xhtml>“ angehören, sind die Unterschiede gering.

Der Medientyp für HTML5 ist „text/html“,
der Dokumententyp schlicht `<!DOCTYPE html>`

Was ist also das Beste?

Verwenden Sie den leeren Doctype für alle HTML Dokumente.

Schreiben Sie dann nach einem Regelwerk 'strict' oder 'xhtml'.

Es besteht aus heutiger Sicht kein Bedarf, die Regeln wegzulassen oder aufzuweichen; sie haben sich in der letzten Dekade 'für ein globales Teamplay' als sinnvoll erwiesen.

Ein XHTML5 Basisdokument

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head> ... </head>
  <body>
    <header> ... </header>
    <nav>...</nav>
    <section id="content" data-role="">
      <article>...</article>
      <article>...</article>
      <article>...</article>
    </section>
    <footer> ... </footer>
  </body>
</html>
```

Der Begriff HTML5

HTML Wortschatz

+ CSS

+ Javascript APIs

= HTML5

HTML5 ist daher keine Versionsnummer,
sondern eher ein "Markenname".

HTML5 und alte Versionen?

HTML5 ist eine abwärtskompatible Verbesserung und Erweiterung von HTML und im Großen und Ganzen kompatibel zu XHTML1 und HTML 4.01.

HTML5 ist eine Evolution mit aus der Praxis abgeleiteten, neuen Elementen.

Wann ist HTML5 einsatzbereit?

HTML5 wird wahrscheinlich erst 2022 Standard geworden sein.

Es sind ursprünglich keine weiteren HTML Versionen mehr vorgesehen; stattdessen ist HTML5 in ständiger Weiterentwicklung.

Allerdings existiert ja nun doch schon eine Spezifikation mit der Nummerierung 5.1 ...

HTML5 heute schon verwenden

Der mobile Markt funktioniert heute bereits mit HTML5.

Im deutschen Consumermarkt sind neue Browerversionen verbreitet.

Aber: in deutschen Unternehmen ist immer noch der Internet Explorer in den Versionen 6, 7 oder 8 Standard. Keiner davon unterstützt HTML5.

Microsoft arbeitet mit Hochdruck daran, Internet Explorer 10 auch für ältere Windows Betriebssysteme zur Verfügung zu stellen.

Andere Hardwarehersteller setzen unter Windows CE beispielsweise eigene Implementationen auf Basis von Webkit ein.

"Vorsicht" bei Frameworks

