



JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN
PROFESSUR BWL – WIRTSCHAFTSINFORMATIK
UNIV.-PROF. DR. AXEL SCHWICKERT

Schwickert, Axel; Schramm, Laura; Patzak, Maximilian;
Schick, Lukas

HTML, CSS und JavaScript – Teil 2 – Reader zur WBT-Serie

ARBEITSPAPIERE WIRTSCHAFTSINFORMATIK

Nr. 02/2018
ISSN 1613-6667

Arbeitspapiere WI Nr. 2 / 2018

- Autoren:** Schwickert, Axel; Schramm, Laura; Patzak, Maximilian; Schick, Lukas
- Titel:** HTML, CSS und JavaScript – Teil 2 – Reader zur WBT-Serie
- Zitation:** Schwickert, Axel; Schramm, Laura; Patzak, Maximilian; Schick, Lukas: HTML, CSS und JavaScript – Teil 2 – Reader zur WBT-Serie, in: Arbeitspapiere WI, Nr. 2/2018, Hrsg.: Professur BWL – Wirtschaftsinformatik, Justus-Liebig-Universität Gießen 2018, 363 Seiten, ISSN 1613-6667.
- Kurzfassung:** Das vorliegende Arbeitspapier dient als Reader zur WBT-Serie „HTML, CSS und JavaScript – Teil 2“, die im E-Campus Wirtschaftsinformatik online zur Verfügung steht. Die WBT-Serie baut auf der WBT-Serie „HTML, CSS und JavaScript – Teil 1“ auf und vertieft die Arbeit mit der Erstellung von Web Sites. Am Beispiel des Web-shops „Casarella“ werden Funktionen wie HTML-Formulare, das CSS-Grid-Layout oder Variablen in Java-Script erläutert. Begleitet von praktischen Übungen im Text-Editor kann das Erlernete direkt umgesetzt werden.
- Schlüsselwörter:** Erstellung von Formularen mit HTML, Erstellung verschiedener Formularfelder mit HTML, Seitenstruktur mit Media Queries, Gestaltung von Formularen mit CSS, Responsives Web Design, Seitenstrukturierung mit CSS-Grid, Variablen und Rechnen mit JavaScript, Interaktivität und Zeichentheorie mit JavaScript, Bedingungen mit JavaScript

A Die Web-Based-Trainings

Der Lernstoff zum Themenbereich „HTML, CSS & JavaScript – Teil 2“ wird durch eine Serie von Web-Based-Trainings (WBT) vermittelt. Die WBT bauen inhaltlich aufeinander auf und sollten daher in der angegebenen Reihenfolge und zum vorgesehenen Zeitpunkt absolviert werden. Um einen Themenbereich vollständig durchdringen zu können, muss jedes WBT mehrfach absolviert werden, bis die jeweiligen Tests in den einzelnen WBT sicher bestanden werden.

WBT-Nr.	WBT-Bezeichnung	Dauer
1	Einführung in HTML – Teil 2	60
2	Formulare	60
3	Radio-Buttons und Checkboxes	60
4	Select-Tag und Datalist-Tag	60
5	Seitenstruktur mit Media Queries	60
6	CSS für Formulare	60
7	CSS-Grid-Layout	60
8	Rechnen mit JavaScript	60
9	Variablen in JavaScript	60
10	Funktionen und Fehleranalyse in JavaScript	60
11	Zeichentheorie	60
12	Bedingungen in JavaScript	60

Tab. 1: Übersicht der WBT-Serie

Die Inhalte der einzelnen WBT werden nachfolgend in diesem Dokument gezeigt. Alle WBT stehen Ihnen rund um die Uhr online zur Verfügung. Sie können jedes WBT beliebig oft durcharbeiten.

Inhaltsverzeichnis

	Seite
A Die Web-Based-Trainings	II
Inhaltsverzeichnis	III
Abbildungsverzeichnis	VIII
Tabellenverzeichnis	XII
1 Einführung in HTML – Teil 2	1
1.1 Einleitung	1
1.2 Der neue Auftrag von Casarella	1
1.2.1 Der neue Auftrag	1
1.2.2 Die Web Site von Casarella	2
1.2.3 Die neuen Anforderungen	2
1.2.4 Die Umsetzung der neuen Anforderungen	3
1.2.5 Anwendung und Nutzung von Formularen – Kontakt	3
1.2.6 Anwendung und Nutzung von Formularen – Bestellungen	5
1.3 XAMPP Installation	6
1.4 Ausblick	6
2 Formulare	7
2.1 Einleitung	7
2.2 Formulare in HTML	7
2.2.1 Der HTML5-Tag für Formulare	7
2.2.2 Wohin werden die Daten geschickt? – Das action-Attribut	8
2.2.3 Wie werden die Daten an den Server geschickt? – Die Methoden „get“ und „post“	8
2.2.4 Die Methoden „get“ und „post“	9
2.3 Der <input>-Tag	10
2.3.1 Was wird an den Server geschickt? – Der <input>-Tag	10
2.3.2 Eingabemöglichkeiten mit dem <input>-Tag	10
2.3.3 Attribute des <input>-Tag – Teil 1	11
2.3.4 Attribute des <input>-Tag – Teil 2	12
2.3.5 Der <label>-Tag	13
2.3.6 Der Absende-Button	14
2.4 Übung	15
2.4.1 Das Gelernte anwenden	15
2.4.2 Vorbereitung für die Übungsaufgaben des HTML-Teils	16
2.4.3 Übungsaufgabe	16

2.5	Ausblick.....	17
3	Radio-Buttons und Checkboxes.....	18
3.1	Einleitung	18
3.2	Radio-Buttons	18
3.2.1	Schwierigkeiten beim Ausfüllen der freien Formularfelder.....	18
3.2.2	Radio-Buttons	19
3.2.3	Der input-Typ „radio“	19
3.2.4	Eine Vorauswahl anbieten – Das Attribut „checked“	20
3.2.5	Übung zu Radio-Buttons	21
3.3	Checkboxes	22
3.3.1	Checkboxes	22
3.3.2	Der input-Typ „checkbox“	23
3.3.3	Übungen zu Checkboxes.....	24
3.4	Zusatz und Übung.....	25
3.4.1	Zusatz zu Radio-Buttons und Checkboxes – <fieldset>	25
3.4.2	Das Gelernte anwenden	26
3.4.3	Umsetzung der Formular-Anforderungen – Teil 1	26
3.5	Ausblick.....	27
4	Select-Tag und Datalist-Tag	28
4.1	Einleitung	28
4.2	Select-Tag.....	28
4.2.1	Die Select-Box.....	28
4.2.2	Der <select>-Tag	29
4.2.3	Der <optgroup>-Tag.....	31
4.2.4	Attribute von Select-Boxen: „multiple“ und „size“	32
4.2.5	Übung zu Select-Boxen	33
4.3	Datalist-Tag	34
4.3.1	Die Datenliste	34
4.3.2	Der <datalist>-Tag.....	35
4.3.3	Übung zu Datenlisten	36
4.4	Übung	37
4.4.1	Das Gelernte anwenden	37
4.4.2	Umsetzung der Formular-Anforderungen – Teil 2.....	37
4.4.3	Umsetzung der Formularanforderungen – Teil 3	38
4.5	Ausblick.....	40

5	Seitenstruktur mit Media Queries	41
5.1	Einleitung	41
5.2	Web-Seiten auf verschiedenen Anzeigegeräten	41
5.2.1	Web-Seiten auf verschiedenen Anzeigegeräten – Teil 1	41
5.2.2	Web-Seiten auf verschiedenen Anzeigegeräten – Teil 2	41
5.2.3	Responsive Webdesign – Teil 1	42
5.2.4	Responsive Webdesign – Teil 2	43
5.2.5	Adaptives Webdesign	43
5.3	Media Queries	44
5.3.1	Erkennung des Anzeigegerätes	44
5.3.2	Media Queries in HTML	44
5.3.3	Aufbau von Media Queries – Teil 1	45
5.3.4	Aufbau von Media Queries – Teil 2	45
5.4	Übung	46
5.4.1	Das Gelernte festigen	46
5.4.2	Abschlusstest	47
5.4.3	Typische Klausurfrage	48
5.5	Ausblick	48
6	CSS für Formulare	49
6.1	Einleitung	49
6.2	Formulare mit CSS gestalten	49
6.2.1	Formulare mit CSS gestalten	49
6.2.2	Pseudoklassen zur Gestaltung von Formularen	49
6.2.3	Pflichtfelder in CSS gestalten	50
6.2.4	Fehlerhafte Eingaben	51
6.2.5	Fehlerhafte Eingaben in Formularen hervorheben	52
6.2.6	Fokussierung von Formularfeldern	53
6.2.7	Formularfelder hervorheben, die in Bearbeitung sind	53
6.2.8	Formularfelder bei Mausover gestalten	54
6.3	Das Wissen vertiefen	55
6.3.1	Anforderungen an die Web Site umsetzen	55
6.3.2	Abschlusstest	57
6.3.3	Typische Klausurfrage	58
6.4	Ausblick	58
7	CSS-Grid-Layout	59
7.1	Einleitung	59
7.2	Seitenstruktur mit CSS Grid	59
7.3	Gestaltung der Seitenstruktur mit dem CSS-Grid-Layout	59

7.4	Die Raster-Bereiche im CSS-Grid-Layout	61
7.5	CSS Grid verknüpfen mit dem HTML-Quelltext.....	62
7.6	CSS-Grid-Areas den Bereichen in HTML zuweisen	63
7.7	Zeilen und Spalten in CSS-Grid gestalten.....	63
7.8	Relative Größenangaben mit „fraction“	64
7.9	CSS-Grid-Layout.....	65
7.10	Elemente im CSS-Grid positionieren	65
7.11	Leere Bereiche im Raster kennzeichnen	67
7.12	Abschluss.....	67
8	Rechnen mit JavaScript	68
8.1	Casarella braucht JavaScript.....	68
8.2	JavaScript – Wir erinnern uns	68
8.3	JavaScript direkt in das HTML-Dokument schreiben.....	68
8.4	JavaScript-Dateien in HTML-Dokumente einbinden.....	69
8.5	JavaScript – Noscript.....	69
8.6	Objekte und Methoden – Befehle in JavaScript	70
8.7	Objekte und Methode in JavaScript – "window.alert"	70
8.8	Objekte und Methode in JavaScript – "document.write"	71
8.9	Datentypen.....	72
8.10	JavaScript kann rechnen	73
8.11	Rechnen mit JavaScript	73
8.12	Rechnen mit JavaScript – Teil 2.....	74
8.13	Übung – Rechnen mit JavaScript	75
8.14	Abschlusstest	76
9	Variablen in JavaScript	77
9.1	Casarella möchte Größentabellen erweitern.....	77
9.2	Was sind Variablen in JavaScript?	77
9.3	Die const-Variable	78
9.4	Was sind Variablen in JavaScript?	78
9.5	Die var-Variable	79
9.6	Die richtige Variable für Casarella.....	79
9.7	Variablen können auch mit Zahlen befüllt werden	79
9.8	Übung – Variable schreiben	80
9.9	Übung – Umrechnung der Ringgrößen	80
9.10	Abschlusstest	82

