CSS3-Selektoren

e-Book

peter schmuhl

learn-it-now.de

Inhaltsverzeichnis

CSS3-Selektoren Startbild

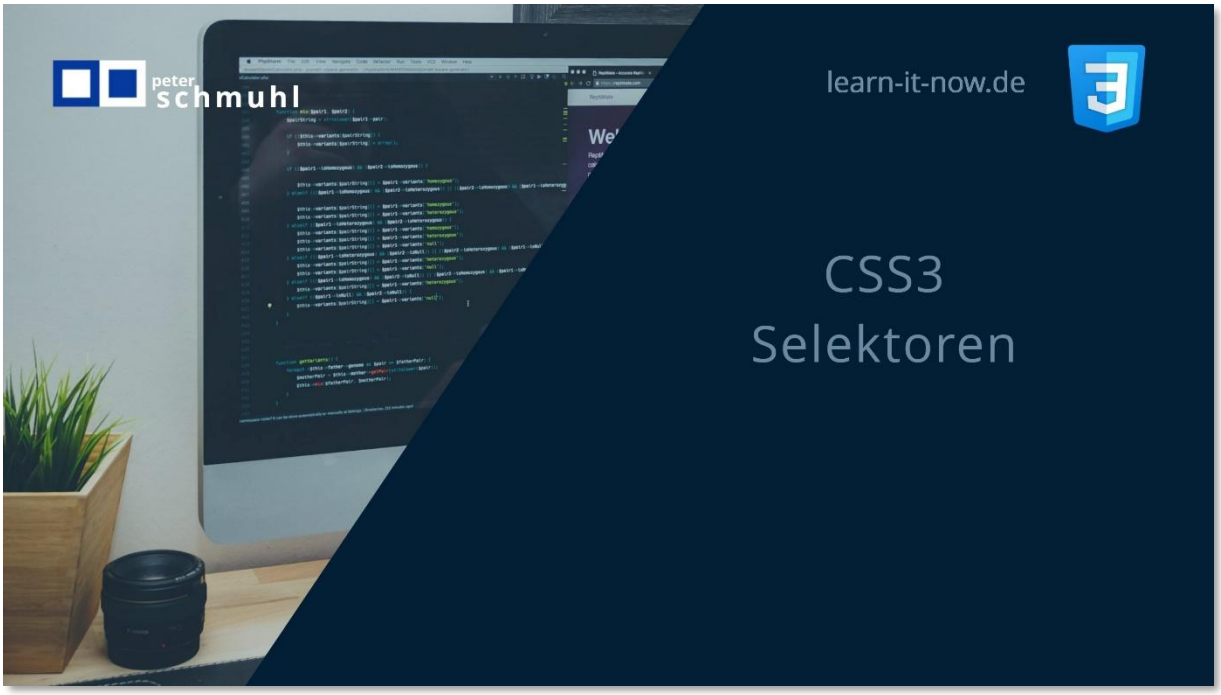


Bild 1 - HTML und CSS benötigen Selektoren

Dieses E-Book ist kein Tutorial für HTML und CSS. Es behandelt ausschließlich die Selektoren, insbesondere die Pseudoklassen und die Kombinatoren. Das Verständnis ist wichtig, um in Joomla das Template Helix Ultimate von JoomShaper anzupassen. Dieses Grundwissen sollte jeder haben, der sich mit der Gestaltung von Webseiten befasst.

CSS und JavaScript nutzen Selektoren. Es spielt dabei keine Rolle, ob du deine Webseite mit einem Content Management System (CMS) wie Joomla, Typo3, Drupal, WordPress oder Contao erstellst oder ohne CMS. Irgendwann stehst du vor dem Problem, dass dein erwünschtes Design-Element nicht mit den angebotenen Mitteln verwirklichen kannst. Dann musst du eingreifen und das vorhandene Design-Element selbst erstellen.

Die CMS haben schon für jedes HTML-Element ein ID und/oder mindestens eine Klasse vergeben, die du für dein Styling verwenden kannst. HTML und CSS solltest du halbwegs drauf haben, davon gehe ich aus. Jetzt musst du nur die Selektoren kennen und benutzen können. Am Ende dieses Tutorials wirst es können, versprochen?

CSS-Selektoren – Vorstellung der Selektoren

Bild 2 - Vorstellung der Selektoren

Selektoren in Custom-Code des Templates verwenden

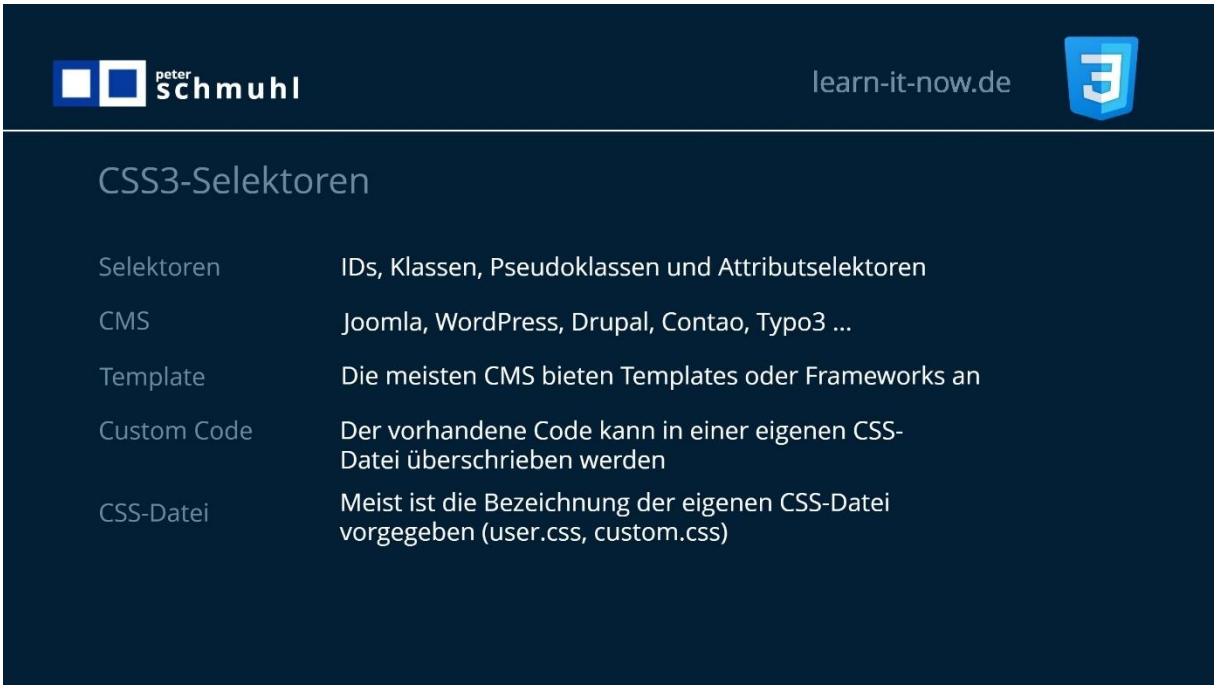


Bild 3 - Selektoren im CMS

Eigenen Code im CMS einbinden

Im HTML-Dokument wird der Inhalt und die Struktur der Webseite bestimmt. In der CSS-Datei wird die Webseite gestaltet. Content Management Systeme (CMS) verwenden Templates, die bei dem Erstellen der Webseite verwendet werden. Für ungeübte Anwender ist es schwierig, das Erscheinungsbild dieses Templates zu beeinflussen, selbst wenn gute HTML- und CSS-Kenntnisse vorhanden sind.

Beispiel Joomla

Im Joomla CMS ist standardmäßig das Cassiopeia-Template vorhanden. Dort muss die CSS-Datei user.css heißen. Das sehr flexible Template Helix Ultimate von JoomShaper verlangt den Namen custom.css für die eigene CSS-Datei.

Kommunikation zwischen HTML und CSS

Die HTML-Datei weiß natürlich nicht, wo er die CSS-Datei mit dem richtigen Namen findet. Der Pfad dahin muss im HEAD-Bereich der Webseite als Link oder als @Import der URL mitgeteilt werden.

In Joomla ist das einfacher. Wenn du die CSS-Datei mit dem richtigen Namen an der richtigen Stelle eingefügt hast, brauchst du dir bei Joomla keine Gedanken machen.

