CSS – Cascading Style Sheets

History, Concept, Examples

Oliver Kerschbaumsteiner

Überblick

1. History and Development

- Web Design before CSS
- History
- Development Process
- CSS Levels

2. Operating Principle

- Concept
- Box-Model
- Syntax and Selectors
- Ways to Insert CSS

3. Example of Use

- Frameworks
- Nutshell Examples

4. Summary

1. History and Development - Before CSS

- SGML, FOSI
- DSSSL – Document Style Semntics and Specification Language
  - Unübersichtliche Syntax
  - Zu Umfangreich
- Kein standardisiertes Konzept für das Stylen von Webseiten
- NCSA Moasaic Browser
- Håkon Wium Lie - Cascading HTML Style Sheets

1. History and Development - History

- Cascading HTML Style Sheets → Cascading Style Sheets
- November 1994: Präsentation des ersten Entwurfs
- Alleinstellungsmerkmal:
  Autor, User und technische Möglichkeiten können das Design beeinflussen
- Ab 1995: steigender Einfluss des World Wide Web Consortiums
- Internet Explorer und Netscape

1. History and Development - History

- 1996: CSS Level 1 wurde eine W3C Recommendation
- 1997: eigen Arbeitsgruppe für CSS
- 1998: bekam CSS Level 2 den Recommendation Status
- 1999 – 2016: Mitglieder der Arbeitsgruppe von 15 auf 115 gestiegen

1. History and Development – Development Process

- World Wide Web Consortium (W3C)
  - wichtigstes Institut für Standardisierung im World Wide Web

- Der Entwicklungsprozess besteht aus drei Stufen:
  - Working Draft (WD)
  - Candidate Recommendation (CR)
  - Recommendation (REC)

1. History and Development – Development Process

Working Draft

- Erste Design Phase einer neuen Spezifikation
- Startet mit dem First Public Working Draft (FPWD)
- Endet mit dem Last Call Working Draft (LCDW)

1. History and Development – Development Process

Candidate Recommendation

- Testen und Implementieren
- Zwei korrekte und unabhängige Implementierungen
- Proposed Recommendation (PR)

Recommendation

- Spezifikation ist vollständig
- Wartung und Fehlerbehebung

1. History and Development – CSS Levels

CSS Level 1

- CSS1 Spezifikation

CSS Level 2

- CSS2 war sehr fehlerhaft
- Überarbeitung von CSS2 –> CSS2.1
- CSS2.1 enthielt weniger Möglichkeiten