Bei Javascript Frameworks, wie zum Beispiel jQuery, ist in nächster Zeit darauf zu achten, dass sie keine unnötigen Simulationen mittlerweile vorhandener nativer Technologien aus HTML5 und CSS 3 "mit sich schleppen".

jQuery entgegnet der Überalterung durch eine Zwei-Wege-Architektur. jQuery 1.9.x wird für "alte" Browser weiterentwickelt.

jQuery 2.x für neue HTML5-Browser.

HTML5 und CSS Neuerungen

HTML5:

Sectioning und das neue Content Model, Phrasing und Semantic Markup

Medieneinbindung für Video/Audio

Übersicht über die Erweiterungen in Webforms 2.0

Link und Anchor Relations

Viele weitere HTML5 Javascript APIs

CSS 3:

Typographie, Grafik und Animationen

Selektoren API

Media Queries

Javascript:

Neue Selectoren

Neue Objekte

Neue Events

Browserkompatibilitäten

Gute Browser

Chrome 3+

Firefox 3+

Opera 10+

Safari 3+

Internet Explorer 9+

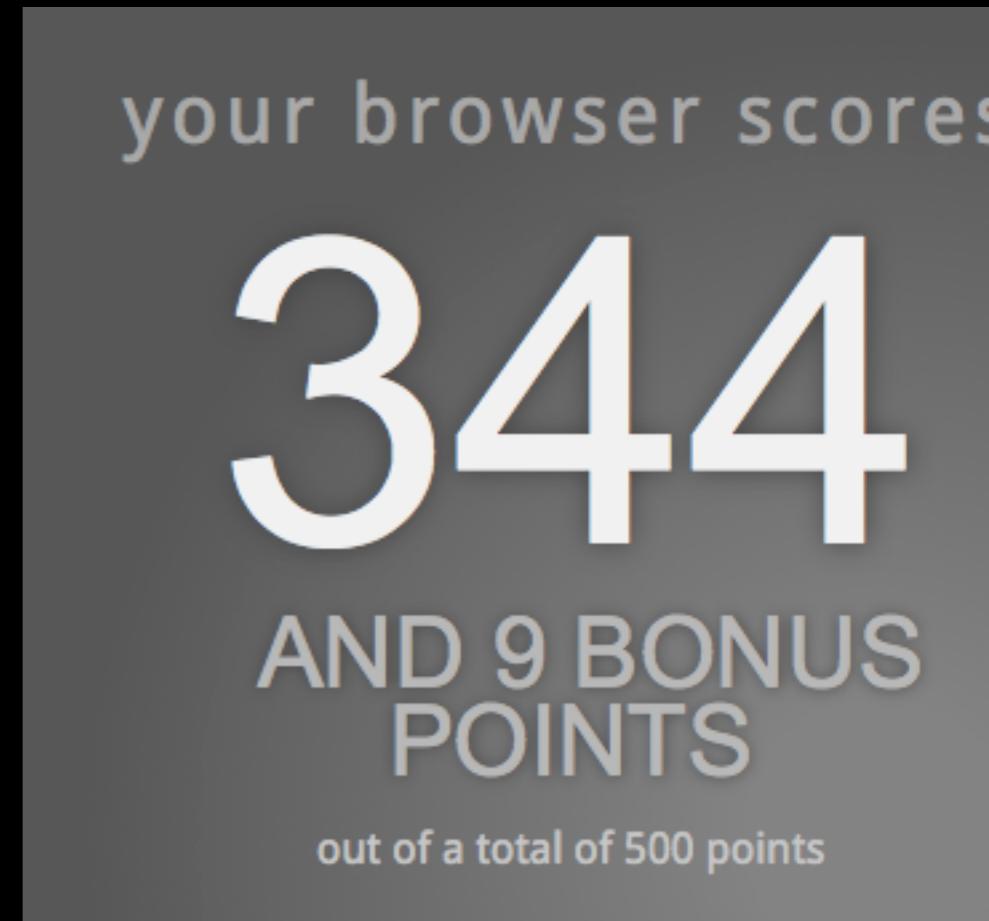
Schlechte Browser

Internet Explorer Version < 9

Den Browser fragen...

Aber: die Guten können auch nicht alles. Mobile Browser sind heute weiter, als deren Desktopvarianten.

<http://html5test.com> und <http://css3test.com> liefern einen Index, wieviel und was ein Browser jeweils umsetzt.



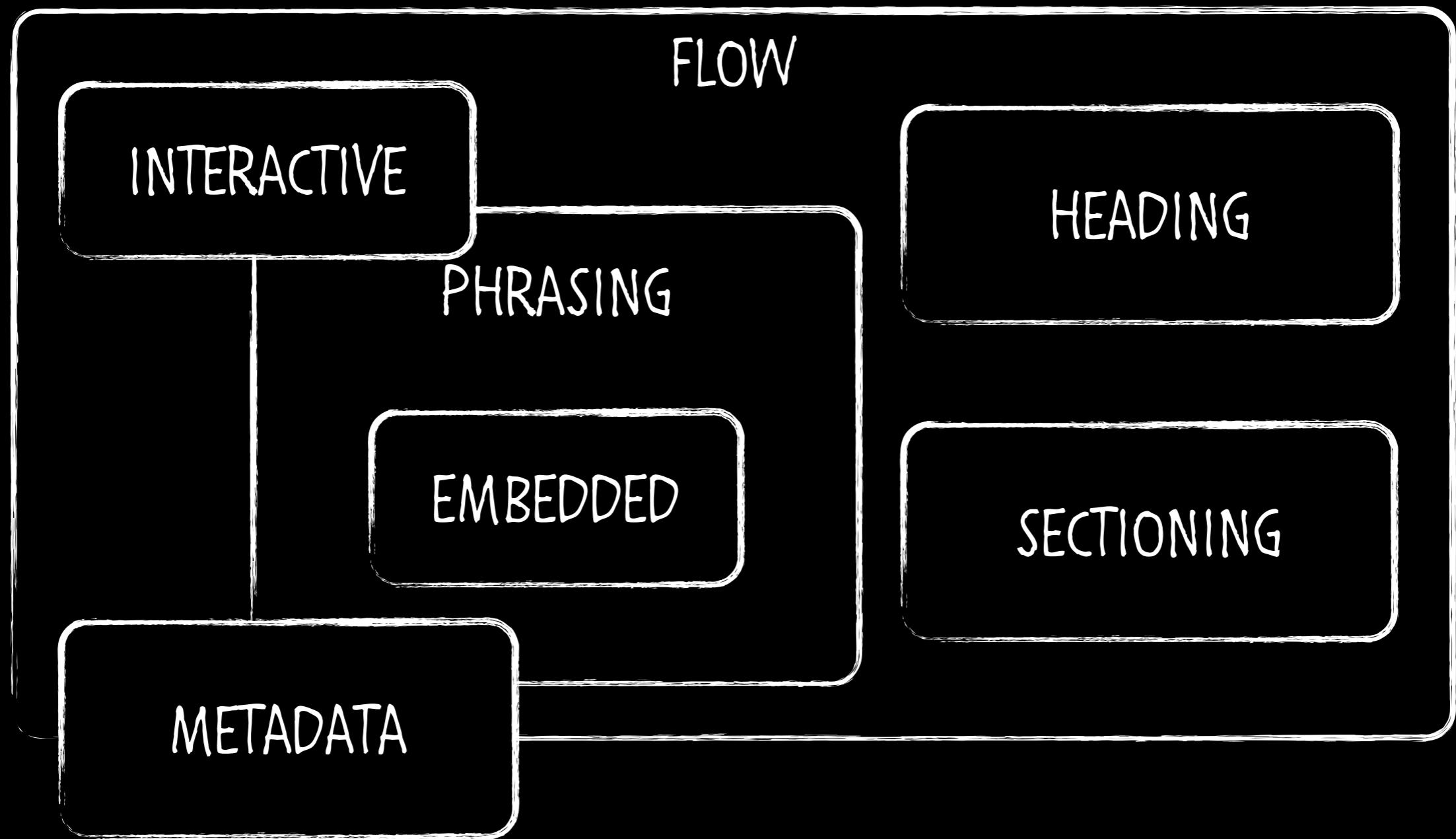
HTML5 Semantik

Aufbau der Dokumentenstruktur
Textstile und Textsemantik
Mikrodaten

Ein neues Content Modell für HTML5

Jetzt inhalteorientiert, (und nicht mehr technisch).

Das Content Modell von HTML 5



Metadaten und Scripting

base, command (5.1), link, meta, noscript, script,
style, title

Textsemantik (5.0)

span, a, rt, rp, dfn, abbr, q, cite, em, time, var, samp,
i, b, sub, sup, small, strong, mark, ruby, ins, del, kbd,
bdo, wbr, code

Flow Content (5.1)

a, abbr, address, article, aside, audio, b, bdi, bdo,
blockquote, br, button, canvas, cite, code,
command, datalist, del, details, dfn, dialog, div, dl,
em, embed, fieldset, figure, footer, form, h1, h2, h3,
h4, h5, h6, header, hgroup, hr, i, iframe, img, input,
ins, kbd, keygen, label, map, mark, math, menu,
meter, nav, noscript, object, ol, output, p, pre,
progress, q, ruby, s, samp, script, section, select,