Die richtige Stelle finden

Im HTML-Code kannst jedem Tag eine ID oder mehrere Klassen vergeben, die der jeweilige Code der CSS-Datei direkt ansprechen kann. Jede ID ist einzigartig und kommt nur jeweils an einer Stelle im Code vor. Die Klasse kann an mehreren Stellen im HTML-Dokument verwendet werden und somit gleichzeitig angesprochen/selektiert werden.

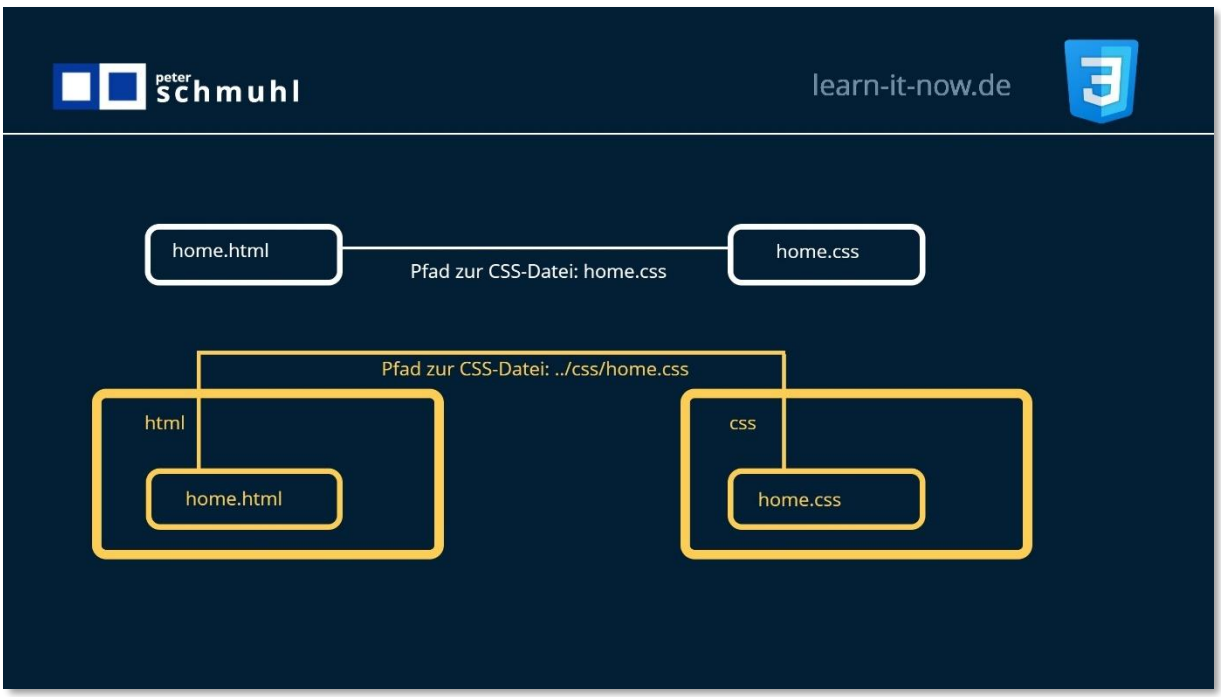


Bild 4 - Der Pfad von der HTML-Datei zur CSS-Datei

Beispiel in weiß

Die CSS-Datei home.css und die HTML-Datei home.html befinden sich auf gleicher Ebene. Der Link `<link rel="stylesheet" href="home.css">` verbindet HTML und CSS.

Beispiel in gelb

Die CSS-Datei home.css befindet sich im Ordner css und die HTML-Datei home.html befindet sich im Ordner html. Die Ordner html und css befinden sich auf gleicher Ebene. Die Wegbeschreibung von der home.html zur home.css ist vergleichbar mit der Bahnfahrt von home.html nach home.css. html und css sind Stadtteile von html oder css. Wir müssen 3-mal umsteigen. Zunächst vom Stadtteil home.html ins Stadtzentrum html (../) . Dort steigen wir in den Fernzug nach css(../css) und steigen dort in die Regionalbahn in den Stadtteil home.css (../ css/home.css.

Der Link `<link rel="stylesheet" href="../css/home.css">` verbindet HTML und CSS.

Was hätte ich damals gegeben, wenn mir das jemand erklärt hätte, wie man in den übergeordneten Ordner gelangt?! Ich musste mir diverse Webseiten anschauen, bevor ich das kapiert hatte. Nirgends war das dokumentiert.

Beispiel Joomla

In Joomla musst du nur wissen, wie die CSS-Datei heißen (user.css oder custom.css) muss. Dann wird die CSS-Datei automatisch gefunden.

ID und Klasse selektieren

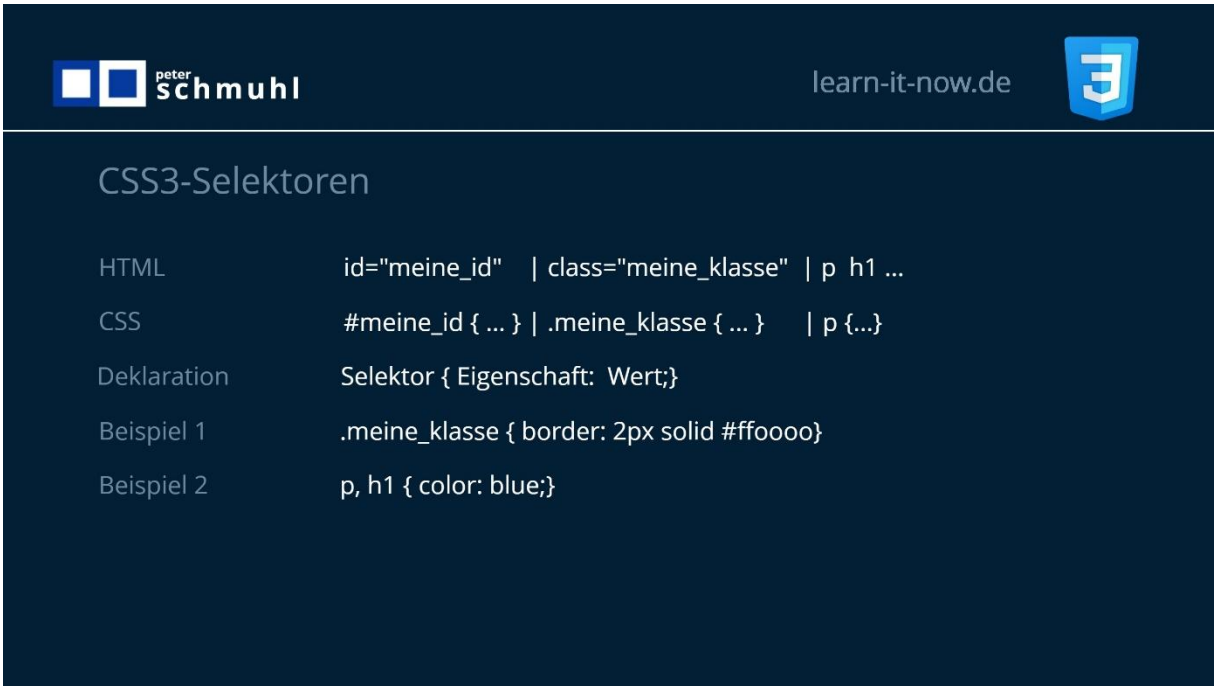


Bild 5 - Zusammenspiel HTML, CSS und Selektor(en)

Klasse und ID in HTML und CSS

Jedes HTML-Element kann eine Klasse oder ID verwenden. Die Schreibweise in HTML und CSS sind voneinander abweichend. Du kannst (<p id="irgendwas"><body class="etwas anderes"><h1 class="abc"><form id="formular1">) in jedem TAG eine ID oder mehrere Klassen vergeben. Klassen und IDs werden benötigt, um das Aussehen an einer einzigen Stelle zu gestalten. CSS verwendet ID und/oder, um genau diesen Bereich zu stylen.

Selektor{Eigenschaft: Wert;} →

```
#meine_id {color:  #ff0000;
    background-color: aqua;
        font-size: 30px;
    text-align: center;}
```

Eigenschaft können zum Beispiel die Schriftfarbe, die Schriftgröße, die Hintergrundfarbe oder die Ausrichtung sein. Der Wert könnte zum Beispiel rot, 30px, hellblauer Hintergrund oder zentriert sein.