1. History and Development – CSS Levels

CSS Level 3

- Basiert auf CSS2.1
- Module werden unabhängig voneinander weiterentwickelt

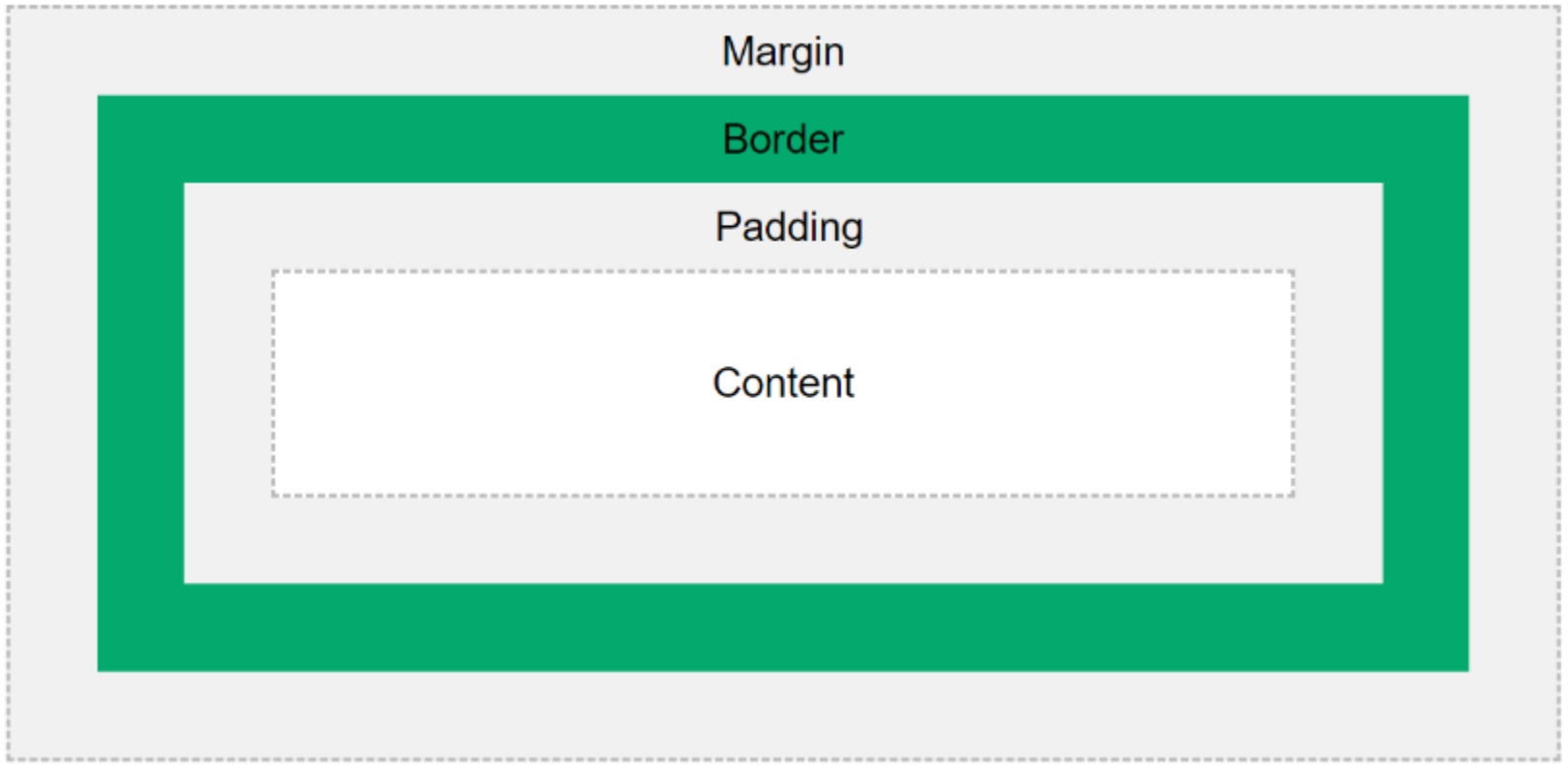
CSS Level 4

- Nicht existent für CSS als gesamtheitliche Sprache gesehen
- Individuelle Module können Level 4 oder höher erreichen

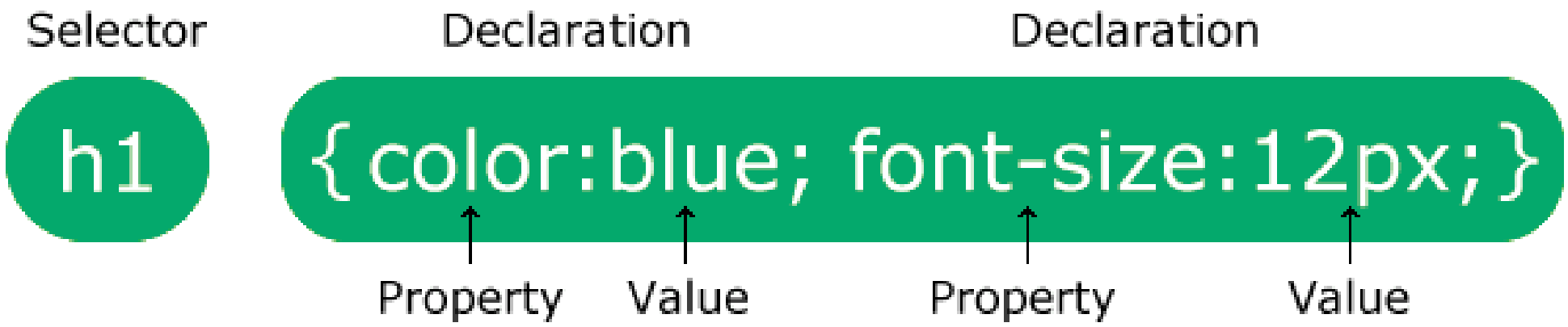
2. Operating Principle - Concept

- Zweck: Darstellung von Inhalten
- Trennung von Struktur und Darstellung des Inhalts
- Auf alle XML basierte Formate anwendbar
  - Hauptsächlich HTML
  - Nutzung der Document Type Definition