small, span, strong, sub, sup, svg, table, textarea,
time, u, ul, var, video, wbr, text

area, (if it is a descendant of a map element),
style, (if the scoped attribute is present)

Palpable Content (5.1)

a, abbr, address, article, aside, b, bdi, bdo, blockquote, button, canvas, cite, code, details, dfn, div, em, embed, fieldset, figure, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, i, iframe, img, ins, kbd, keygen, label, map, mark, math, meter, nav, object, output, p, pre, progress, q, ruby, s, samp, section, select, small, span, strong, sub, sup, svg, table, textarea, time, u, var, video

audio (if the controls attribute is present),

dl (if the element's children include at least one name-value group),

input (if the type attribute is not in the hidden state),

menu (if the type attribute is in the toolbar state or the list state),

ol (if the element's children include at least one li element),

ul (if the element's children include at least one li element),

text that is not inter-element whitespace

Inhalte gruppieren

hr, br, p, figcaption, figure, pre, div, ol, ul, li,
blockquote, dl, dt, dd

Formulare

fieldset, form, meter, select, legend, optgroup, label,
option, datalist, input, output, keygen, textarea,
button, progress

Dokumentensektionen

body, aside, address, h1, h2, h3, h4, h5, h6, section,
header, nav, article, footer, hgroup

article, aside, nav, section

Heading Sektionen

h1, h2, h3, h4, h5, h6, hgroup

Tabellen

table, tr, td, th, tbody, thead, tfoot

Interaktive Elemente

button, details, embed, iframe, menu, command,
summary, details, keygen, label, select, textarea,
a, audio (if the controls attribute is present),
img, (if the usemap attribute is present),
input (if the type attribute is not in the hidden state),
menu (if the type attribute is in the toolbar state),
object (if the usemap attribute is present),
video (if the controls attribute is present)

Eingebetteter Inhalt

img, area, map, embed, object, param, source,
iframe, canvas, track, audio, video, device

Meta Data und Scripting