Kommentare

Kommentare oder Code, der nicht angewendet werden soll, werden als Kommentar formatiert.

HTML-Kommentar:

```
<!--  Einzeiliger Kommentar -->
<!--
Mehrzeiliger Kommentar
-->
```

CSS-Kommentar:

```
/* Einzeiliger Kommentar */
/*
Mehrzeiliger Kommentar
*/
```

Kommentare sind nicht in allen Editoren (Eclipse, NetBeans, Visual Studio Code, Notepad++ ...) durch die grüne Schriftfarbe erkennbar. Jedes System hat sein eigenes Farbkonzept. Code, der innerhalb des Kommentars steht, wir als Kommentar angesehen und nicht verarbeitet.

Selektoren – Typ, Klasse, ID, Universalselektor

Bild 6 - Übersicht der einfachen Selektoren

Das Schaubild zeigt die Möglichkeit auf ein HTML-Element zuzugreifen. Entweder über jedes bereits im Code vorhandene oder selbst vergebene Klasse oder ID oder direkt über das HTML-Element (<p>, <h1>, < article> ...). Wenn alles ausgewählt werden soll, wird der Universalselektor (*)verwendet.

Spezifität

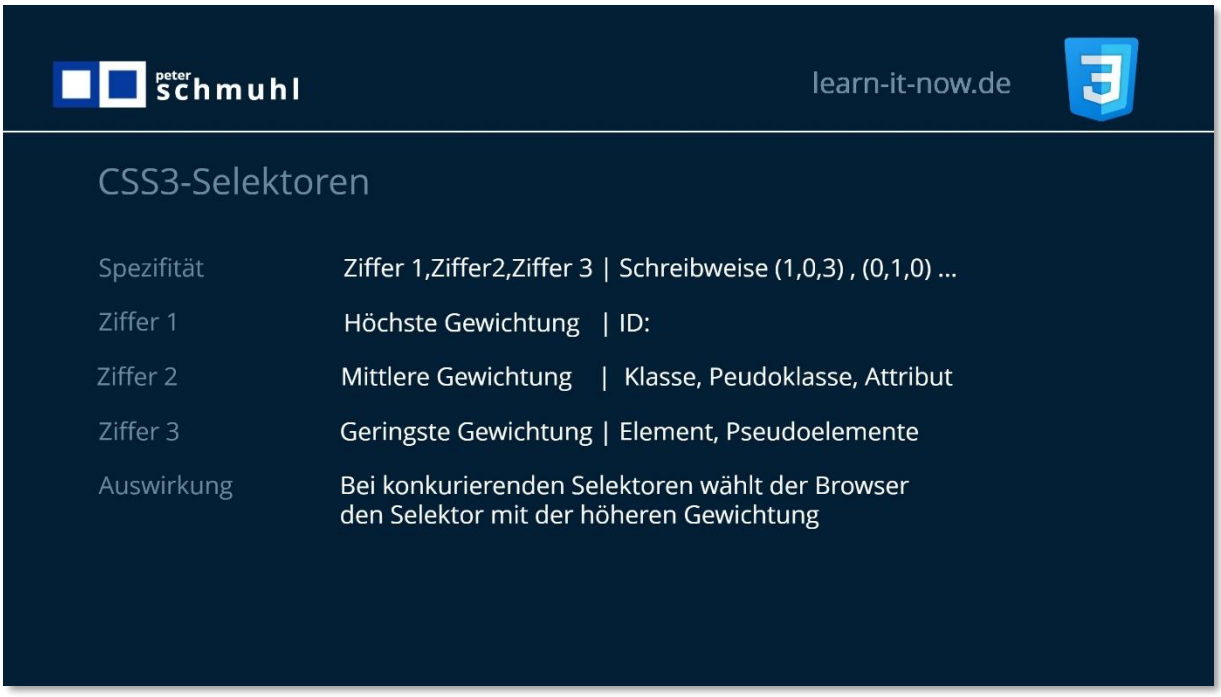


Bild 7 - Spezifität der Selektoren

Die Spezifität ist ein Algorithmus, den die Browser verwenden, um bei konkurrierenden Angaben, die in der Hierarchie höhere Anweisung auszuführen.

In meinem Handbuch HTML5 und CSS3 sind 4 Ebenen (A, B, C, D) beschrieben. Mein Browser verwendet nur 3 Ebenen (B, C, D). Im Schaubild habe ich die Schreibweise mit 3 Ebenen aufgeführt. Es können mehrere Selektoren in einer Hierarchie vorhanden sein. Kollidiert die CSS-Anweisung mit dem Selektor der Klasse mit den Anweisungen einer ID, dann gilt die Anweisung, die in der ID festgelegt ist. Verglichen mit einem Unternehmen will der Chef die Weihnachtsfeier im Ratskeller ausrichten (1,0,0) und der Abteilungsleiter (0,1,0) in der Pressebar oder der Sachbearbeiter(0,0,1) lieber ins Pulverfass gehen, wird klar, wie es funktioniert. Hier kollidieren sogar beide Spezifitäten mit der des Chefs.

Die vierte Ebene (A) ist dann in der Hierarchie die erste Ziffer(1,0,0,0). Sie wird nur verwendet, wenn für das HTML-Element Inline-Styling mit dem style-Attribut verwendet wird.

```
<p style="border: 2px solid red;">service</p>
```

Bild 7 - Inline-Style

Inline-Style sollte nicht verwendet werden. Inhalt (HTML) und Design (CSS) sollten immer getrennt erstellt werden.

CSS3-Selektoren Codebeispiele

Bild 8 - Codebeispiele HTML und CSS

001 HTML - Pseudoelement und Pseudoklasse

001-pseudoelemente.html > html

```
1   <!DOCTYPE html>
2   <html lang="de">
3   <head>
4       <meta charset="UTF-8">
5
6       <title>001 Pseudoklasse und Pseudoelement</title>
7       <link rel="stylesheet" href="001-pseudoelemente.css">
8   </head>
9   <body>
10      <p id="foo"></p>
11  </body>
12  </html>
```

Bild 9 - Pseudoklasse(n) und Pseudoelement(e)

001 CSS – Pseudoelement und Pseudoklasse

Die Pseudoelemente ::before und :: after werden verwendet, um vor oder nach einem Element etwas einzufügen oder zu ändern. Ein Beispiel ist auf meiner Webseite joomla-dummy.de zu finden. Es handelt sich um den Pfeil nach unten , der anzeigt, dass eine weitere (Unter-)Menüebene vorhanden ist.

Bild 10 - HTML- und CSS-Codebeispiele

*Bild 11 - Pseudoelement **::after***

```
# 001-pseudoelemente.css > ...
1   /*****************************************************
2   * Pseudoelemente (::)sollten durch die Schreibweise *
3   * von Pseudoklassen (:) unterscheiden!              *
4   *****************************************************/
5   #foo::before {
6       color: red;
7
8   }
9
10  #bar:hover {
11      color: blue;
12  }
```

Bild 12 - Pseudoklasse und Pseudoelement

002 HTML – Kombinatoren

Kombinatoren zwischen 2 Selektoren bestimmen, ob der zweite Selektor ein Nachfahre, Kind oder Geschwister vom ersten Selektor sein soll, das ausgewählt wird.