2. Operating Principle – Box-Model



2. Operating Principle - Syntax



2. Operating Principle - Selectors

- Mit den Selectors werden die zu stylenden Elemente definiert

- 5 Kategorien:
  - Simple selectors
  - Combinator selectors
  - Pseudo-class selectors
  - Pseudo-element selectors
  - Attribute selectors

2. Operating Principle – Selectors

Simple Selectors

```
p{
  font-size: 14px;
  color: grey;
}
```

Simple Selector

```
#menu1{
  font-size: 14px;
  color: grey;
}
```

ID Selector

2. Operating Principle – Selectors

Simple Selectors

```
.sidebar{
  font-size: 14px;
  color: grey;
}
```

Class Selector

```
p.sidebar{
  font-size: 14px;
  color: grey;
}
```

Simple und Class Selector kombiniert

2. Operating Principle – Selectors

Simple Selectors

```
*{
    font-size: 14px;
    color: grey;
}
```

Universal Selector

2. Operating Principle – Selectors

Combinator Selectors

```
div ul{
    color: red;
    padding-left: 10px;
}
```

Descendant Selector

```
div>p{
    font-weight: bold;
    background-color: seashell;
}
```

Child Selector

2. Operating Principle – Selectors

Combinator Selectors

```
div+a{
    font-size: 14px;
    font-weight: bold;
}
```

Adjacent Sibling Selector

```
div~p{
    color: lightskyblue;
    font-weight: 500;
}
```

General Sibing Selector

2. Operating Principle – Selectors

Pseudo-Class Selectors

```
selector:pseudo-class{
    property: value
}
```

Pseudo-Element Selectors

```
selector::pseudo-element{
    property: value
}
```

2. Operating Principle – Selectors

Attribute Selectors

```
a[href]{
  color: red;
}
```

Selektor für bestimmte Attribute

```
[title="comment"]{
  font-size: 12px;
}
```

Selektor für bestimmte Attribute mit bestimmtem Wert

2. Operating Principle – Selectors

Attribute Selectors

```
p[title~="CSS"]{
  font-weight:bold;
}
```

Selektor für bestimmte Attribute deren Wert ein bestimmtes Wort enthält

```
p[title*="Casca"]{
  font-weight:bold;
}
```

Selektor für bestimmte Attribute deren Wert einen bestimmten String enthält

2. Operating Principle – Selectors

Attribute Selectors

```
[class|="side"]{
  border: burlywood;
}
```

Selektor für bestimmte Attribute deren Wert mit einem bestimmten Wort beginnt

```
[class^="side"]{
  border: burlywood;
}
```

Selektor für bestimmte Attribute deren Wert mit einem bestimmten String beginnt

2. Operating Principle – Selectors

Attribute Selectors

```
[class$="bar"]{
  border: blue;
}
```

Selektor für bestimmte Attribute
deren Wert mit einem
bestimmten String endet

2. Operating Principle – Inserting CSS

Einfügen in HTML – External

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="external.css">
</head>
<body>
```

2. Operating Principle – Inserting CSS

Einfügen in HTML – Internal

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: grey;
}
</style>
</head>
```

2. Operating Principle – Inserting CSS

Einfügen in HTML – Inline

```
<body>

  <p style="background-color: beige">This is inline CSS</p>

</body>
```

2. Operating Principle – Inserting CSS

Einfügen in XML

- Unterschiede zu HTML
- Keine vordefinierten Elemente (DTD)
- Browser kann Elemente nicht erkennen
- CSS kann nicht „Inline“ eingebunden werden

2. Operating Principle – Inserting CSS

Einfügen in XML

```
<?xml version="1.0"?>
<?xml-stylesheet href="CSSandXML.css" type="text/css"?>

    <!DOCTYPE html>
    <html>
    <head>
    <link rel="stylesheet" href="external.css">
    </head>
    <body>
```

3. Example of Use - Frameworks

Komponenten

- Grids
- Typographie Elemente
- Cross-Browser Kompabilität
- Hilfsklassen
- Navigationselemente
- Pre-Processors
- Mediaelemente