```
<head> <title> <meta> <base>  
<link> <style> <noscript> <script>
```

Metadata

Metadaten, die für das gesamte aktuelle HTML-Dokument gelten, können in HTML5 einerseits im Dokumentkopf vorkommen, andererseits aber auch im Dokumentkörper.

Im Kopf:

meta link base script style
 title

Jetzt auch im Body:

script noscript style title

script, style, title im Body?

Ajax-getriebene Dokumente laden Inhalte nach, zum Beispiel Skripte, aber auch zusätzliche CSS-Anweisungen. Diese werden in der Regel im body-Knoten eingefügt.

Das Element title bietet eine gute Möglichkeit, die Dokumentensektionen zu betiteln, ohne das dieser im Dokument erscheint. Dies ist auch für Maschinen nutzbar (Suchmaschinen, Screenreader, Bots).

```
<html>
  <head>...</head>
  <body>
    <section>
      <title>Content</title>
      ...
    </section>
    <script defer src="..."></script>
  </body>
</html>
```

Neuer Aufbau der Dokumentenstruktur

Tags für Dokumentensektionen

<header> <article> <footer> <nav> <section>
<aside>

Sectioning Elements

header, footer

können für die Seite, einer Sektion, Artikel, Navigationsgruppen und sonstige inhaltliche Gruppen verwendet werden.

article

Gruppierung eines inhaltlich unabhängig und abgeschlossen Contents.

main

Kennzeichnet den Hauptinhalt einer Seite oder einer Section.

aside

Verwandter oder weiterführender Inhalt oder Anmerkungen innerhalb eines Artikels oder einer Sektion.

nav

Navigationsgruppierung für Seitennavigationen (Nicht einzelne Links). Die Navigationsgruppe ist vollständig semantisch und kann header/footer etc. enthalten.

section

Eine Sektion trennt inhaltliche Blöcke. Steht sie im body-Kontext, kann sie als "Seite in der Seite" gesehen werden. Sie sollte nur verwendet werden, wenn es kein besser geeignetes semantisches Tag gibt.

<header>, <footer>

Kopfteil einer Sektion oder einer Seite
oder einer anderen Gruppe.

Das header-Element definiert den Kopfbereich für den aktuellen Abschnitt (z. B. in einem section-Element). Neben einer Überschrift kann es auch weitere Informationen (z. B. Versionsdaten) enthalten.

Das footer-Element definiert eine Fußzeile für den aktuellen Abschnitt (z. B. in einem article-Element) und gibt z. B. Auskunft über den Autor oder die Erstellungszeit des Textes.

<aside>

Nebenbeiinhalt, Marginalspalteninhalt.

Bezeichnet Inhalt, der thematisch zum umschließenden Inhalt passt, aber nicht direkt dazugehört. Ein Nachrichtentext könnte in einem aside-Element Querverweise zu ähnlichen Nachrichten beinhalten.

<main>

Kennzeichnet den Hauptinhalt einer Seite eines Artikels oder einer Sektion.

<nav>

Ein Navigationsbereich.

Dieses Element umschließt einen Navigationsbereich. Damit sind nicht nur Verweise zu anderen Seiten eines Webauftritts gemeint, sondern auch Verweise, die auf Abschnitte innerhalb eines Dokuments zeigen.

Eine Navigationgruppe mit Headergruppe und Linkliste