002-kombinatoren.html > html > body > h2

```
1   <!DOCTYPE html>
2   <html lang="de">
3   <head>
4       <meta charset="UTF-8">
5
6       <title>002 Kombinatoren</title>
7       <link rel="stylesheet" href="002-kombinatoren.css">
8   </head>
9   <body>
10      <p>Hallo</p>
11      <p>Hallo</p>
12      <h1>Hallo <span>Welt </span>hier bin ich!</h1>
13      <p>Hallo</p>
14      <p>Hallo</p>
15      <h2>ich bin vom Mars</h2>
16      <p>Hallo</p>
17      <h3>Wo bist du?</h3>
18      <p>Hallo</p>
19      <div>
20      <p>Hallo</p>
21      <p>Hallo</p>
22      </div>
23      <p>Hallo Welt</p>
24      <p>Hallo du schöne Welt</p>
25      <p>Hallo meine liebe Welt</p>
26  </body>
27  </html>
```

Bild 13 - Kombinatoren

002 CSS – Kombinatoren

*002 – Kombinator **Leerzeichen***

```
# 002-kombinatoren.css > ...
1   /*******************************************************
2   * Kombinatoren verbinden 2 CSS-Selektoren.            *
3   * Kombinatoren: Leerzeichen, >, ~, +, $               *
4   * *****************************************************/
5
6   /*******************************************************
7   * Beispiel 001: #foo #bar --> NACHFAHRENSELEKTOR      *
8   *                                                     *
9   *Selektiert werden soll #bar, das ein Nachfahre       *
10  * von #foo ist.                                       *
11  *******************************************************/
12
13  h1 span {
14      background-color: antiquewhite;
15      color: blueviolet;
16  }
17
```

*Bild 14 - Kombinator **Leerzeichen***

*002 – Kombinator **>***

```
/*******************************************************
* Beispiel 002: #foo > #bar --> KINDSELEKTOR           *
* bar ist ein Element                                  *
* Selektiert werden soll #bar, das ein direkter        *
* Nachfahre von #foo ist.                              *
*******************************************************/

/*Beispiel 002*/

div > p {
    background-color: rgb(252, 232, 8);
    color: rgb(124, 9, 246);
    text-align: center;
}
```

*Bild 15 Kombinator **>***

002 – Kombinator ~

```
/*************************************************
* Beispiel 003: #foo ~ #bar --> GESCHWISTERSELEKTOR  *
* Selektiert werden soll #bar, das ein Geschwister-  *
* Element von #foo ist und ihm folgt.                *
* (im selben Elternelement)                          *
*************************************************/

/*Beispiel 003*/

h2 ~ p {
    background-color: aquamarine;
    color: red;
}
```

Bild 16 - Kombinator ~

002 – Kombinator +

```
/*****************************************************
* Beispiel 004: #foo + #bar --> NACHBARSELEKTOR       *
* Selektiert werden soll #bar, das das direkte       *
* Nachbarelement von #foo ist und ihm folgt.         *
* (im selben Elternelement)                          *
*****************************************************/

/*Beispiel 004*/

h3 + p {
    background-color: rgba(255, 127, 242, 0.697);
    color: rgb(89, 255, 0);
}
```

Bild 17 - Kombinator +

003 HTML – Attributselektoren

Der Attributselektor wählt eine sucht in einer Zeichenkette, ob sie mit dem Attribut, beginnt (Hallo), ob sie mit dem Attribut endet (Welt) oder ob die Zeichenkette die Zeichen enthält (arme).

003-attributselektoren.html > ...

```html
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">

    <title>003 Attributselektoren</title>
    <link rel="stylesheet" href="003-attributselektoren.css">
</head>
<body>
    <p title="Hallo">Hallo</p>
    <p title="Hallo Welt">Hallo Welt</p>
    <p title="Hallo meine Welt">Hallo meine Welt</p>
    <p title="Hallo du meine Welt">Hallo meine Welt</p>
    <p title="Hallo du meine schöne Welt">Hallo meine schöne Welt</p>
    <p title="Hallo du meine schöne arme Welt">Hallo meine schöne arme Welt</p>
    <p title="Hallo du meine schöne arme Welt bist einzigartig">Hallo du meine schöne Welt bist einzigartig</p>
</body>
</html>

```

Bild 18 – Attributselektoren (HTML)

003 CSS - Attributselektoren

003-attributselektoren.css > ...

```css
[title^="Hallo"] {
    background-color: chartreuse;
    color: brown;
}

[title$="Welt"] {
    background-color: rgb(72, 0, 255);
    color: white;
}

/*********************************************************
* Die Auswahl richtet sich nach dem title-attribut "arme"  *
* Es spiel keine Rolle, welchen Text der Titel hat !!!     *
*********************************************************/
[title*="arme"] {
    background-color: #fffb00;
    color: rgb(60, 8, 248);
    font-weight: bold;
}

```

Bild 19 Attributselektoren (CSS)

004 HTML – (an+b)nth-child

004- nth_pseudoklassen.html > html > head > link

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">

    <title>004 (an+x) - NTH-Pseudoklassen</title>
    <link rel="stylesheet" href="004-nth_pseudoklassen.css">
</head>
<body>
<h1>Pseudoklassen nth-child(an+b)</h1>
<ul id="navi">
    <li>Home</li>
    <li>Produkte</li>
    <li>Dienstleistungen</li>
    <li>Service</li>
    <li>Preise</li>
    <li>Kontakt</li>
    <li>Impressum</li>
    <li>Datenschutzerklärung</li>
    <li>Cookie-Consent</li>
</ul>
</body>
</html>
```

Bild 20 - nth-child(an+b)

004 CSS – nth-Child(x) | nth-Child(an+x)

004 CSS – nth-Child(x)

Im der ersten CSS-Abbildung werden die Kinder 2, 4, und 6 ausgewählt und gestylt.

```css
# 004-nth_pseudoklassen.css > li:nth-child(4)
h1 {
    font-size: 250%;
    text-align: center;
}

/*****************************************************************************************
* :nth-child(x) - Alle, die sich in einem Elternelement befinden - von oben nach unten      *
* :nth-last-child(x) - Alle, die sich in einem Elternelement befinden - von unten nach oben *
* :nth-of-type(x) - Alle des selben Typs - von oben nach unten                              *
* :nth-last-of-type(x) - Alle des selben Typs von unten nach oben                           *
*  x --> Schlüsselwörter, Zahlen oder Positionsbestimmung durch (an+b)                      *
* a und b können negativ sein (-3n+2), (-3n-2), (3n-2)                                      *
* :last-child, only-child, :only-of-type,  :first-of-type, empty, :not und weitere          *
*****************************************************************************************/

/*Das Kindelement 2, 4 und 6*/
li:nth-child(2){
    font: sans-serif;
    font-weight: bold;
    color: blueviolet;
    background-color: antiquewhite;

}
li:nth-child(4) {
    background-color: crimson;
    color: azure;

}
li:nth-child(6){
    border: 3px solid green;
}
```

Bild 21 - nth-child(x)

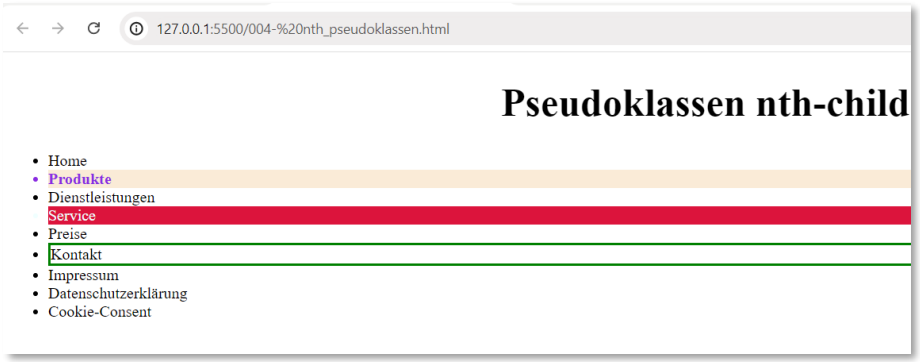


Bild 22 - Ergebnis der Auswahl

004 CSS – nth-Child(an+x)

In der zweiten CSS- Abbildung werden nacheinander jedes 2. Element ab dem 3. Element ausgewählt (2n+3), jedes 4. Element an dem 3. Element und jedes 2. Element ab dem 2. Element.

004-nth_pseudoklassen.css > li:nth-child(3n)