10	Interaktivität und Fehleranalyse.....	83
10.1	Einleitung – Funktionen für die Casarella Web-Site.....	83
10.2	Funktionen in JavaScript	83
10.3	Ging das zu schnell?	83
10.4	Aufbau einer Funktion in JavaScript.....	83
10.5	Welches Verfahren eignet sich zur Darstellung der Funktion?.....	84
10.6	Darstellung von Funktionen durch IDs	85
10.7	Ergebnisdarstellung der JavaScript-Funktion.....	85
10.8	Übung – Funktion schreiben.....	86
10.9	Fehleranalyse in JavaScript	86
10.10	Fehleranalyse mit Google Chrome	87
10.11	Fehleranalyse mit Firefox	88
10.12	Abschlusstest	89
11	Zeichenketten und Eingabefelder	90
11.1	Zeichentheorie	90
11.1.1	Einführung	90
11.1.2	Strings.....	90
11.1.3	Verkettung von Strings.....	90
11.1.4	Eigenschaften von Strings	91
11.1.5	Methoden von Strings.....	91
11.1.6	Methode charAt()	91
11.1.7	Methoden indexOf() und lastIndexOf().....	92
11.1.8	Methoden indexOf() und lastIndexOf() – Teil 2	92
11.1.9	Methoden – Übung	93
11.2	Eingabefelder mit JavaScript	93
11.2.1	Eingabefelder.....	93
11.2.2	Eingabefelder des Typs "number" – HTML-Elemente	94
11.2.3	Eingabefelder des Typs "number" – JavaScript-Elemente.....	94
11.2.4	Eingabefelder – Browser-Ansicht	95
	Anhang.....	IX

Abbildungsverzeichnis

	Seite
Abb. 1: Web Site Casarella.....	2
Abb. 2: Informationsfluss vom Server zum Endgerät	3
Abb. 3: Informationsfluss vom Server zum Endgerät und vom Endgerät zum Server.....	3
Abb. 4: Grafische Gestaltung des neuen Kontaktformulars	4
Abb. 5: Grafische Gestaltung des neuen Bestellformulars	5
Abb. 6: HTML5-Tag für Formulare	7
Abb. 7: Das action-Attribut – HTML	8
Abb. 8: Die Methode „get“	8
Abb. 9: Die Methode „post“	9
Abb. 10: HTML-Code zur Methode „get“	9
Abb. 11: Der <input>-Tag	10
Abb. 12: Attribute im <input>-Tag – HTML	11
Abb. 13: Attribute im <input>-Tag – Browserdarstellung	11
Abb. 14: value – HTML.....	12
Abb. 15: value – Browserdarstellung.....	13
Abb. 16: password – HTML	13
Abb. 17: password – Browserdarstellung	13
Abb. 18: Der <label>-Tag – HTML	14
Abb. 19: Der <label>-Tag – Browserdarstellung	14
Abb. 20: Der Absende-Button – HTML	15
Abb. 21: Der Absende-Button – Browserdarstellung	15
Abb. 22: Bestellformular	16
Abb. 23: Radio-Buttons – HTML.....	19
Abb. 24: Radio-Buttons – Browseransicht	19
Abb. 25: Der input-Typ „radio“ – HTML.....	20
Abb. 26: Das Attribut „checked“ – HTML.....	21
Abb. 27: Das Attribut „checked“ – Browseransicht	21
Abb. 28: Übung zu Radio-Buttons.....	22
Abb. 29: Checkbox – Browseransicht.....	22
Abb. 30: Der input-Typ „checkbox“ – HTML	23
Abb. 31: Der input-Typ „checkbox“ – Browseransicht.....	24
Abb. 32: Übung zu Checkboxes.....	24
Abb. 33: Das <fieldset>-Element – Browseransicht.....	25
Abb. 34: Das <fieldset>-Element – HTML	26
Abb. 35: Formularanforderungen – Teil 1	27

Abb. 36:	Die Select-Box – Browseransicht	28
Abb. 37:	Klick auf die Select-Box – Browseransicht	29
Abb. 38:	Der <select>-Tag – HTML	29
Abb. 39:	Der <select>-Tag – Browseransicht.....	30
Abb. 40:	Vorauswahl beim <select>-Tag – HTML.....	30
Abb. 41:	Vorauswahl beim <select>-Tag – Browseransicht	30
Abb. 42:	Der <optgroup>-Tag – Browseransicht	31
Abb. 43:	Der <optgroup>-Tag – HTML.....	31
Abb. 44:	Die Attribute „multiple“ und „size“ – HTML	32
Abb. 45:	Die Attribute „multiple“ und „size“ – Browseransicht.....	33
Abb. 46:	Übung zu Select-Boxen	33
Abb. 47:	Lösung zu Select-Boxen	34
Abb. 48:	Die Datenliste.....	35
Abb. 49:	Der <datalist>-Tag – HTML.....	35
Abb. 50:	Formularanforderungen – Teil 2.....	38
Abb. 51:	Formularanforderungen – Teil 3	39
Abb. 52:	Eine Web Site auf verschiedenen Anzeigegeräten	42
Abb. 53:	Responsive Webdesign	43
Abb. 54:	Media Queries in HTML	44
Abb. 55:	Aufbau von Media Queries	45
Abb. 56:	Pseudoklasse zu Pflichtfeldern – Browseransicht.....	50
Abb. 57:	Pseudoklasse zu Pflichtfeldern – HTML	50
Abb. 58:	Pseudoklasse zu Pflichtfeldern – CSS	50
Abb. 59:	Input:required und input:optional – HTML.....	51
Abb. 60:	Input:required und input:optional – CSS	51
Abb. 61:	Fehlerhafte Eingaben in Formularfelder hervorheben – HTML.....	52
Abb. 62:	Fehlerhafte Eingaben in Formularfelder hervorheben – CSS.....	52
Abb. 63:	Fehlerhafte Eingaben in Formularfelder hervorheben – HTML & CSS	53
Abb. 64:	:focus – HTML	54
Abb. 65:	:focus – CSS.....	54
Abb. 66:	:focus – HTML & CSS	54
Abb. 67:	:hover – HTML	54
Abb. 68:	:hover – CSS	55
Abb. 69:	:hover – HTML & CSS.....	55
Abb. 70:	Formularanforderungen – Teil 4	56
Abb. 71:	Wireframe der Casarella-Web-Site.....	60
Abb. 72:	Struktur der Casarella-Web-Site	60
Abb. 73:	CSS-Grid-Raster	61

Abb. 74:	CSS-Grid-template-areas	61
Abb. 75:	HTML-Dokument mit der Klasse „container“	62
Abb. 76:	Zuweisung der Web-Seiten-Bereiche	63
Abb. 77:	Grid-template-rows	63
Abb. 78:	Grid-template-culums	64
Abb. 79:	Monitorauflösung und die Einheit „fraction“	65
Abb. 80:	Zeilen, Spalten und Abstände in CSS-Grid gestalten	65
Abb. 81:	Elemente in CSS-Grid positionieren.....	66
Abb. 82:	Rasterlinien	66
Abb. 83:	Leere Bereiche in CSS-Grid	67
Abb. 84:	Der HTML-Tag <script>	68
Abb. 85:	Einbindung von JavaScript in HTML-Quelltext.....	69
Abb. 86:	Der HTML-Tag <noscript>	69
Abb. 87:	Bestandteile eines JavaScript-Befehls	70
Abb. 88:	Dialogfenster auf der Web Site Casarella.....	71
Abb. 89:	Inhalte dargestellt im Browser durch document.write	71
Abb. 90:	Document.write im Quelltext.....	72
Abb. 91:	Window.alert im Texteditor.....	73
Abb. 92:	Window.alert im Browser	74
Abb. 93:	Document.write im Texteditor.....	74
Abb. 94:	Document.write im Browser	75
Abb. 95:	Variablen in JavaScript	77
Abb. 96:	Aufbau einer Funktion in JavaScript	84
Abb. 97:	Darstellung von Funktionen.....	85
Abb. 98:	Lösung der Übung.....	86
Abb. 99:	Quelltext zur Fehleranalyse	87
Abb. 100:	Fehleranalyse mit Google Chrome	87
Abb. 101:	Fehleranalyse mit Firefox	88
Abb. 102:	Verkettung von Strings im Browser	91
Abb. 103:	Beispiele der Methode charAt().....	92
Abb. 104:	Beispiele von Methoden in JavaScript.....	92
Abb. 105:	Beispiele von Methoden in JavaScript – Teil 2	93
Abb. 106:	HTML-Elemente eines Eingabefeldes	94
Abb. 107:	JavaScript-Elemente eines Eingabefeldes.....	95
Abb. 108:	Eingabefelder im Browser	95
Abb. 109:	Bedingungen in JavaScript	98
Abb. 110:	Operatoren in JavaScript.....	98
Abb. 111:	Beispielhaftes if-else-Statement.....	99

Abb. 112:	Beispiel der Negations-Bedingung	100
Abb. 113:	Lösung if-else-Übung	101

Tabellenverzeichnis

Tab. 1:	Übersicht der WBT-Serie	II
Tab. 2:	Übung zu Datenlisten	36
Tab. 3:	Abschlusstest WBT 05	48
Tab. 4:	Abschlusstest WBT 06	58
Tab. 5:	Abschlusstest WBT 08	76
Tab. 6:	Abschlusstest WBT 09	82
Tab. 7:	Abschlusstest WBT 10	89
Tab. 8:	Abschlusstest WBT 11	96
Tab. 9:	Lösung zu Datenlisten	IX
Tab. 10:	Lösung zu Abschlusstest WBT 05.....	XI
Tab. 11:	Lösung zu Abschlusstest WBT 06.....	XIII
Tab. 12:	Lösung zu Abschlusstest WBT 08.....	XIV
Tab. 13:	Lösung zu Abschlusstest WBT 09.....	XV
Tab. 14:	Lösung zu Abschlusstest WBT 10.....	XVI
Tab. 15:	Lösung zu Abschlusstest WBT 11.....	XVII

1 Einführung in HTML – Teil 2

1.1 Einleitung

Lin W. Lan: „Hallo, ich bin es wieder, Lin W. Lan. In der ersten WBT-Serie "HTML, CSS & JavaScript – Teil 1" sind Sie mir schon tatkräftig zur Hand gegangen und haben mich bei der Web-Site-Gestaltung des Start-up-Unternehmens Casarella unterstützt.