```
<nav>
  <header>
    <h4>Überschrift</h4>
    <time>2013</time>
  </header>
  <ul>
    <li><a href="">Linktext</a></li>
    <li><a href="">Linktext</a></li>
  </ul>
</nav>
```

<article>

Kennzeichnet einen inhaltlichen Artikel.

Das article-Element umfasst einen unabhängigen Abschnitt, der jedoch im Kontext zu den Vorfahren-Elementen steht. Damit sind unter anderem Nachrichtenartikel, Blog- oder Foreneinträge gemeint.

Eine Artikelgruppe mit verschiedenen Inhalten

```
<article>
  <header>
    <h4>Überschrift der Nav</h4>
    <address>Autorenangaben</address>
    <time>November 2012</time>
  </header>
  <p>Das ist der Text für den Artikel.</p>
  <figure>
    
    <figcaption>Bildunterschrift</figcaption>
  </figure>
  <aside>
    <h2>Weitere Links</h2>
    <nav>
      <ul>
        <li><a href="">Linktext</a></li>
        <li><a href="">Linktext</a></li>
      </ul>
    </nav>
  </aside>
</article>
```

<section>

Definiert eine inhaltliche Sektion.

Ein section-Element gruppiert thematisch zusammenhängenden Inhalt und besitzt meist einen Kopf- und Fußbereich. Eine typische Webseite wird section-Elemente ein Produkt-, ein Nachrichten- und ein Kontaktbereich verwenden.

Textstile und Textsemantik

Bekannte HTML Elemente mit neuer Bedeutung

Folgende semantische Elemente werden jetzt mehr ihrer ursprünglichen Bedeutung nach eingesetzt.
Die Änderungen sind zum Beispiel für Search Engine Optimizer wichtig.

<i>

Bisher kursiv (italic).

Jetzt Text, der üblicherweise kursiv geschrieben wird,
also beispielsweise Gedanken, veränderte
Stimmungslage, usw.

Bisher Fettschrift.

Kennzeichnet jetzt Text, der zwar stilistisch hervorgehoben werden soll, aber keine wichtigere Bedeutung als der umgebende Text hat.

Emphasis (Betonung)

Das em-Element kann nun verschachtelt werden, um die Betonung eines Textes zu verstärken.

Das strong-Element zeichnet einen Text als inhaltlich sehr wichtig aus.

<hr>

Bisher eine horizontale Linie.

Das <hr>-Element symbolisiert nun einen thematischen Umbruch zwischen Absätzen in einem Text.

Dabei kann es weiterhin wie eine Linie aussehen.
Muss aber nicht. (hr {border:0;}).

<small>

Bisher kleine Schrift.

Nun dient es nun der Auszeichnung von Kleingedrucktem, wie Rechtshinweisen oder zusätzlichen Angaben zu Preisen.

Tags, die nur der Formatierung dienen, fallen weg.

```
<applet> <acronym> <bgsound> <basefont> <big>  
<blink> <center> <dir> <font> <frame> <frameset>  
<isindex> <listing> <marquee> <noframes> <nobr>  
<s> <spacer> <strike> <tt> <u>
```

Anstelle von Frames werden CSS oder iframes verwendet.

Das <acronym> fällt weg, da es einen Sonderfall des <abbr> darstellt. Alles Andere besitzt semantisch keinen Nutzen.

Aber auch alle Attribute, die nur der Formatierung dienen, fallen weg.

align, color, size, width, height, hspace, vspace und margin

in Verbindung mit zum Beispiel <body>, <table>, <iframe>.

Aber nicht width und height des img-Tags. Denn hier sind das Eigenschaften des Bildes.

Der HTMLList muss hier gegebenenfalls selbst entscheiden, ob eine Angabe technisch benötigt wird.

```
<canvas width="100" height="100">  
</canvas> !!!
```

Neue semantische HTML Elemente

Die nachfolgenden neuen Elemente werden entweder für die Verbesserung der Dokumentensemantik verwendet, oder zur Bildung von Knotengruppierungen.

~~<hgroup>~~

Sammlung/Gruppierung von h1-h6 Elemente über eine Sektion eines Dokuments. Das erste Element repräsentiert die Gruppe in der Dokumentenoutline.