```
34
35  /* an+b --> Jedes a-te Element ab dem b-ten Element */
36
37  li:nth-child(2n+3){
38
39      font-weight: bold;
40      font-size: 20px;
41      color: rgb(246, 2, 2);
42      background-color: rgba(215, 243, 250, 0.452);
43  }
44  li:nth-child(4n+3) {
45      border: 3px solid rgb(4, 245, 4);
46      background-color:
47      rgba(233, 243, 182, 0.333);
48      color: rgb(96, 194, 194);
49  }
50  li:nth-child(2n+2){
51      text-decoration: line-through;
52  }
53
54  li:nth-child(3n){
55      text-decoration: line-through;
56      text-transform: uppercase;
57  }
```

Bild 23 - nth-child(an+b)

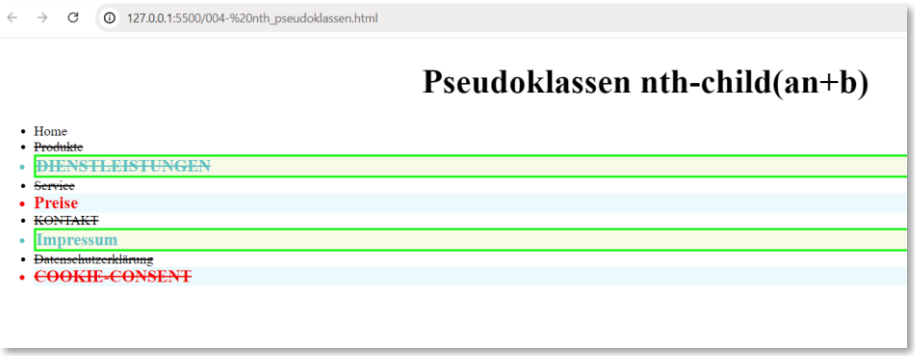


Bild 24 - Ergebnis der Auswahl

005 HTML- Even und Odd

005-pseudoklassen_even_odd.html | 005-pseudoklassen_even_odd.css | 004-nth_pseudoklassen.css

005-pseudoklassen_even_odd.html > html > head > title

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">

    <title>005 - Even und Odd</title>
    <link rel="stylesheet" href="005-pseudoklassen_even_odd.css">
</head>
<body>
    <h1>Pseudoklassen Even und Odd</h1>
    <ul id="navi">
        <li>Home</li>
        <li>Produkte</li>
        <li>Dienstleistungen</li>
        <li>Service</li>
        <li>Preise</li>
        <li>Kontakt</li>
        <li>Impressum</li>
        <li>Datenschutzerklärung</li>
        <li>Cookie-Consent</li>
    </ul>
</body>
</html>
```

Bild 25 - even und odd

005 CSS- Even und Odd

005-pseudoklassen_even_odd.html | 005-pseudoklassen_even_odd.css

005-pseudoklassen_even_odd.css > ...

```
li:nth-child(odd){
    background-color: aquamarine;
}


li:nth-child(even){
    background-color: rgb(245, 181, 7);
}
```

Bild 26 - even und odd

Bild 27 - Ergebnis der Auswahl

Even und Odd wählen jedes gerade oder/und jedes ungerade Element aus.

006 Unterschied zwischen nth-child und nth-of-type

006-pseudoklassen-2.html > html > head > title

```
1   <!DOCTYPE html>
2   <html lang="de">
3   <head>
4       <meta charset="UTF-8">
5       <link rel="stylesheet" href="006-pseudoklassen-2.css">
6       <title>006-Unterschied zwischen nth-child und nth-of-type</title>
7   </head>
8   <body>
9
10      <p>Hallo</p>
11      <p>Welt</p>
12      <p>Hier bin ich</p>
13      <ol>
14          <li>Home</li>
15          <li>Produkte</li>
16          <li>Dienstlestungen</li>
17          <li>Service</li>
18          <li>Preislisten</li>
19          <li>Impressum</li>
20          <li>Datenschutzerklärung</li>
21          <li>Cookie-Consent</li>
22      </ol>
23      <ol>
24          <li>Einzelkind</li>
25      </ol>
26      <p>Ich wünsche dir weniger CO2,</p>
27      <p>weniger Plastik in den Meeren,</p>
28      <p>Freiheit für alle!</p>
29  </body>
30  </html>
```

Bild 28 - Vergleich nth-child und nth- of-type

006 CSS – Vergleich zwischen nth-child und nth-of-type

```
# 006-pseudoklassen-2.css > ...
1   /* p:nth-child(an+b)
2   (2n+2)Jedes 2. Element, beginnend ab dem 2. Element
3   wird nicht eingefärbt, weil sie kein p-Element ist.
4   Die Liste (OL) wird als Element mitgezählt.
5    */
6   p:nth-child(2n+2){
7       color: #ff0000;
8       background-color: azure;
9   }
10  /*
11  (3n+2)Jedes 3. Element, beginnend ab dem 2. Element
12  erhält einen grünen Rahmen, weil sie kein p-Element ist.
13  Die Liste (OL) wird als Element mitgezählt.
14   */
15  p:nth-child(3n+2){
16      border: 4px solid green;
17  }
18  /*******************************************************************************/
19  /* p:nth-of-type(an+b)
20  (2n+2)Jedes 2. Element, beginnend ab dem 2. Element
21  wird nicht eingefärbt, weil sie kein p-Element ist.
22  Die Liste (OL) wird als Element NICHT mitgezählt.
23   */
24  p:nth-of-type(2n+1){
25      font-size: 150%;
26      text-decoration: overline;
27      border: 2px solid burlywood;
28      background-color: aquamarine;
29
30  }
```

Bild 29 - Vergleich nth child und nth-of-type

006 Ergebnis – Vergleich zwischen nth-child und nth-of-type

Hallo

Welt

Hier bin ich

1. Home
2. Produkte
3. Dienstlestungen
4. Service
5. Preislisten
6. Impressum
7. Datenschutzerklärung
8. Cookie-Consent

1. Einzelkind

Ich wünsche dir weniger CO2,

weniger Plastik in den Meeren,

Freiheit für alle!

Bild 30 – Ergebnis der Auswahl – nth-child

Die rote Schrift und der hellblaue Hintergrund werden von “(p:nth-child(2n+2))“ und der grüne Rahmen aus “(p:nth-child(3n+2))“.

“(p:nth-child(2n+2))“ → Welt ist das 2 Kindelement und erhält den hellblauen Hintergrund und die rote Schriftfarbe. Jedes weitere 2. Kind wird wie Welt gestylt, wenn es ein P-Element ist. **Die Liste OL wir als Kind mitgezählt.** Die Liste ist ein OL-Element und würde den Syle erhalten, wenn es ein P-Element wäre. Danach ist “weniger CO_2“ wieder das 2. Element und “ Freiheit“ ebenso wieder das 2. Element. Da beide Elemente P-Elemente sind, erhalten sie den Style.

“(p:nth-child(3n+2))“ → hier wird Welt und dann jedes 3. Element ab Welt mit einem grünen Rahmen versehen, wenn es sich um ein P-Element handelt. “Einzelkind“ ist das 3. Element, jedoch ein OL-Element. Daher erhält es keinen grünen Rahmen. “Freiheit ist dann wieder das 3. Element“. Da es ein P-Element ist, erhält die Freiheit einen grünen Rahmen.

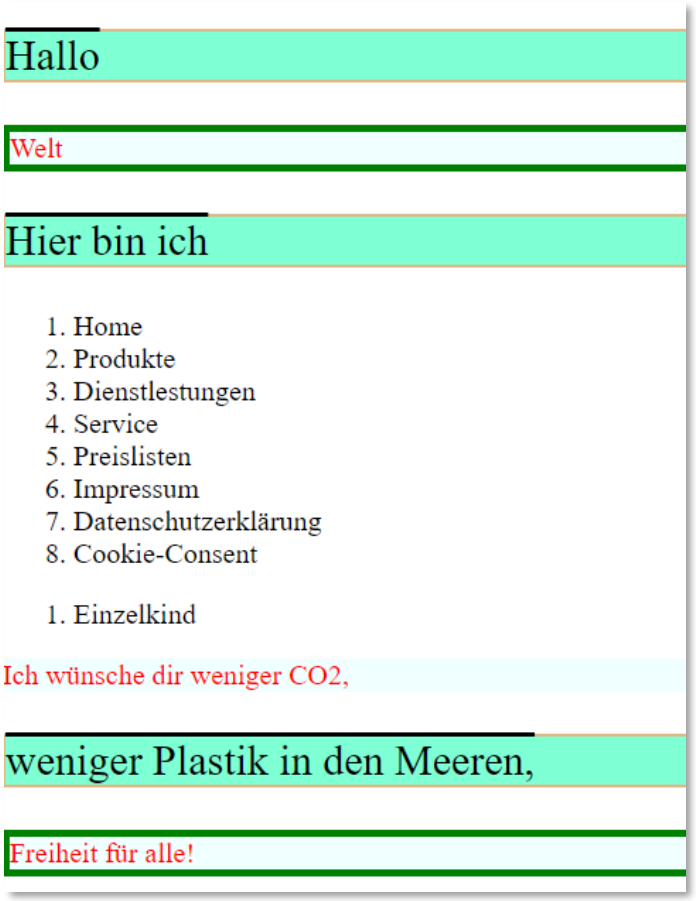


Bild 31 - Ergebnis der Auswahl – nth-of-type

Bei “(p:nth-of type(3n+2))“ werden nur die P-Elemente gezählt.