Casarella hat die gemeinsame Zusammenarbeit sehr gefallen und bittet uns nun erneut um Hilfe, um einige Anpassungen an der Web Site vorzunehmen.

Meine Mitarbeiter, Herr Neumann und Herr Huber, haben bereits erste Gespräche mit Casarella geführt und werden Ihnen auf den nächsten Seiten den neuen Auftrag erläutern.

Ich hoffe, dass Sie mir bei diesem Auftrag wieder unterstützend zur Seite stehen und wir gemeinsam die Wünsche von Casarella erfüllen können!“

1.2 Der neue Auftrag von Casarella

1.2.1 Der neue Auftrag

Herr Neumann: „Hallo! Ich bin Herr Neumann, der Projektleiter von Lin W. Lan. Ich habe bereits das letzte Projekt von Casarella betreut und abgewickelt. Lin W. Lan hat mich gebeten, Sie schon einmal mit den neuen Anpassungsanforderungen vertraut zu machen: Das Kleinunternehmen Casarella hat seit Bestehen der unternehmenseigenen Web Site viele neue Kunden dazugewonnen. Ihre harte Arbeit bei der Erstellung dieser Web Site zahlt sich also aus!

Die Kunden und Interessenten von Casarella können sich mit Hilfe der neuen Internetpräsenz über das Unternehmen und die angebotenen Produkte informieren und nehmen dieses Angebot dankend an. Mit dem Wachsen des Kundenstamms hört Casarella jedoch immer häufiger von ihren Kunden, dass diese gerne auch die Möglichkeit hätten, direkt über die Web Site mit den Mitarbeitern von Casarella in Kontakt zu treten oder Produkte zu bestellen.

Für ein erfolgreiches Geschäft ist es sehr wichtig, dass Kontaktaufnahmen über die üblichen Kontaktzeiten und über das Telefon hinausgehen. Somit sollten auch Bestellungen online und zu jeder Zeit durchzuführen sein.

Daher bittet Casarella Sie, schnellstmöglich mit der Umsetzung einiger Anpassungen auf der Web Site zu beginnen, um den Kundenanforderungen gerecht zu werden.“

1.2.2 Die Web Site von Casarella

Lin W. Lan: Unsere neue Aufgabe steht also! Zur Erinnerung hier noch einmal die bisherige Web Site von Casarella. Die Arbeit ist uns bisher wirklich sehr gut gelungen!

Hinweis: An dieser Stelle im WBT können Sie sich den Quelltext der bisherigen Web Site von Casarella herunterladen.

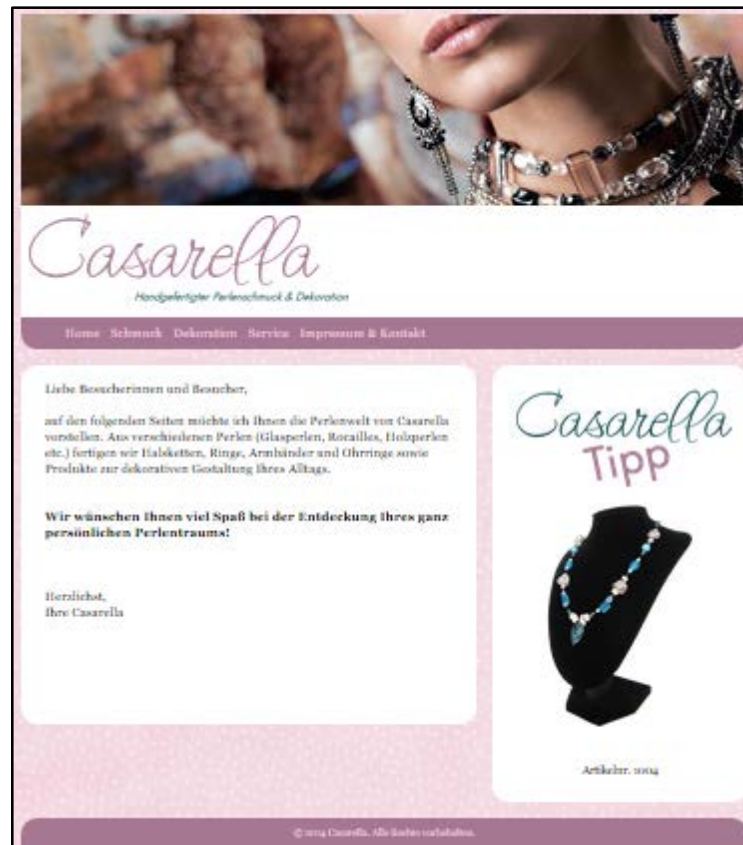


Abb. 1: Web Site Casarella

1.2.3 Die neuen Anforderungen

Die bisherige Web Site von Casarella ist eine reine "Lese-Web-Site". Das bedeutet, dass durch die Eingabe einer URL die Web Site angezeigt wird und die Interessenten sich lediglich über Casarella und ihre Produkte informieren können.

Hierbei besteht das Problem, dass die Web Site Informationen nur vom Server zum Endgerät überträgt. Die Informationen fließen also nur in eine Richtung.

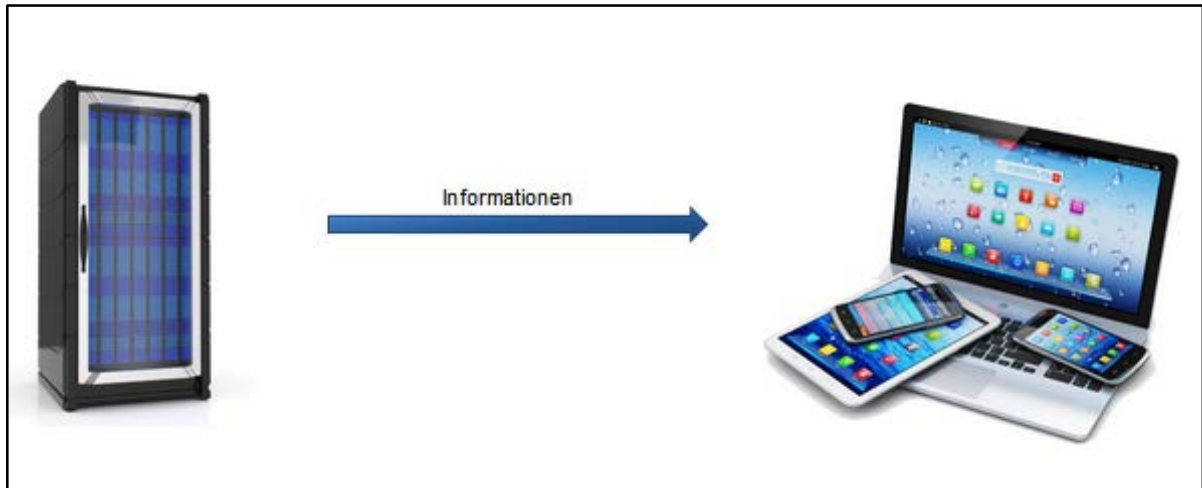


Abb. 2: Informationsfluss vom Server zum Endgerät

1.2.4 Die Umsetzung der neuen Anforderungen

Damit Interessenten mit der Web Site kommunizieren können, müssen Informationen auch in die andere Richtung geschickt werden. Das Endgerät des Interessenten muss die Informationen also auch an den Server von Casarella senden können.

Dafür bieten sich Formulare an, die es ermöglichen, Kontaktanfragen oder Bestellungen von Interessenten an den Server zu senden.

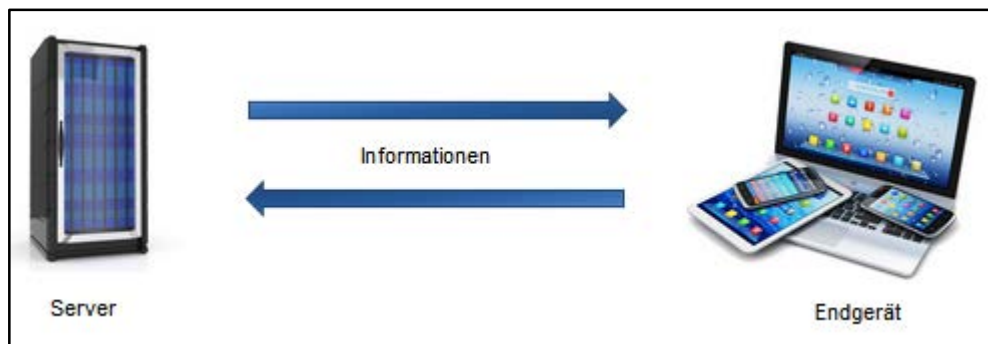


Abb. 3: Informationsfluss vom Server zum Endgerät und vom Endgerät zum Server

Lin W. Lan: „Ein Formular besteht aus Eingabefeldern auf einer Web Site zur Erfassung von Texten oder Zahlen. Auf einer unternehmenseigenen Web Site ermöglichen sie es somit, dem Unternehmen bestimmte Informationen aus den Eingabefeldern der Formulare zu übermitteln.“

1.2.5 Anwendung und Nutzung von Formularen – Kontakt

Herr Huber: „Formulare können für verschiedene Zwecke genutzt werden. In Absprache mit Casarella habe ich bereits ein neues Kontaktformular grafisch gestaltet, welches jedoch noch nicht in HTML umgesetzt ist.“

Casarella wünscht sich auf der neuen Impressumsseite eine direkte Kontaktmöglichkeit mit den Interessenten.

Die Interessenten sollen neben der Nachricht an Casarella auch ihren Namen und ihre E-Mail-Adresse eingeben müssen.

Mit der Nutzung solcher Pflichtfelder wird sichergestellt, dass der Interessent seinen Namen und seine E-Mail-Adresse auch wirklich angibt.

So wird den bisherigen Problemen bei der Kontaktaufnahme der Kunden durch E-Mails entgegengewirkt, in denen die Nachrichten anonym gestaltet sein konnten.“

Impressum & Kontakt

Casarella

Bella Perle
Am Schmuckweiher 14
35394 Gießen, Deutschland

Telefon: 0172/1239702
E-Mail: bella.perle@casarella.de

Alle auf der Web Site von Casarella angegebenen Preise sind Endpreise zzgl. Versandkosten. Aufgrund des Kleinunternehmertums gem. §19 UStG erheben wir keine Umsatzsteuer und weisen diese daher auch nicht aus.

Hinweise zum Versand: Versand innerhalb Deutschland möglich, per Deutsche Post Warensendung. Versandkosten 1,90 €.