```
<hgroup>
    <h1>Heroüberschrift</h1>
    <h2>Unterüberschrift</h2>
</hgroup>
```

<mark>

Markierter Text.

Das mark-Element stellt hervorgehobenen Text dar.

Dabei geht es hier etwa um Text, der von einer Suchmaschine gefunden wurde und sichtbar markiert wird, sodass er schnell von Benutzern gefunden werden kann.

```
<mark>Fundstelle</mark>
```

```
mark {  
    background-color : #ffff00;  
    color : black;  
}
```

<ruby>, <rp>, <rt>

<ruby>

汉 <rp>(</rp><rt> hàn </rt><rp>)</rp>

字 <rp>(</rp><rt> zì </rt><rp>)</rp>

</ruby>

<ruby>

 <rp> Ruby Notiz für Nicht-Rubybrowser.

 <rt> Erklärung für eine Ruby Notiz.

</ruby>

Beispiel in Ruby-Browsern

hàn zì
汉 字

Abbildung in Nicht-Ruby-Browsern

汉(hàn) 字(zì)

<figure> und <figcaption>

Definiert eine Mediencontent Gruppe und deren Beschriftung.

Das figure-Element erlaubt es, ein bestimmtes Medium (z. B. Bilder, Videos etc.) mit einer Bildunterschrift zu versehen.

```
<figure>
  <video src="video.webm"></video>
  
  
  
  <figcaption>Bildunterschrift</figcaption>
</figure>
```

<wbr>, ­

Definiert einen "Wordbreak", einen Umbruch zwischen Worten, in denen der Browser bevorzugt umbrechen darf.

```
<p>
    To learn AJAX, you must be
    fa&shy;mi&shy;liar with the
    XML<wbr>Http<wbr>Request&nbsp;Object.
</p>
```

Im Kontext CSS3 Silbentrennung:

```
p {
    hyphens : auto; /*lang-Attribut verwenden*/
    word-break : break-all;
}
```

<address>

Kontaktinformationen einem aktuellen Artikel. Die Kontaktinformation bezieht sich auf den Besitzer oder Autor des Dokumentes.

```
<article>
  <address>
    <p>Magda Mustermann</p>
    <p>Bei Spielweg 12</p>
    <p>54321 Ort im Satz</p>
    <p><a href="mailto:mail@beispiel.de">
      mail@beispiel.de</a></p>
  </address>
  ...
</article>
```

<time>

Definiert eine Datums-/Zeitangabe.

Das time-Element wird verwendet, um Zeitangaben auszuzeichnen und Zeitangaben mit Metadaten auszustatten, sodass ein Browser damit z. B. einen interaktiven Kalender erzeugen kann.

Um <time>10:00 h</time> ist geöffnet.

```
<time datetime="2012-11-19">Today</time>
```

<picture>

Für die Platzierung von Bildern wird ein neues <picture>-Element eingeführt. Damit können auflösungsabhängig verschiedene Bildformate bereitgestellt werden.

```
<picture>
  <source media="(min-width: 45em)" srcset="large.jpg">
    <source media="(min-width: 32em)" srcset="med.jpg">
      
</picture>
```

```
<picture>
  <source srcset="/uploads/100-marie-lloyd.webp" type="image/webp">
  <source srcset="/uploads/100-marie-lloyd.jxr" type="image/vnd.ms-photo">
  
</picture>
```

Ebenfalls für auflösungsabhängig Bildformate kann das srcset-Attribut verwendet werden.

```

```

Das template Element

Es stellt ein **Vorlagenelement zur Wiederverwendung** innerhalb des HTML DOMs zur Verfügung. Das template Element ist im User Agent Stylesheet mit `display:none` versehen. Es muss vor seiner Verwendung aktiviert werden. Dies geschieht zweckmäßigerweise mit einer tiefen Kopie via `cloneNode()`.

```
<table>
<tr>
  <template id="cells-to-repeat">
    <td></td>
  </template>
</tr>
</table>
```

```
var t = document.querySelector('#cells-to-repeat');
t.content.querySelector('img').src = 'logo.png';
document.body.appendChild(t.content.cloneNode(true));
```

link relations

Das rel - Attribut weist die Beziehung zu einer Dateiverlinkung aus. Dies kann eine semantische oder auch eine funktionale Beziehung sein. Oft muss auch der MIME-Type angegeben werden.