Jedes 2. P-Element ab dem 1. P-Element erhält einen braunen Rahmen, einen leicht grünen Hintergrund und einen Strich über dem Text. Vorher gemachte CSS-Styles werden überschrieben.
“Hier bin ich“ ist das dann 2. P-Element und “weniger Plastik“ wiederum das jeweils 2. P-Element. **Die OL-Elemente werden nicht mitgezählt.**

007 HTML - Child-Selektoren

```html
<body>
    <p id="meine_id">Hallo</p>
    <ol id="navi">
        <li>Home</li>
        <li>Produkte</li>
        <li>Dienstleistungen</li>
        <li>Service</li>
        <li>Preise</li>
        <li>Kontakt</li>
        <li>Impressum</li>
        <li>Datenschutzerklärung</li>
        <li>Cookie-Consent</li>
    </ol>
    <p>Welt</p>
    <!--1. Empty P-Element--><p></p>
    <!--2. Empty P-Element mit einem Leerzeichen--><p> </p>
    <!--3. Empty P-Element ohne Inhalt mit Zeilenumbruch in Quelltext--><p>
    </p>
    <p><input type="text" value="mein_wert"></p>
    <!-- Artikel 1 mit einem Einzelkind--><article>
        <p>Einzelkind</p>
    </article>
    <!--Artikel 2 mit 2 Kindern--><article>
        <p>Erstes Kind</p>
        <p>Zweites Kind</p>
    </article>
    <!--3 Eingabefelder 1 und 2 enabled, 3 disabled.--><h3>
        <hr>
        <input type="text" value="mein_wert">
        <hr>
            <input type="text" value="mein_wert_2" enabled>
        <hr>
            <input type="text" value="mein_wert_3" disabled>
        <hr>
    </h3>
</body>
```

Bild 32 - Selektoren first-child und last child (HTML)

007 CSS – Child-Selektoren

007 CSS – first-child und last-child

007-children-selektoren.css > ...

```css
/*
Das erste Kind-Element und das letzte
Kind-Element von li sollen selektiert
und gestylt werden.
*/
/**/
li:first-child{
    color: greenyellow;
    background-color: rgb(245, 152, 3);
    font-weight: bold;
}
/**/

li:last-child {
    color: rgb(158, 47, 255);
    background-color: rgb(233, 245, 3);
    font-weight: bold;
}
```

Bild 33 - Selektoren first-child und last-child (CSS)

007 Ergebnis first-child und last-child

Ich habe unsere Seite etwas umgebaut. Der erste und der letzte Menüpunkt unserer Liste sollen selektiert werden und erhalten jeweils eine Schriftfarbe, eine Hintergrundfarbe und die Schrift soll fett angezeigt werden. First-Child der Liste ist Home mit orangem Hintergrund und Last-Child ist Cookie-Consent mit dem gelben Hintergrund.

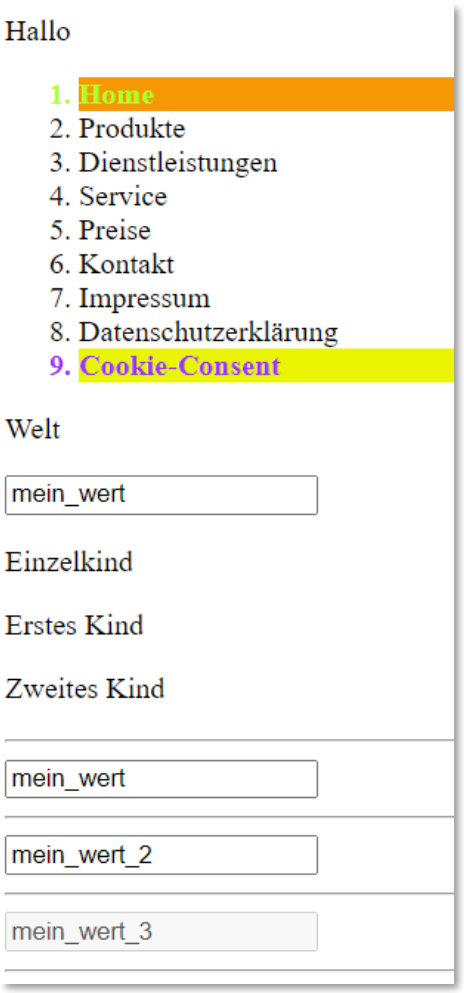


Bild 34 Ergebnis first-child und last-child

007 CSS – only-child und empty

Wenn in dem jeweiligen Element nur Kind-Element (only-child) vorhanden ist, können wir dieses Element mit only-child stylen.

Ein leeres Element ist nicht immer ein leeres Element! Es kommt auf die Feinheiten an, wie du gleich sehen wirst.

```css
/*
Alle Einzelkinder erhalten einen orangen Rahmen.
body ist Einzelkind von html,
input ist Einzelkind von p,
p ist Einzelkind von article.
In zweiten Artikel sind 2 Kind-Elemente p
*/

:only-child {
    border: 3px solid orange;
}

/*
emty ist nur erfüllt, wenn das p-Element keinen Inhalt hat.
Ein Zeilenumbruch im Code und/oder ein Leerzeichen werden
als Inhalt angesehen.
*/

p:empty {
    border: 3px seagreen solid;
}
```

Bild 35 Selektoren only-child und empty

007 Ergebnis only-child und empty

Bild 36 - Ergebnis only-child und empty

Bild 37 Ergebnis empty und first-child

007 Only-Child

Im Falle only-child werden nur alle Elemente ausgewählt, die ein Einzelkind sind. Body ist Einzelkind von HTML und erhält den orangen Rahmen um das gesamte Dokument.

Das Input-Element mit dem Value “mein_wert“ ist Einzelkind und Das P-Element mit dem Text “Einzelkind“ ist in dem Artikel-Element das einzige Kind. Beide Elemente erhalten ebenfalls einen orangen Rahmen.

007 Empty

Im HTML-Dokument sind 3 leere P-Elemente vorhanden. Element 1 ist ganz leer, Element 2 hat ein Leerzeichen als Inhalt und Element 3 einen Zeilenumbruch. Alle 3 P-Elemente werden selektiert und erhalten eine grüne Linie. Nur das erste P-Element wird angezeigt, da Leerzeichen und sogar der Zeilenumbruch als Inhalt gewertet werden.

007 CSS - root, enabled, disabled und not

```
40
41  /*
42  root wäht das Wurzel-Element eines Dokumentes auf.
43  Das ist in HTML-Dokumenten immer das HTML-Element
44  */
45  :root {
46      border: 10px solid tomato;
47  }
48  /*
49  enabled und disabled sollen selektiert werden.
50  Die Inputfelder mit Wert 1 (enabled) und Wert 2 (disabled)
51  sollen eine Hintergrundfarbe erhalten. Dabei wird das
52  Eingabefeld mit Wert ebenfalls als enabled erkannt.
53  */
54  :enabled {
55      background-color: aqua;
56  }
57  */
58  :disabled {
59      background-color: blue;
60      color: white;
61  }
62  /*
63  not schließt das entsprechende Element aus.
64  Das erste p-Element mit der ID meine_id
65  und dem Inhalt Hallo werden von der
66  CSS-Anweisung ausgeschlossen
67  */
68  p:not(#meine_id) {
69      font-size: 24px;
70      color: rgb(234, 11, 11);
71  }
```

Bild 38 - Selektoren root, enabled, disabled und not

007 Ergebnis root, enabled, disabled und not

007 root

Mit root wird das gesamte HTML-Dokument angesprochen und erhält in unserem Fall einen roten Rahmen.

007 enabled und disabled

Es sind 3 Eingabefelder in einer Gruppe vorhanden. Eingabefeld 1“mein_wert“, Eingabefeld 2“mein_wert_2“(enabled), Eingabefeld 3“mein_wert_3“(disabled).