Sie möchten direkt Kontakt mit uns aufnehmen? Gerne können Sie mir über das nachstehende Kontaktformular Ihre Nachricht zukommen lassen.

Name*

E-Mail-Adresse*

Ihre Nachricht*

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 4: Grafische Gestaltung des neuen Kontaktformulars

1.2.6 Anwendung und Nutzung von Formularen – Bestellungen

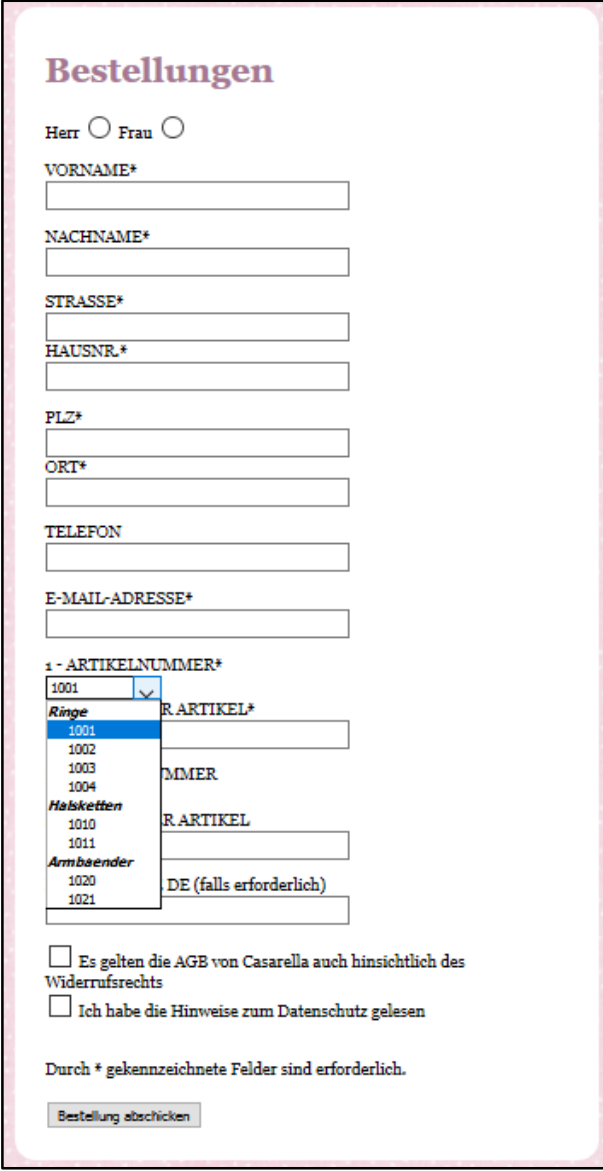
Herr Huber: „Zudem soll es für die Kunden nun die Möglichkeit geben, Bestellungen direkt über die Web Site von Casarella zu tätigen.

Sie sollen nicht mehr darauf angewiesen sein, Bestellungen telefonisch von Montag bis Freitag zwischen 10:00 Uhr und 16:00 Uhr aufgeben zu müssen.

Über die Web Site können Kunden zu jeder Tageszeit die gewünschten Produkte anfordern.

Dafür wünscht sich Casarella neben den üblichen Eingabefeldern für die persönlichen Daten und die Akzeptanz der AGB noch eine Dropdown-Liste zur Auswahl der Produkte.

Auch dafür habe ich bereits das Bestellformular grafisch gestaltet, welches jedoch noch nicht in HTML umgesetzt ist.“



Bestellungen

Herr Frau

VORNAME*

NACHNAME*

STRASSE*

HAUSNR.*

PLZ*

ORT*

TELEFON

E-MAIL-ADRESSE*

1 - ARTIKELNUMMER*

Ringe	R ARTIKEL*
1001	<input type="text"/>
1002	<input type="text"/>
1003	MMER
1004	<input type="text"/>
Halsketten	R ARTIKEL
1010	<input type="text"/>
1011	<input type="text"/>
Armbänder	DE (falls erforderlich)
1020	<input type="text"/>
1021	<input type="text"/>

Es gelten die AGB von Casarella auch hinsichtlich des Widerrufsrechts

Ich habe die Hinweise zum Datenschutz gelesen

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 5: Grafische Gestaltung des neuen Bestellformulars

1.3 XAMPP Installation

Lin W. Lan: „Um die gewünschten Formulare von Casarella umzusetzen, brauchen wir einen Server, der die Kommunikation mit der Web Site ermöglicht.

Der Server sorgt dafür, dass die Informationen in den Formular-Feldern vom Endgerät des Kunden an die Datenbank des Servers gesendet werden, um die Informationen dort zu speichern.“

Für die Übungen dieser WBT-Serie wird dafür XAMPP verwendet. XAMPP ermöglicht es, einen Server auf dem Computer zu Übungszwecken zu simulieren und mit diesem HTML zu lernen, ohne eine eigene Domäne registrieren und für ein Hosting bezahlen zu müssen.

XAMPP ist ein Installationspaket für verschiedene Betriebssysteme X, das folgende Komponenten enthält:

- den Apache-Web-Server,
- das Datenbankverwaltungssystem MariaDB (eine Abspaltung von MySQL),
- die Skriptsprache Perl und
- PHP für serverseitige Programmierung.

Hinweis: An dieser Stelle im WBT werden Sie durch die Installation von XAMPP geführt.

1.4 Ausblick

Lin W. Lan: „Mit Hilfe von XAMPP werden wir in den nächsten WBT zuerst einige Übungen zum Erlernen der Formular-Erstellung durchführen, bevor Sie im Anschluss die Web Site von Casarella anpassen werden!

Ich hoffe, ich konnte Ihnen zusammen mit meinen Kollegen einen ersten Eindruck vermitteln, welche Anforderungen Casarella an die Änderungen ihrer unternehmenseigenen Web Site stellt.

Durch eine kleine Einführung in das Thema "Formulare" wissen Sie nun, wie wir die Umsetzung der Anforderungen gestalten werden.

Im zweiten WBT werde ich Ihnen erklären, wie Formulare in HTML erstellt werden.

Ich freue mich sehr, dass Sie mich erneut bei einem Auftrag von Casarella unterstützen werden.

Auf eine gute Zusammenarbeit!“

2 Formulare

2.1 Einleitung

Lin W. Lan: “Guten Morgen. Schön, dass Sie da sind.

Im letzten WBT haben wir uns mit dem neuen Auftrag von Casarella vertraut gemacht und gelernt, dass wir die entstandenen Anforderungen mit Formularen umsetzen können.

In diesem WBT werden wir lernen, wie diese Formulare in HTML geschrieben werden, wohin die eingegebenen Informationen aus den Formularen geschickt werden, wie dies geschieht und welche verschiedenen Arten von Eingabefeldern es gibt.

Im Anschluss werden wir noch einige Vorbereitungen für die kommenden Übungsaufgaben des HTML-Teils dieser WBT-Serie treffen.“

2.2 Formulare in HTML

2.2.1 Der HTML5-Tag für Formulare

Um Informationen zum Server zu schicken und vom Server zu bekommen, wird ein Formular benötigt. Für Formulare gibt es in HTML5 den `<form>`-Tag.

`<form>` ist ein HTML-Tag, der Eingabefelder erfasst und an den Server überträgt. Dafür müssen im Formular folgende Angaben gemacht werden: wohin mit den Informationen und wie?

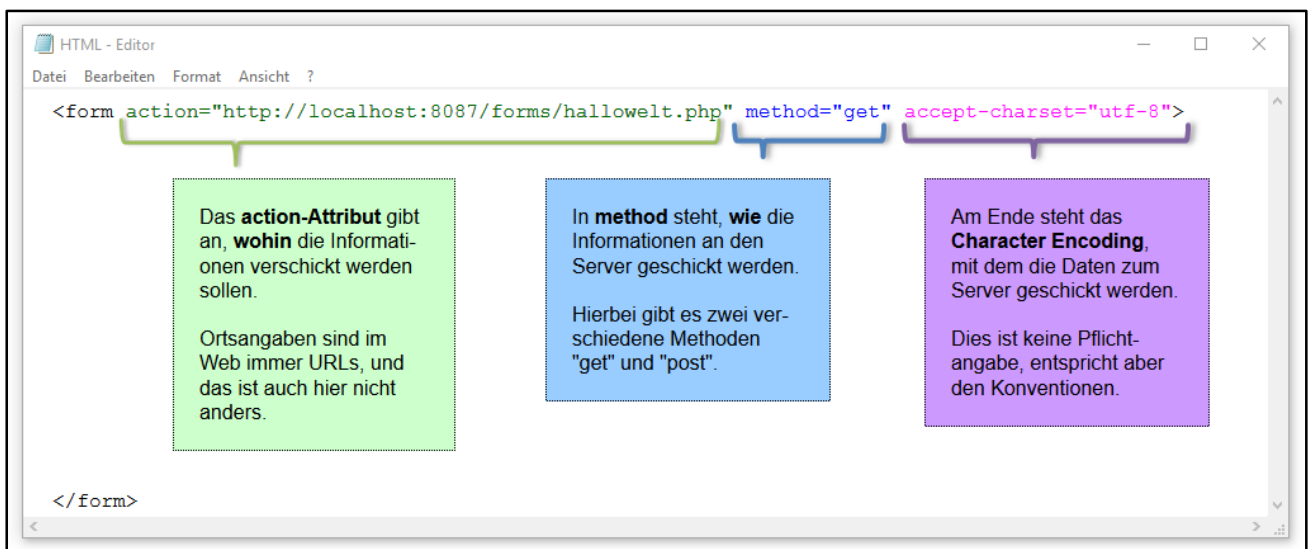


Abb. 6: HTML5-Tag für Formulare

2.2.2 Wohin werden die Daten geschickt? – Das action-Attribut

Das action-Attribut gibt an, **wohin die Informationen** aus den Formularen **verschickt werden** sollen. In den meisten Fällen enthält das Attribut die URL eines Skriptes, welches auf dem Server liegt, an den die Formulareingaben gesendet werden. Das Skript wertet daraufhin die gesendeten Informationen auf dem Server aus und sendet eine Antwort an den Absender des Formulars zurück.

Lin W. Lan: „Für unsere Übungen verwenden wir die URL eines vorgefertigten Skriptes, das im späteren Verlauf dieses WBT noch in das Verzeichnis von XAMPP eingebunden wird.“

Geben wir diese URL, bspw. "hallowelt.php", in das action-Attribut unseres <form>-tag ein, senden wir unsere Informationen an dieses Skript auf unserem XAMPP-Server, welches die Eingaben auswertet und uns Antworten auf die eingegebenen Informationen sendet.“



Abb. 7: Das action-Attribut – HTML

2.2.3 Wie werden die Daten an den Server geschickt? – Die Methoden „get“ und „post“

Mit dem method-Attribut wird die **Versand-Methode** gewählt, mit der die eingegebenen Informationen in den Formularfeldern an den Server geschickt werden.