```
<link rel="alternate" type="application/rss  
+xml" href="http://myblog.com/feed" />
```

```
<link rel="icon" href="/favicon.ico" />
```

```
<link rel="pingback" href="http://myblog.com/  
xmlrpc.php">
```

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

a relations

Auch für das Anchortag kann bei Links eine "relation" angegeben werden. Diese verweist zum Beispiel auf die Art des Inhalts, der verlinkt ist oder ergänzt eine funktionale Anweisung.

```
<a rel="archives" href="http://myblog.com/archives">old posts</a>
```

```
<a rel="external" href="http://notmysite.com">tutorial</a>
```

```
<a rel="license" href="http://www.apache.org/licenses/LICENSE-2.0">license</a>
```

```
<a rel="tag" href="http://myblog.com/category/games">games posts</a>
```

```
<a rel="nofollow" href="http://notmysite.com/sample">wannabe</a>
```

```
<a rel="tel" href="tel:+497117676123">+49 (0) 711 8 76 76 76-123 </a>
```

rel - Typen gehören mittlerweile zur Microformats Spezifikation.

Mehr Informationen zu den rel - Tags finden Sie unter
<http://microformats.org/wiki/existing-rel-values>

Mikrodaten

Einbinden maschinenlesbarer Informationen in
HTML-Dokumente.

Microdaten

Einbinden maschinenlesbarer Informationen in HTML-Dokumente.

Es wird eine Kompatibilität zu anderen Formaten wie dem Resource Description Framework (RDF) und der JavaScript Object Notation (JSON) angestrebt.

Der Standard "Microdata" ist nicht mehr in der Entwicklung, sonder hat sich für RDF entschieden.

Rich Snippets durch Microdaten

```
<div itemscope itemtype="http://schema.org/Event">
  <p>My name is <span itemprop="name">Neil</span>.</p>
  <p>My band is called <span itemprop="band">Four Parts Water</span>.</p>
  <p>I am <span itemprop="nationality">British</span>.</p>
</div>
```

band.txt

name Name des Künstlers

band Bandname

nationality Nationalität des Künstlers

Rich Snippets

Rich Snippets Testing Tool at <http://www.google.com/webmasters/tools/richsnippet>

Google webmaster tools

Rich Snippets

Help with:
[Documentation](#)
[Tips & Tricks](#)

Rich Snippets Testing Tool Beta

Rich Snippets allows you to enhance your Google search results by marking up web pages with Microformats, RDFa or Microdata.

Test your website

Enter a web page URL to see how it may appear in search results:
 [Preview](#)

Examples: [Urbanspoon](#), [LinkedIn](#)

Google search preview

[Pizza My Heart - Santa Cruz | Urbanspoon](#)
★★★★★ 10 reviews - Price range: Under \$10 per entree
Excerpt from the page will show up here. Excerpt from the page will show up here.
Excerpt from the page will show up here. Excerpt from the page will show up here.
www.urbanspoon.com/r/6/765421/restaurant/Pizza-My-Heart-Santa-Cruz - [Cached](#) - [Similar pages](#)

Note that there is no guarantee that a Rich Snippet will be shown for this page on actual search results. For more details, see the [FAQ](#).

Extracted Rich Snippet data from the page

hreview-aggregate
item hcard
fn = Pizza My Heart
org
organization.name = Pizza My Heart

"Data-" Attribute erfinden

Mit dem "data dash" Basis Attribut können Sie machen, was Sie möchten.

data-mein-eigenes-attribut

Oft ist es notwendig, in einem HTML Element Attribute zweckentfremdet zu verwenden. So wird das title-Attribut oft als eine zusätzliche Informationsebene in Tags oder als CSS-Hook angewandt.

Mit dem Prefix "data-" können eigene Attribute an beliebiger Stelle hinzugefügt werden:

```
<a href="" title="" data-tooltip="Mein  
Tooltiptext">...</a>  
<div id="header" data-section="page-header">...</  
div>
```

Diese lassen sich mit Selektoren ansprechen:

```
[data-section=page-header] { display : block; }  
document.querySelectorAll("[data-  
tooltip]").addEventListener( ... );
```

Webforms 2.0