3. Example of Use - Frameworks

Vorteile

- Reduzierte Menge an Code
- Kürzere Entwicklungszeit
- Geringerer Wartungsaufwand
- Erleichtert Veränderbarkeit und Erweiterbarkeit
- Einheitliche Codestruktur

3. Example of Use – Nutshell Examples

3. Example of Use – Nutshell Examples

Allgemeines, Importierte Schrift und Untertitel

```
@import url('https://fonts.googleapis.com/css2?family=Archivo:wght@900&display=swap');
```

```
*{
    box-sizing:border-box;
    margin:0;
    padding:0;
}
```

```
.sub-title{
    font-size: 60px;
    padding-top:80px;
}
```

3. Example of Use – Nutshell Examples

Navigationsleiste – Container und Text


```css
header{
    display: flex;
    justify-content:space-between;
    align-items:center;
    overflow:hidden;
    position: fixed;
    top:0;
    width:100%;
    padding-left:60px;
    background: rgb(74, 132, 155);
    height:70px;
}
```

```css
div.head-text{
    font-family:"Archivo", sans-serif;
    font-weight: 900;
    font-size:30px;
    color: white;
    display:flex;
    justify-content: flex-start;
}
```

3. Example of Use – Nutshell Examples

Navigationsleiste – Text und Links

```css
span p::first-letter{
    color: rgb(152, 205, 255);
    font-size: 40px;
}
```

```css
.nav-links li, a{
    font-family:Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size:20px;
    color: white;
    text-decoration: none;
    list-style: none;
    display:inline;
    padding: 0px 10px;
}
```

```css
.nav-links li a:hover{
    color: rgb(152, 205, 255);
    background-color: rgb(0, 46, 59);
    border-radius: 10mm;
    padding:15px;
    transition-duration: 0.8s;
}
```

3. Example of Use – Nutshell Examples

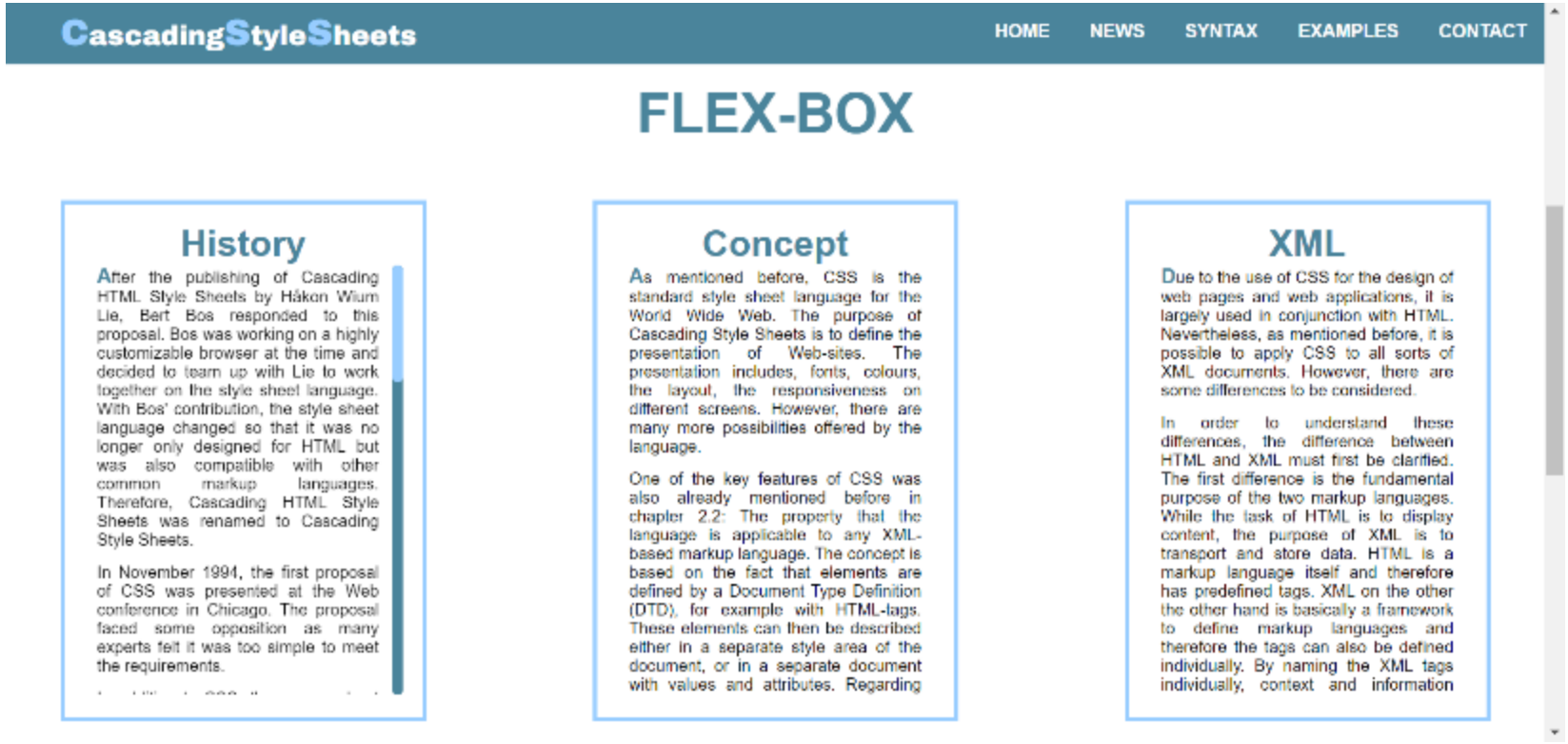
Picture – Text und Bild


```css
.top-pic-text{
    background:none;
    font-size:100px;
    color: rgb(1, 35, 131);
    margin:0;
    font-weight: 900;
    font-family: 'Archivo', sans-serif;
}
```

```css
.cont-top-pic{
    margin-top: 70px;
    height:500px;
    background-image:url(Pictures/code.jpg);
    background-position: center left;
    display:flex;
    align-items: center;
    justify-content: center;
}
```