Eingabefeld 1 und 2 sind aktiviert. Dies gilt ausdrücklich automatisch auch für das Eingabefeld 1, da Eingabefelder automatisch aktiviert sind, wenn sie nicht ausdrücklich deaktiviert sind. Daher erhalten sie einen hellblauen Hintergrund. Das gilt auch für unser als Einzelkind vorhandenes Eingabefeld weiter oben, außerhalb der Dreiergruppe.

Nur das Eingabefeld hat die Eigenschaft disabled und wird blau eingefärbt und erhält eine weiße Schrift.

007 not

Alle P-Elemente, die nicht die ID “meine_id“ haben, werden in roter Schrift in der Größe 24 Pixel dargestellt.

Bild 39 - Ergebnis root, enabled, disabled und not

Bild 40 - Verschachtelte DIV-Ebenen

008 HTML – Verschachtelte Elemente ansprechen

010-verkettung.html > html

```html
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>010-Verschachtelt 1 - Vererbung</title>
    <link rel="stylesheet" href="010-verkettung.css">
</head>
<body>
    <div class="ebene_0">
        <div class="ebene_1">
            <div class="ebene_2">
                <div class="ebene_3">
                </div>
            </div>
        </div>
        <div class="ebene_1">
            <div class="ebene_2">
                <div class="ebene_3">
                </div>
            </div>
        </div>
        <div class="ebene_1">
            <div class="ebene_2">
                <div class="ebene_3">
                </div>
            </div>
        </div>
    </div>

</body>
</html>
```

Bild 41 – HTML der DIVs in 4 Ebenen verschachtelt

Wie wird das DIV-Element einer Untergeordneten Ebene selektiert? Das werden wir uns in diesem Kapitel ansehen. Zum besseren Verständnis habe ich auf Text-Inhalte verzichtet und baue die Grafiken mit euch gemeinsam auf.

008 CSS – Verschachtelte Elemente ansprechen

```css
*{
    margin: 3px;
    padding: 20px;
}

.ebene_0 {
    border: 10px solid rgb(117, 93, 227);
    padding: 0px;
}
.ebene_1{
    background-color: yellow;
    color: blue;
    border: 3px solid red;
}
.ebene_2 {
    background-color: rgb(0, 255, 64);
    color: rgb(247, 7, 7);
    border: 3px solid blue;
}
.ebene_3 {
    background-color: rgb(191, 228, 242);
    color: rgb(139, 7, 247);
    border: 3px solid yellow;
}
```

Bild 42 - CSS- Style der DIV-Ebenen

008 Verschachtelte Elemente ansprechen

Zunächst stylen wir unsere DIVs mit unterschiedlichen Rahmen und Hintergrundfarben.

008 Ebene 0

Unser DIV der Ebene 0 soll ein Rahmen werden. Da keine Inhalte oder weitere Eigenschaften vergeben wurden liegen der obere Rahmen und der untere Rahmen direkt untereinander.

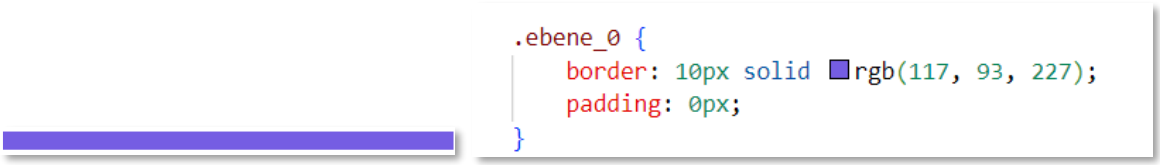
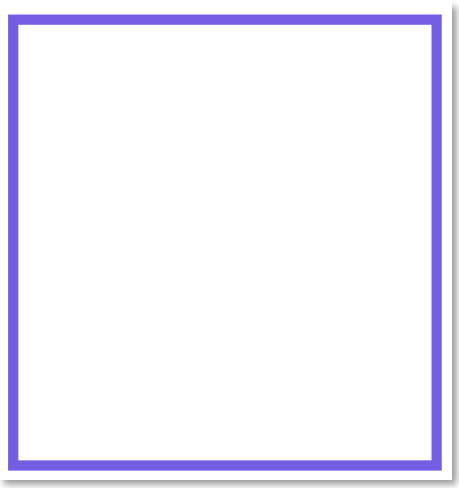


Bild 43 - Ebene 0 ohne padding und margin

008 Ebene 0 mit Padding und Margin

Wir fügen Padding und Margin ein, um ein Rechteck (Quadrat) zu erhalten.

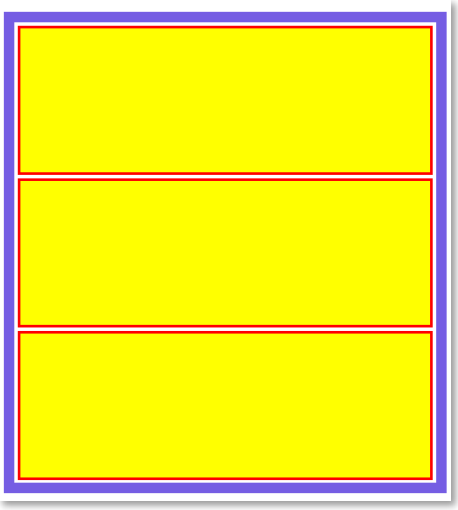


```
*{
    margin: 3px;
    padding: 20px;
}
```

Bild 44 - Ergebnis nach Einfügen von CSS margin und padding

008 Ebene 0, und Ebene 1

Wir fügen unsere Ebene 1 als Unterebene von Ebene 0 hinzu. Die Ebene 1 ist dreimal vorhanden. Die Hintergrundfarbe ist gelb und hat einen roten Rand.
Die Textfarbe blau wird hier momentan nicht benötigt.

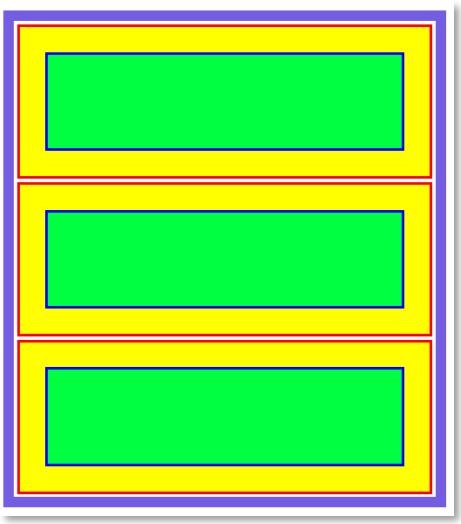


```
.ebene_1{
    background-color: yellow;
    color: blue;
    border: 3px solid red;
}
```

Bild 45 - Ebene 1 mit 3 weitere DIVs hinzugefügt

008 Ebene 0, Ebene 1 und Ebene 2

Wir wiederholen den Schritt mit der Ebene 2, die jeweils als Unterebene der Ebene 1 eingefügt wird. Auch hier erhalten Hintergrundfarbe, Rahmen und Text abweichende Farben zugewiesen.