```
<form> <fieldset> <label>  
<input> <select> <option> <optgroup> <textarea>  
<meter> <legend> <datalist> <output> <keygen>  
      <button> <progress>
```

Webforms 2.0 verbessern das Userinterface für Entwickler

Es wurden **neue Input Typen** und neue Attribute eingeführt.

Im Browser findet eine **integrierte Validierung** der Eingaben statt.

Es steht ein **Javascript Objekt** zur **Steuerung der Validierung** zu Verfügung.

Mit einer **Message API** können die Validierungsergebnisse ausgegeben werden.

Mit **CSS-Pseudoklassen** können Validierungen gestaltet werden.

Definierte Input Typen

```
<input type="email">  
<input type="url">  
<input type="text">  
<input type="search">  
<input type="date"> und Subtypen  
<input type="color">  
<input type="range">  
<input type="tel">  
<input type="number">
```

Datentyp für Datums- und Zeitangaben

```
<input type="day">  
<input type="month">  
<input type="year">  
<input type="datetime">  
Jahr, Monat, Tag, Stunde, Minute, Sekunde und Millisekunde nach UTC (Weltzeit)  
<input type="date">  
Jahr, Monat, Tag ohne Zeitzone  
<input type="week">  
<input type="time">  
Stunde, Minute, Sekunde und Millisekunde ohne Zeitzone
```

Stateklassen für Eingabeelemente

```
<style>
  :invalid { box-shadow : 0px 0px 5px red; }
  :valid { box-shadow : 0px 0px 5px green; }
  :focus { box-shadow : 0px 0px 5px blue; }
</style>
```

Ein Eingabefeld für Suchanfragen gestalten

```
<form action="http://www.google.com/search">
  <label>Google:</label>
  <input type="search" name="q">
  <input type="submit" value="Suchen">
</form>
```

```
input[type=search] {
  height          : 1.5em;
  vertical-align : center;
  border-radius   : 0.75em;
  background      : url(src/ico/lupe.png);
  background-repeat : no-repeat;
  left            : 0.25em; top    : 0.25em;
}
```

Range

```
<input type="range"
      min="14" max="100"
      step="1" value="14"
      onchange="showValue(this.value)"
>
<output>18</output>

<script>
function showValue(val) {
  document.querySelector('output').innerHTML = val;
}
</script>
```

Eingabefeld für Farben in sRGB 8-bit

```
<input type="color">
```

Dropdown Liste für eine Kombobox

```
<input type="email" id="contacts" list="contactList">  
  
<datalist id="contactList">  
  <option value="michael@zenbox.de" label="Michael">  
  <option value="peter@zenbox.de" label="Peter">  
</datalist>
```

Neue Attribute für Eingabefelder

placeholder- Ein Platzhaltertext für den User

```
<label>Runner:</label>
<input name="name" placeholder="Ihr Name . . .">
```

autocomplete

- | | |
|-----|---|
| on | Der Wert des Feldes darf gespeichert und wiederhergestellt werden. |
| off | Das Feld ist sicherheitskritisch die Eingabe darf nicht gespeichert werden. |

```
<label>Loginname</label>
<input name="username" autocomplete="off">
```

autofocus <input type="search" autofocus>

required

```
<label>Runner:</label>
<input name="name" required>
```

pattern

```
<input name="name" pattern="[0-9a-zA-Z]">
```

Neue Elemente

<progress></progress>

<output></output>

<meter></meter>

<keygen></keygen>

<progress>

```
<progress  
    min="0" max="100" value="50"  
></progress>
```

Das progress-Element gibt den Fortschritt einer bestimmten Aufgabe wieder. Das Element wird ebenfalls mit dem DOM gesteuert.

Anwendungsbeispiel wäre z. B. der Fortschritt bei einer mehrseitigen Umfrage.

<output>

Markiert Inhalte, die dynamisch berechnet werden,
beispielsweise die Summe eines Warenkorbes, oder
die Ausgabe eines Slider-Elementes.

<meter>

Messwerte (innerhalb eines definiertes Wertebereich).

Das meter-Element stellt einen Anteil seinem Maximum gegenüber. Verwenden kann man dieses Element beispielsweise um anzuzeigen, wie gut ein Suchergebnis im Vergleich zum Suchwort ist, wie gut ein Produkt im Vergleich zur besten Bewertung abschneidet etc.

```
<meter min="0" max="5"  
       value="4" step="1">  
</meter>
```

<keygen>

Generierter Key in einem Formular.