Die beiden Methoden "get" und "post" unterscheiden sich vor allem darin, wofür sie genutzt werden.

Die Methode "get"

get-Requests sind dafür da, um Web-Seiten zu holen. Typische get-Requests sind Suchanfragen.

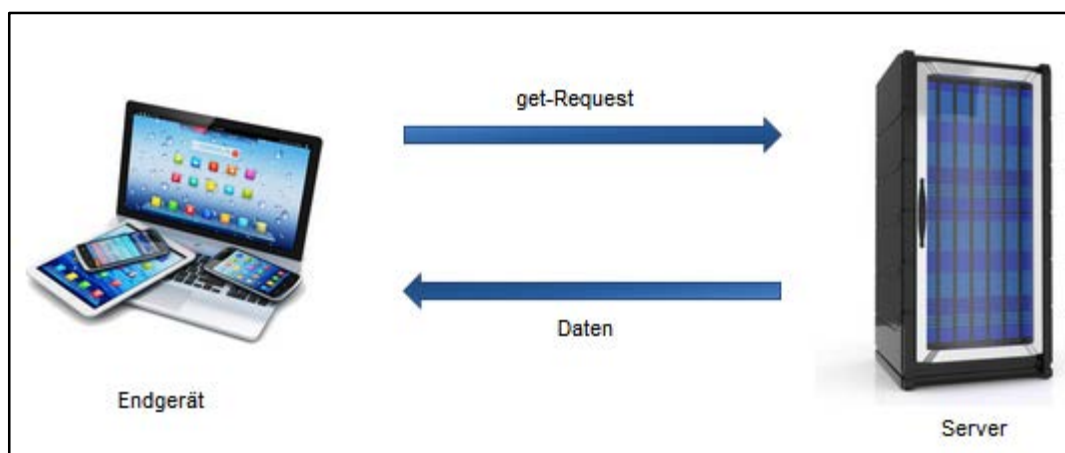


Abb. 8: Die Methode „get“

Die Methode "post"

post-Requests sind dafür da, um Daten an den Server zu senden. Typische post-Requests sind (Pizza-) Bestellungen.

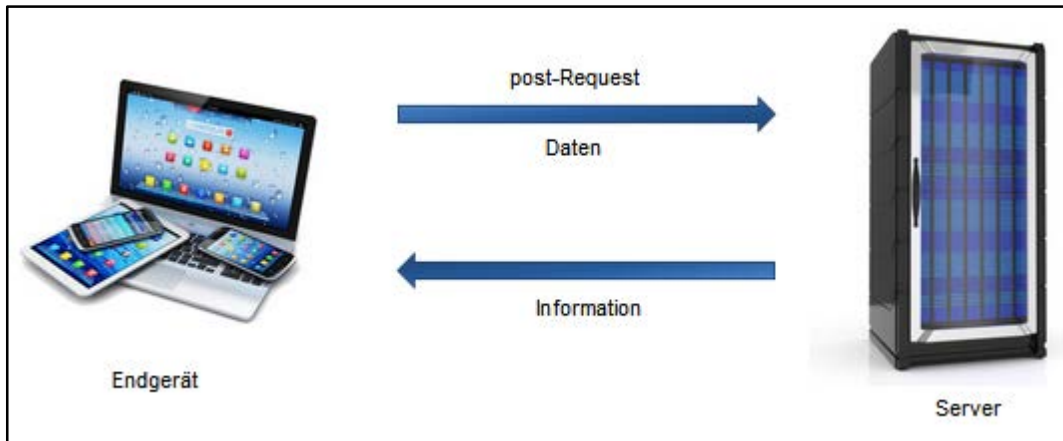


Abb. 9: Die Methode „post“

2.2.4 Die Methoden „get“ und „post“

Über das Attribut "method" wählt man die beiden Methoden "get" und "post" aus. Die Benutzung hängt ganz von dem Verwendungszweck ab.

Die im unteren Beispiel angegebene **Methode "get"** verwendet man, um Web-Seiten "zu holen". Informationen, die mit dieser Methode gesendet werden, sagen dem Server welche Daten man haben möchte und der Server gibt diese dann zurück. Ein klassisches Beispiel dafür sind **Suchanfragen**.

Das Bild zeigt einen Screenshot eines Texteditors mit dem Titel 'HTML - Editor.txt - Editor'. Die Menüleiste enthält 'Datei', 'Bearbeiten', 'Format' und 'Ansicht ?'. Der Hauptbereich zeigt den folgenden HTML-Code:

```
<form action="http://localhost:8087/forms/hallowelt.php" method="get" accept-charset="utf-8">
```

Abb. 10: HTML-Code zur Methode „get“

Bei der **Methode "post"** geht es nicht nur darum, etwas vom Server zu holen. Zwar kommt auch hier ein Ergebnis zurück, aber der Zweck von "post" ist, Daten an den Server zu schicken, die dieser dann speichern soll. Typische "post"-Anfragen sind das Hochladen von Profilbildern in sozialen Netzwerken oder das Abschicken einer Produktbestellung.

Lin W. Lan: „Es ist nicht immer ganz eindeutig, welche Methode die Richtige ist. Als Faustregel können wir uns merken: Wenn man die Anfrage an den Server wiederholen kann, ohne dass ungewollte Nebenwirkungen eintreten, ist "get" richtig, ansonsten "post".“

Eine Suchanfrage 50-mal durchzuführen, hat keine Folgen. Eine Produktbestellung 50-mal abzuschicken, hätte eindeutige Folgen.

2.3 Der <input>-Tag

2.3.1 Was wird an den Server geschickt? – Der <input>-Tag

Herr Neumann: „Von Lin W. Lan haben Sie auf den letzten Seiten bereits erfahren, wohin die Daten aus den Eingabefeldern der Formulare geschickt werden und wie die Daten an den Server gesendet werden.“

Nun bleibt uns noch das Wichtigste zu definieren: **was** wir eigentlich verschicken wollen. Um Inhalte über Formulare an den Server zu schicken, wird also eine **Eingabe des Nutzers** benötigt. Der dazugehörige HTML5-Tag sieht wie folgt aus:“

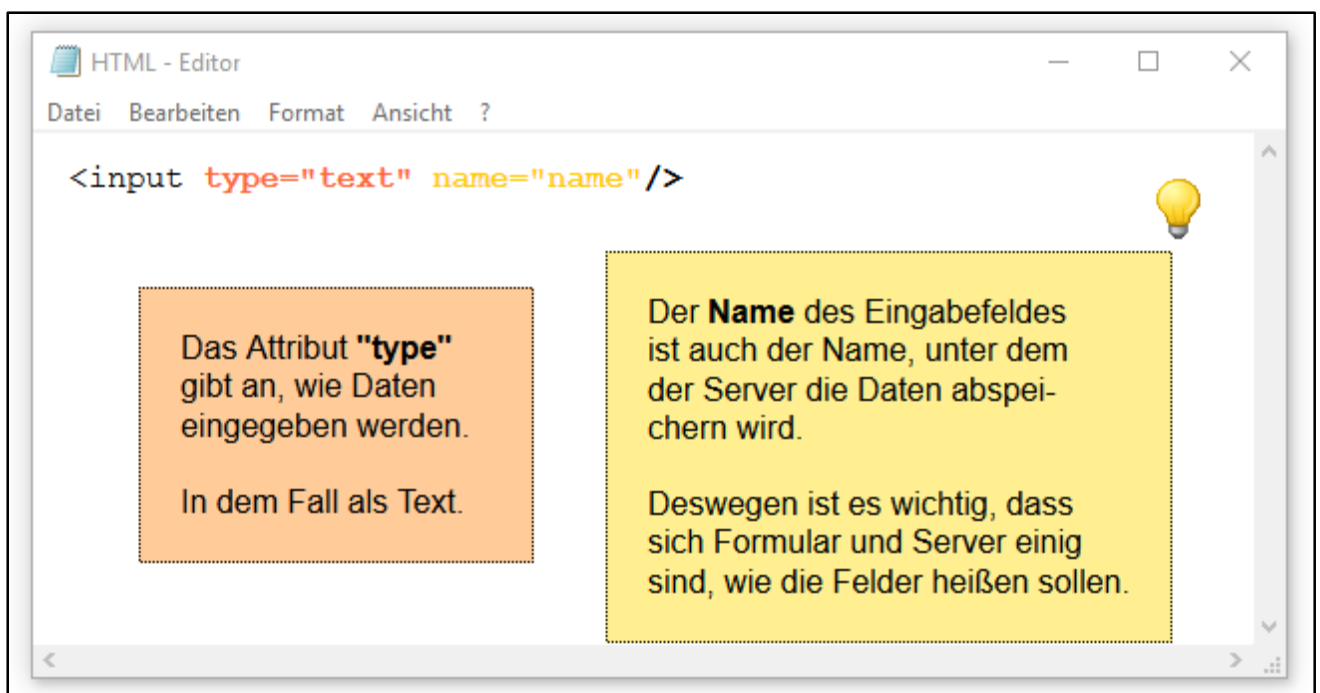


Abb. 11: Der <input>-Tag

2.3.2 Eingabemöglichkeiten mit dem <input>-Tag

Der <input>-Tag hat neben dem Typ "text" noch verschiedene andere Alternativen und deckt somit fast alle Arten von Eingaben ab. Die gängigsten Typen sind:

- **type="text"** - Ein Eingabefeld vom Typ "text" ist für die Eingabe kurzer einzeiliger Texte vorgesehen.
- **type="number"** - Ein Eingabefeld mit dem type="number" beschränkt ein Eingabefeld nur auf Zahleneingaben.
- **type="email"** - Mit type="email" können Eingabefelder erstellt werden, die als Inhalt eine E-Mail-Adresse benötigen. Bei mobilen Geräten (iOS, Android) erscheint bei der Eingabe das @-Symbol automatisch im Tastenfeld.

- **type="date"** - Mit `input type="date"` können Sie Datumseingaben abfragen.
- **type="file"** - Mithilfe des Formularelements "file" kann der Anwender eine Datei von seinem lokalen Rechner zusammen mit dem Formular übertragen. Dieses Element funktioniert nur mit der POST-Methode und nicht mit GET.
- **type="hidden"** - Sie können Felder in einem Formular definieren, die dem Anwender nicht angezeigt werden. Versteckte Felder können Daten enthalten. Beim Absenden des Formulars werden die Daten versteckter Felder mit übertragen. Auf diese Weise können Sie beispielsweise zusätzliche Informationen an das verarbeitende Script übergeben.