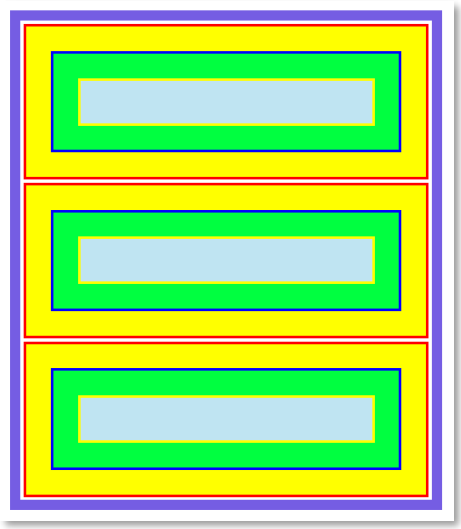


```
.ebene_2 {
    background-color: rgb(0, 255, 64);
    color: rgb(247, 7, 7);
    border: 3px solid blue;
}
```

Bild 46 - Ebene 2 in alle DIVs der Ebene 1 hinzugefügt

008 Ebene 0, Ebene 1, Ebene 2, und Ebene 3

Zuletzt fügen wir die Ebene 3 als Unterebene der Ebene 2 hinzu und schließen die Vorarbeiten ab. Wie in den oberen Schritten erhalten Text Rahmen und Hintergrund Farben zugewiesen.



```
.ebene_3 {
    background-color: rgb(191, 228, 242);
    color: rgb(139, 7, 247);
    border: 3px solid yellow;
}
```

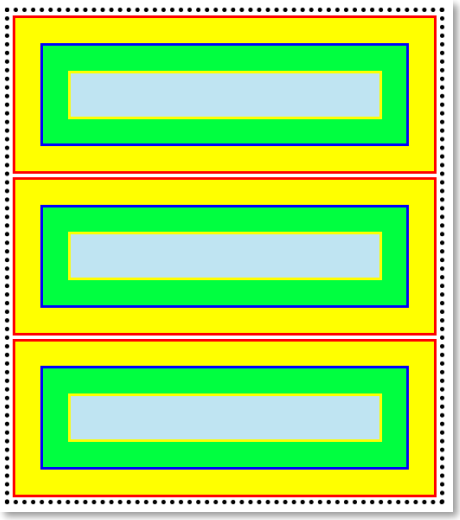
Bild 47 - Ebene 3 in alle DIVs der Ebene 2 hinzugefügt

008 Selektoren in allen Ebenen verwenden

Im nächsten Schritt werden wir die Ebenen nacheinander ansprechen und sehen, wie die Selektoren verwendet werden-

008 Border Ebene 0 wird überschrieben

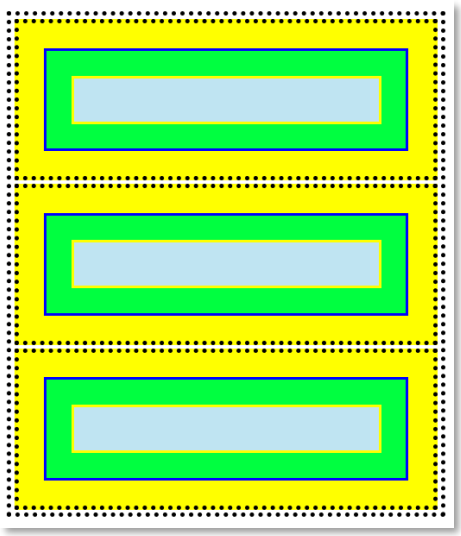
Die Ebene 0 erhält jetzt einen gepunkteten Rand.



```
.ebene_0{
    border: 4px dotted black;
}
```

008 Border Ebene 1 wird ebenfalls überschrieben

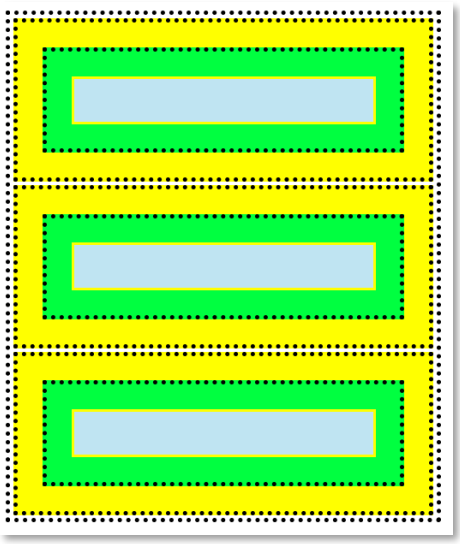
Der korrekte Selektor für die Ebene 1 ist ".ebene_0 .ebene_1".



```
.ebene_0{
    border: 4px dotted black;
}
.ebene_0 .ebene_1{
    border: 4px dotted black;
}
```

008 Border Ebene 2 wird ebenfalls überschrieben

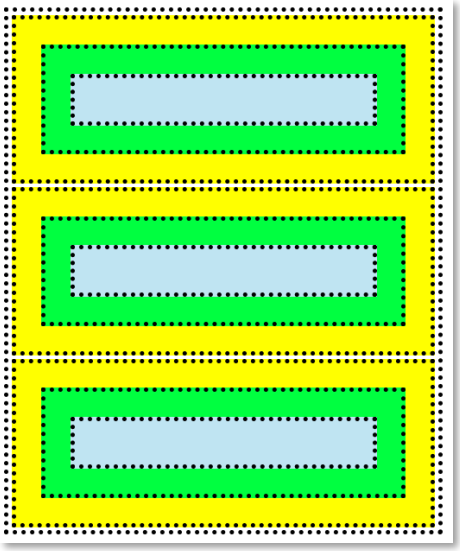
Der korrekte Selektor für die Ebene 1 ist ".ebene_0 .ebene_1 .ebene_02".



```
.ebene_0{
    border: 4px dotted black;
}
.ebene_0 .ebene_1{
    border: 4px dotted black;
}
.ebene_0 .ebene_1 .ebene_2{
    border: 4px dotted black;
}
```

008 Border Ebene 3 wird ebenfalls überschrieben

Der korrekte Selektor für die Ebene 1 ist ".ebene_0 .ebene_1 .ebene_02 .ebene_3".



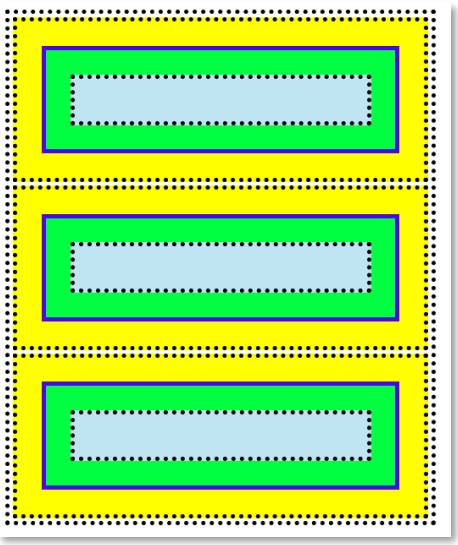
```
.ebene_0{
    border: 4px dotted black;
}
.ebene_0 .ebene_1{
    border: 4px dotted black;
}
.ebene_0 .ebene_1 .ebene_2{
    border: 4px dotted black;
}

.ebene_0 .ebene_1 .ebene_2 .ebene_3 {
    border: 4px dotted black;
}
```

Das hat bisher prima funktioniert, da wir immer jeweils den korrekten Selektor angesprochen haben. Ebene 3 ist Nachfahre von Ebene 2, Ebene 2 ist Nachfahre von Ebene 1 und Ebene 1 ist Nachfahre von Ebene 0

008 Korrekter Selektor und falscher Selektor im Vergleich

Ebene 2 soll überschrieben werden. Die erste CSS-Anweisung ist aufgrund des unscharfen Selektors unwirksam und wird nicht ausgeführt. Die zweite CSS-Anweisung bildet den Pfad zur Ebene 2 korrekt ab und führt zum gewünschten Ergebnis. Ebene 2 erhält jetzt einen blauen Rand.

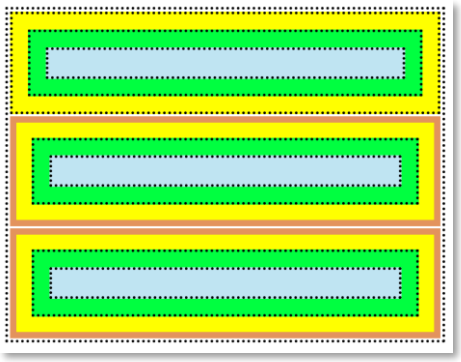


```css
.ebene_2 {
    border: 10px solid rgb(227, 147, 93);
}

.ebene_0 .ebene_1 .ebene_2{
    border: 4px solid rgb(69, 8, 238);
}
```

008 Border wird bei den Geschwistern überschieben.

In unserem letzten Beispiel sollen die Geschwister-Elemente der Ebene 1 überschrieben werden. Alle 3 Ebenen 1 haben vor dem Überschreiben einen gepunkteten Rand und einen gelben Hintergrund. Der Selektor ".ebene_1 ~ .ebene_1" selektiert die beiden Geschwister **von** der obersten **Ebene 1** und führt die CSS-Anweisung aus, den Rand der/aller Geschwister (**von**)der obersten Ebene braun darzustellen.



```css
.ebene_1 ~ .ebene_1{
    border: 10px solid rgb(227, 147, 93);
```