3. Example of Use – Nutshell Examples

Flexbox



3. Example of Use – Nutshell Examples

Flexbox – Flex-Container und Spalten

```css
.flex-cont-text{
    display: flex;
    justify-content: space-between;
    margin-top:50px;
    margin-left:50px;
    margin-right:50px;
}
```

```css
[class|="column"]{
    background: white;
    border:5px solid rgb(152, 205, 255);
    margin:10px;
    width:400px;
    padding:20px;
    height:570px;
}
```

3. Example of Use – Nutshell Examples

Flexbox – Text-Container und Scroll-Bar

```css
.pad{
    height:470px;
    overflow-y: hidden;
}
.pad:hover{
    overflow-y: scroll;
}
```

```css
.pad::-webkit-scrollbar {
    width: 12px;
}

.pad::-webkit-scrollbar-track {
    background: rgb(74, 132, 155);
    border-radius: 20px;
}

.pad::-webkit-scrollbar-thumb {
    background-color: rgb(152, 205, 255);
    border-radius: 20px;
    border: 3px solid rgb(152, 205, 255);
}
```

3. Example of Use – Nutshell Examples

Grid-System

3. Example of Use – Nutshell Examples

Grid – Grid-Container und Items

```css
.grid-container{
    display:grid;
    grid: 150px 150px/auto auto auto;
    grid-row-gap: 10px;
    grid-column-gap: 55px;
    margin: 60px;
    padding-top:20px;
    padding-bottom:20px;
}
```

```css
[class|="grid-item"]{
    display:flex;
    justify-content: center;
    align-items:center;
}
```

3. Example of Use – Nutshell Examples

Grid – Gestaltung des Inhalts

```
170  [class|="grid-item"] p{
171      color: rgb(74, 132, 155);
172      font-family: Arial, Helvetica, sans-serif;
173      font-size: 20px;
174      text-align:justify;
175  }
176  img.logo{
177      width:100px;
178  }
179  .border-grid{
180      border: rgb(152, 205, 255) 3px solid;
181      margin:3px;
182  }
```

4. Summary

- Standard im World Wide Web
- Auf alle XML basierte Formate anwendbar
- Einfache Syntax und Box-Model
  - Unmenge an Eigenschaften und Werten
- Frameworks
- Stetige Weiterentwicklung von einzelnen Modulen
- Neuheiten wie Pre-Processors

Diskussion und Fragen