Versteckte Felder werden beispielsweise dafür eingesetzt, schon vorhandene Daten in ein zweites Formular zu übernehmen. Wenn beispielsweise im ersten Formular Name und Anschrift eingetragen wurden und in einem zweiten Formular nochmals nach Name und Anschrift gefragt wird, wäre es mühsam, wenn beides erneut eingeben werden müsste.

2.3.3 Attribute des `<input>`-Tag – Teil 1

Herr Huber: „Formulare können zudem im `<input>`-Tag mit Hilfe von Attributen gestaltet werden. Dadurch ist es möglich, typische Anforderungen an Formularfelder zu lösen. Als Beispiel habe ich Ihnen den `<input>`-Tag einer E-Mail-Adresse mitgebracht, der einige solcher Attribute enthält.“



```
HTML - Editor
Datei Bearbeiten Format Ansicht ?
E-MAIL-ADRESSE*<br/>
<input type="email" name="E-Mail-Adresse" maxlength="60"
size="40" placeholder="beispiel@123.de" required/>
```

Abb. 12: Attribute im `<input>`-Tag – HTML

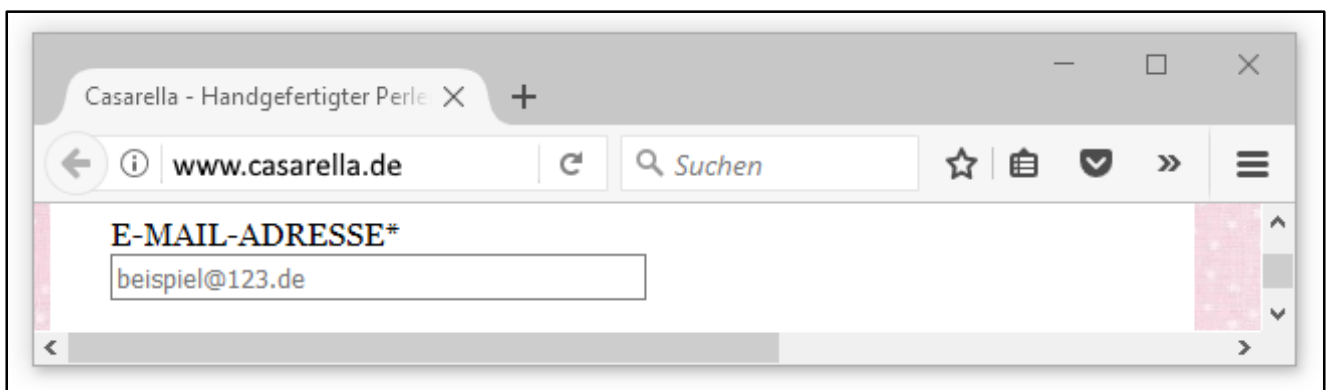


Abb. 13: Attribute im `<input>`-Tag – Browserdarstellung

- **maxlength:** Um die Anzahl der einzugebenden Zeichen zu beschränken, benutzt man das Attribut "maxlength".
- **placeholder:** Das Attribut "placeholder" sorgt dafür, dass ein gewünschter Placeholder-Wert (hier: beispiel@123.de) im Eingabefeld des Formulars in Grau angezeigt wird. Er verschwindet aber, sobald man etwas anderes eintippt und wird auch nicht an den Server übermittelt. Dies dient als zusätzlicher Hinweis, welche Daten erwartet werden.
- **required:** Durch die Angabe von "required" kann eine Eingabe erzwungen werden. Ein vorzeitiges Absenden des Formulars erzeugt eine browser-eigene Fehlermeldung und das noch auszufüllende Formularfeld wird hervorgehoben. So wird sichergestellt, dass die gewünschte Angabe im Formular auf jeden Fall getätigt wird.

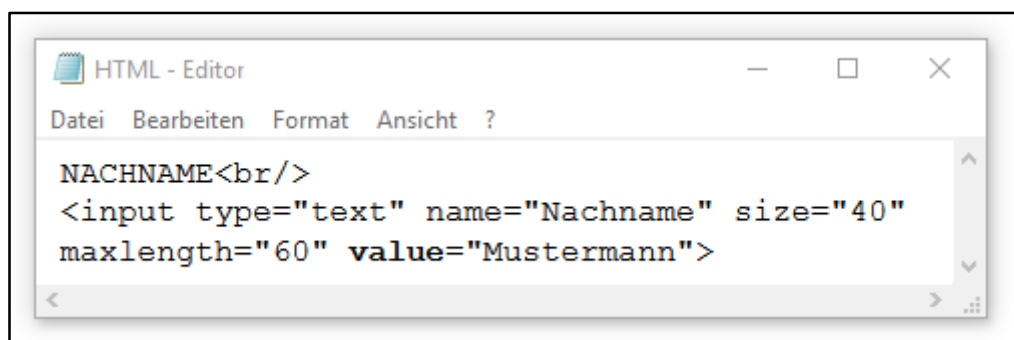
2.3.4 Attribute des <input>-Tag – Teil 2

Neben den bereits vorgestellten Attributen des <input>-Tag, gibt es noch zwei weitere, die von besonderer Bedeutung sind: „value“ und „password“.

- **value:** Mit dem Attribut "value" können Eingabefelder mit einem Inhalt vorbelegt werden. Der Inhalt des vorbelegten Feldes kann vom Nutzer des Formulars geändert werden.

Im Gegensatz zu dem Attribut "placeholder", das ebenfalls einen gewünschten Wert in dem Eingabefeld vorgibt, wird der Wert von "value" an den Server übertragen.

- **password:** In einem Eingabefeld können Sie mit type="password" Geheimnummern, Passwörtern usw. eingeben. Die eingegebenen Zeichen werden dabei durch Platzhalter (Punkte) dargestellt, sodass Personen im Raum des Anwenders nicht zufällig das eingegebene Passwort mitlesen können.



```
HTML - Editor
Datei Bearbeiten Format Ansicht ?
NACHNAME<br/>
<input type="text" name="Nachname" size="40"
maxlength="60" value="Mustermann">
```

Abb. 14: value – HTML

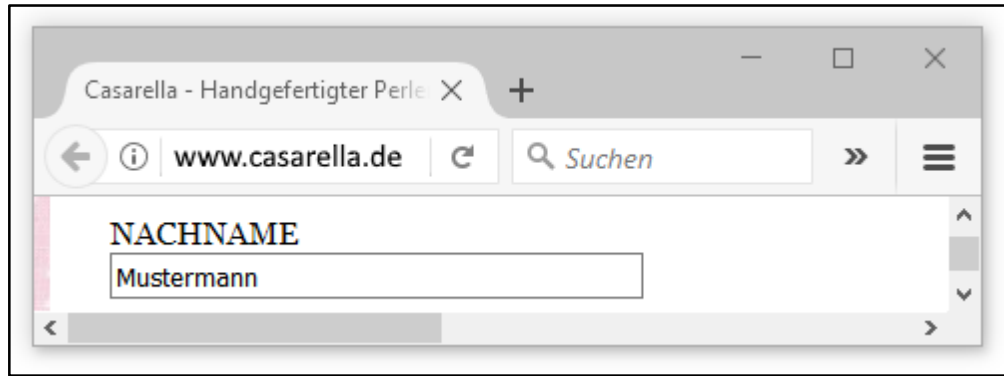


Abb. 15: value – Browserdarstellung

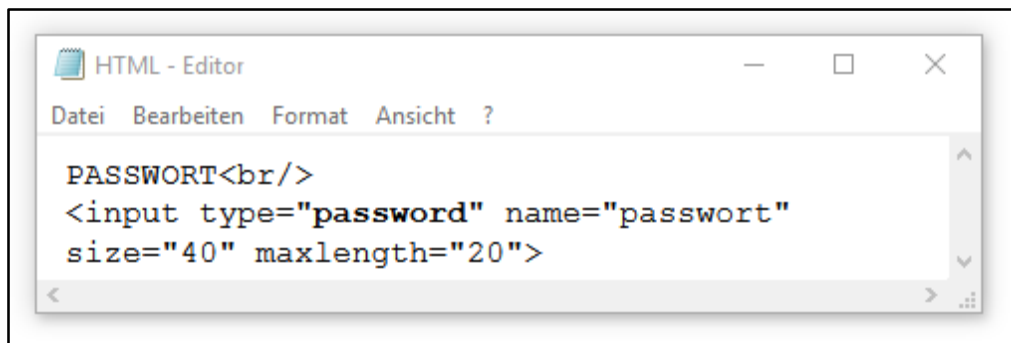


Abb. 16: password – HTML

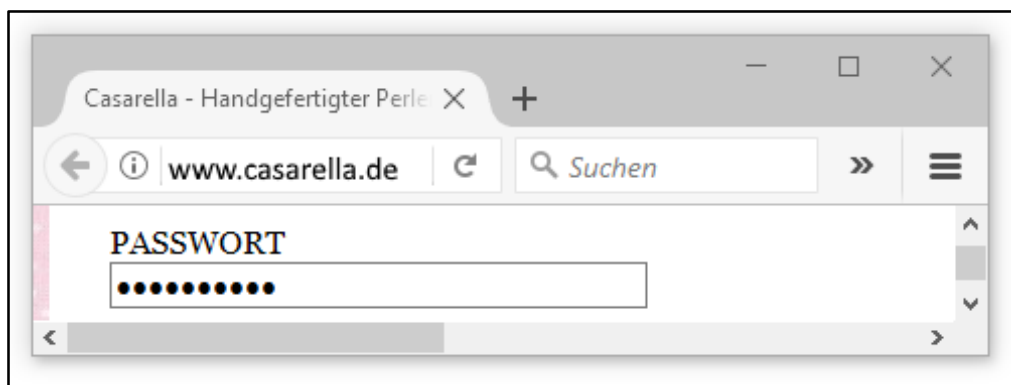
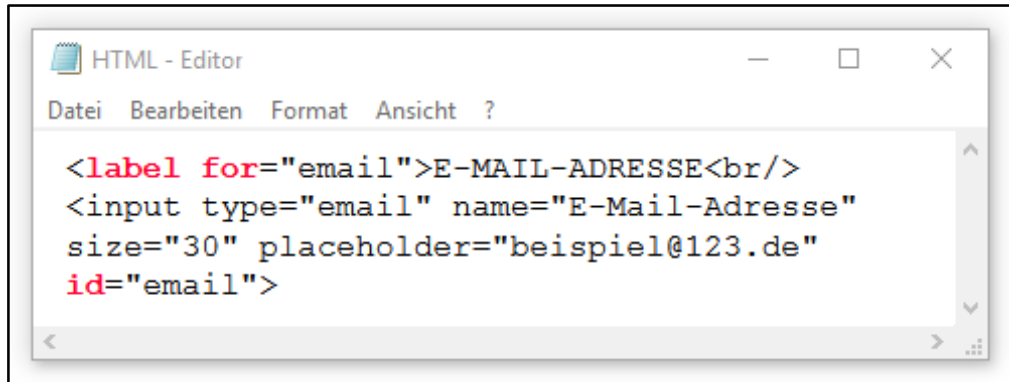


Abb. 17: password – Browserdarstellung

2.3.5 Der <label>-Tag

Herr Huber: „Um Formular-Eingabefelder deutlicher mit der dazugehörigen Beschriftung des Input-Feldes zu verbinden, gibt es den **<label>-Tag**. Die Verbindung kommt über das **for-Attribut** des label-Tags und das **id-Attribut** im Eingabefeld zustande, die denselben Wert aufweisen müssen.“



```

HTML - Editor
Datei Bearbeiten Format Ansicht ?

<label for="email">E-MAIL-ADRESSE<br/>
<input type="email" name="E-Mail-Adresse"
size="30" placeholder="beispiel@123.de"
id="email">

```

Abb. 18: Der <label>-Tag – HTML

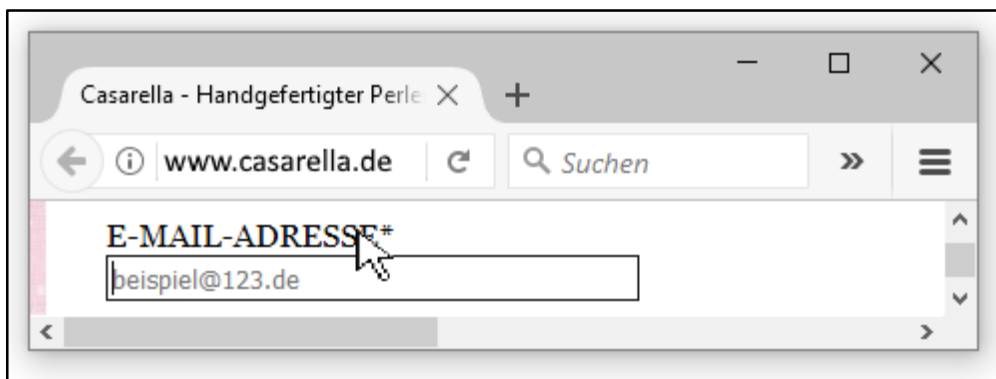


Abb. 19: Der <label>-Tag – Browserdarstellung

Ein guter Grund für das <label>-Tag ist die Barrierefreiheit: Jeder <label>-Tag für ein Eingabefeld verbessert sofort die Benutzeroberfläche für den Benutzer: Es vergrößert den klickbaren Bereich. Ein Klick auf die Beschriftung »E-Mail-Adresse« des Eingabefeldes reicht schon aus, um den Cursor in das Eingabefeld zu versetzen. Der <label>-Tag hat zudem noch folgende Eigenschaften:

- Jedes Eingabefeld kann **nur ein <label>-Tag** haben
- Wenn der Text für die Beschreibung des Eingabefeldes in ein label-Tag gesetzt wird, entstehen keine sichtbaren Änderungen. **Ohne CSS-Stile hat das <label>-Tag keinen Einfluss auf die Gestaltung**
- Wenn das **Formularfeld innerhalb des <label>-Tags** gesetzt wird, werden das for- und das id-Attribut nicht gebraucht, da Eingabefeld und Label direkt miteinander verknüpft sind.

2.3.6 Der Absende-Button

Herr Neumann: „Auf den letzten Seiten haben Sie bereits die Grundlagen zur Formular-Erstellung mit HTML gelernt. Damit Sie ein solches Formular auch an den Server übermitteln können, benötigt es noch etwas, das die Informationen aus den Eingabefeldern abschickt: einen **Absende-Button!**“

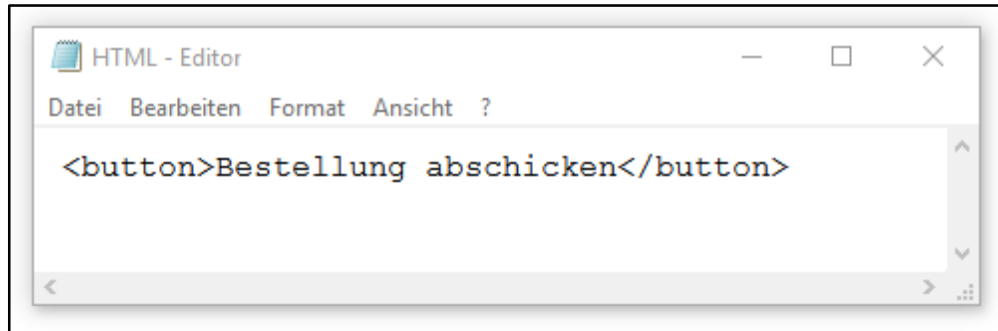


Abb. 20: Der Absende-Button – HTML

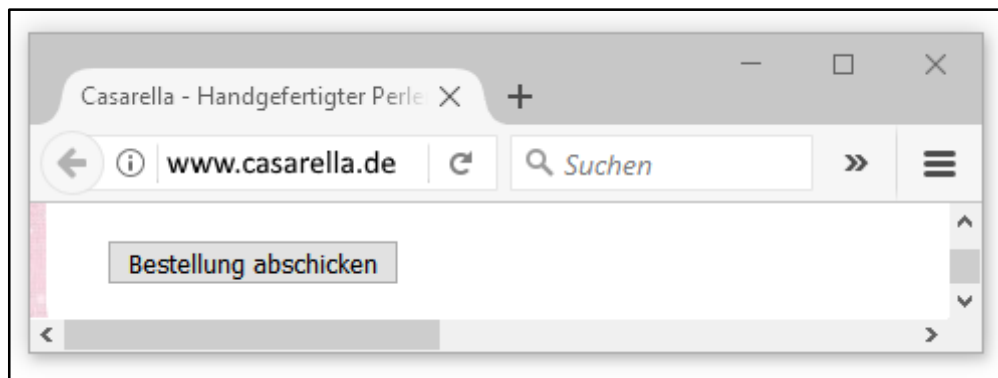


Abb. 21: Der Absende-Button – Browserdarstellung

Im Tag-Body von `<button>` steht die Beschriftung, die auf dem Button angezeigt werden soll. Wenn nichts anderes angegeben wird, schickt `<button>` das umgebende Formular ab. Man kann aber auch mit `<button type="reset">` einen Button anbieten, der das Formular komplett leert.

2.4 Übung

2.4.1 Das Gelernte anwenden

Lin W. Lan: „Nachdem Sie von meinen Kollegen und mir alles Notwendige gelernt haben, um selbst Formulare erstellen zu können, möchten wir nun gerne mit Ihnen testen, wie gut dies schon klappt!

Dafür haben Herr Neumann, Herr Huber und ich uns zusammengesetzt und Ihnen eine Übungsaufgabe erstellt.

Doch bevor Sie mit der Übung beginnen, müssen Sie noch eine kleine Vorbereitung für die kommenden Übungsaufgaben der WBT-Serie treffen.

Legen wir los!“

2.4.2 Vorbereitung für die Übungsaufgaben des HTML-Teils

Um die Übungsaufgaben dieser WBT-Serie durchführen zu können, haben Sie bereits im letzten WBT einen Server auf Ihrem Computer installiert, der zum Lernen alles kann, was Sie brauchen: den XAMPP-Server.

Im XAMPP-Verzeichnis auf Ihrem Computer gibt es den Ordner "htdocs". Alles, was in diesem Ordner liegt, ist durch den Web-Server Apache zugänglich. Legen Sie dort bitte den vorgefertigten und zum Herunterladen bereitstehenden Ordner "forms" für die Formulare ab. In diesem Ordner befindet sich das Script "bestellung.php" und "hallowelt.php".

Auf der Seite "Wohin werden die Daten geschickt? – Das action-Attribut" haben wir bereits gelernt, dass wir ein Script benötigen, welches auf dem Server die Formulareingaben auswertet.

Adressieren wir in unseren Übungen die Eingaben der Formulare mit Hilfe des action-Attributs an dieses Script, erhalten wir Antworten auf die eingegebenen Informationen.

Hinweis: An dieser Stelle im WBT finden Sie den „forms“-Ordner zum Download und eine dazugehörige Anleitung.

Wichtig: Fügen Sie den forms-Ordner nicht dem XAMPP-Verzeichnis hinzu, erhalten Sie nach dem Absenden der Formulare keine Antwort vom Server.

2.4.3 Übungsaufgabe

Bitte erstellen Sie anhand der zum Download bereitstehenden Aufgabenstellung nebenstehendes Formular.

Hinweis: An dieser Stelle im WBT finden Sie ein zum Download bereitstehendes Formular, welches die Aufgabenstellung sowie detaillierte Lösungen enthält.



Bestellung

ANREDE

VORNAME

NACHNAME*

ADRESSE*

PLZ*

WOHNORT*

E-MAIL-ADRESSE*

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 22: Bestellformular

2.5 Ausblick

Herzlichen Glückwunsch, Sie haben in diesem WBT gelernt,

- wie Formulare in HTML geschrieben werden,
- wohin die eingegebenen Informationen aus den Formularen geschickt werden,
- wie dies geschieht,
- welche verschiedenen Arten von Eingabefeldern es gibt,
- wie Eingabefelder mit Hilfe von Attributen gestaltet werden können und
- haben mit diesem Wissen soeben Ihr erstes Formular erstellt!

Im nächsten WBT werden wir zwei weitere Eingabefelder-Typen "Radio-Buttons" und "Checkboxes" kennenlernen.

Ich freue mich auf Sie!“

3 Radio-Buttons und Checkboxes

3.1 Einleitung

Lin W. Lan: „Hallo, willkommen zurück! Nachdem Sie im letzten WBT bereits das grundlegende Wissen zu Formularen erlangt haben, möchte ich Ihnen mit der Unterstützung meiner Kollegen nun gerne zwei weitere Eingabetypen von Formularfeldern näherbringen:

- die Radio-Buttons und
- Checkboxes.

Wir werden Ihnen anhand dieser input-Typen zeigen, dass Formulare mehr können, als nur Texte zu übermitteln und Ihnen erläutern, bei welchen Formularfeldern so etwas sinnvoll ist.

Nach Abschluss dieses WBT kennen Sie bereits den Großteil der gängigsten Anzeigetypen und können den ersten Teil der Formular-Anforderungen an die Web Site von Casarella umsetzen!“

3.2 Radio-Buttons

3.2.1 Schwierigkeiten beim Ausfüllen der freien Formularfelder

Herr Neumann: „In der Übungsaufgabe des letzten WBT, in der Sie Ihr erstes Formular erstellt haben, könnte Ihnen aufgefallen sein, dass die Eingabe der Anrede bzw. des Geschlechts in ein Formular-Textfeld problematisch sein kann.

Es gibt eine Vielzahl an Möglichkeiten, wie ein Benutzer bei bspw. einer Produktbestellung sein Geschlecht in ein Formular-Textfeld eingeben kann, wie z. B.:

- Männlich/Weiblich,
- Mann/Frau,
- Herr/Frau,
- m/w,
- etc...

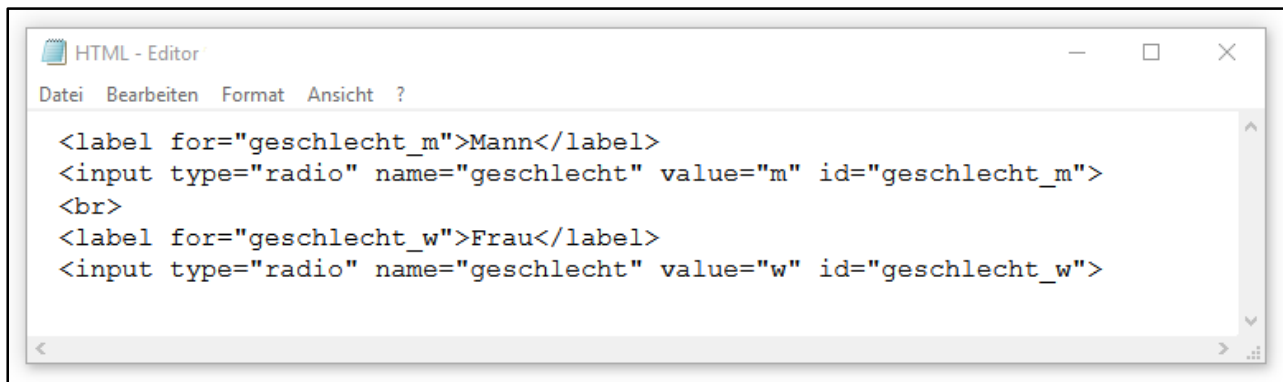
Der Server, der die Formulareingaben auswertet, müsste alle diese Eingaben verstehen können, um die richtige Auswertung vorzunehmen.

Einfacher ist es, dem Benutzer Antwortmöglichkeiten vorzugeben, aus denen dieser eine auswählen kann. So umgeht man das Problem, dass verschiedene Benutzer unterschiedliche Angaben machen.

Dafür bieten sich die sogenannten "**Radio-Buttons**" an.“

3.2.2 Radio-Buttons

Radio-Buttons sind eine Gruppe von beschrifteten Knöpfen, zwischen denen der Anwender entscheiden kann. Es kann immer nur einer der Radio-Buttons ausgewählt sein.



```
<label for="geschlecht_m">Mann</label>
<input type="radio" name="geschlecht" value="m" id="geschlecht_m">
<br>
<label for="geschlecht_w">Frau</label>
<input type="radio" name="geschlecht" value="w" id="geschlecht_w">
```

Abb. 23: Radio-Buttons – HTML

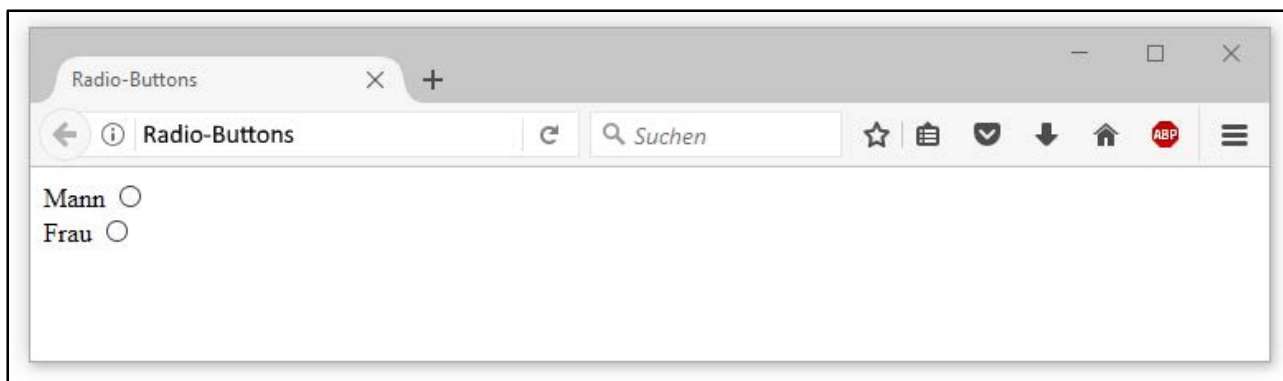


Abb. 24: Radio-Buttons – Browseransicht

Radio-Buttons werden in HTML mit dem input-Typ "**radio**" erstellt. Sie werden demnach ebenso wie die bereits vorgestellten input-Typen "email", "text", "number" etc. durch den **<input>-Tag** eingeleitet.

Bei Radio-Buttons ist es sehr nützlich, das bereits bekannte **<label>** zu verwenden.

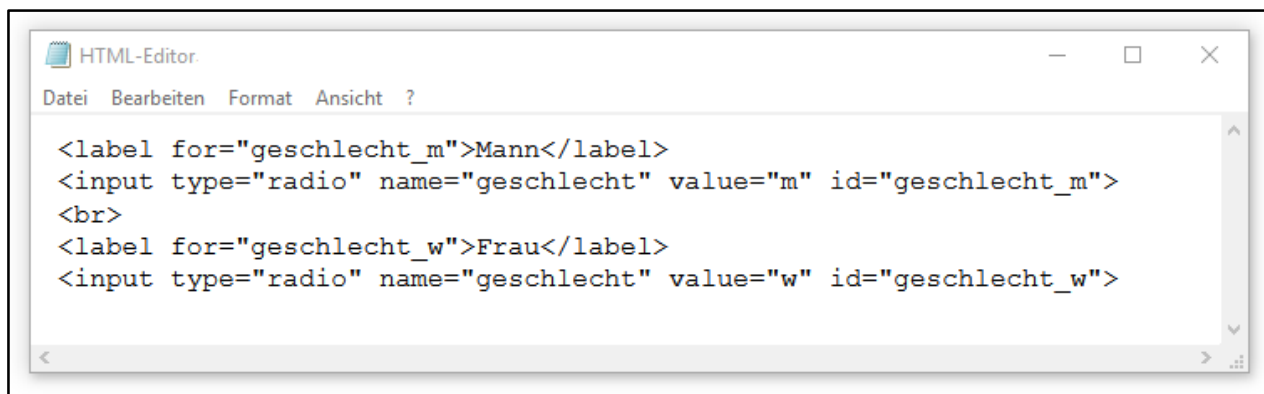
So muss man nicht versuchen, die schwer anwählbaren Radio-Buttons direkt zu treffen, sondern wählt das gewünschte Feld aus, indem man auf den voranstehenden Namen klickt. Besonders relevant ist dies bei der Anzeige auf mobilen Endgeräten und Geräten mit Touch-Funktion.

3.2.3 Der input-Typ „radio“

Ein Eingabefeld vom Typ "radio" ist immer dann nützlich, wenn **nur eine** der vorgegebenen **Auswahlmöglichkeiten** anklickbar sein soll, wie z. B. bei der Geschlechterabfrage.

Damit auch nur eine Möglichkeit auswählbar ist, müssen alle Knöpfe, die zu einer Abfrage gehören (z. B. "Mann" und "Frau"), **einer Gruppe angehören**.

Das wird durch das name-Attribut gewährleistet. Alle zusammengehörigen Radio-Buttons müssen somit **denselben Namen** haben.



```
HTML-Editor
Datei Bearbeiten Format Ansicht ?

<label for="geschlecht_m">Mann</label>
<input type="radio" name="geschlecht" value="m" id="geschlecht_m">
<br>
<label for="geschlecht_w">Frau</label>
<input type="radio" name="geschlecht" value="w" id="geschlecht_w">
```

Abb. 25: Der input-Typ „radio“ – HTML

Bei Radio-Buttons ist zudem vor allem das value-Attribut wichtig, denn beim Absenden des Formulars wird immer der festgelegte Wert, z. B. "Mann", des ausgewählten Radio-Buttons übermittelt.

Das value-Attribut bei Radio-Buttons:

Mit dem Attribut "value" können Eingabefelder mit einem Inhalt vorbelegt werden, welcher beim Absenden des Formulars an den Server übertragen wird.

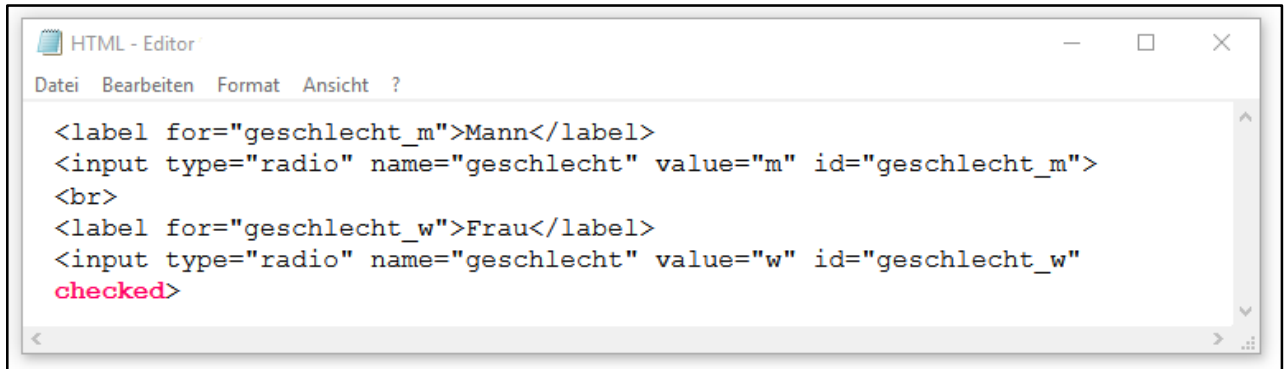
Im Gegensatz zu dem value-Attribut bei Textfeldern, ist der value hier nicht nur eine Vorbelegung, sondern der einzige Wert, den ein Radio-Button hat!

3.2.4 Eine Vorauswahl anbieten – Das Attribut „checked“

Der <input>-Tag bietet neben den bereits erlernten Attributen "maxlength", "placeholder", "required", "value" und "password", noch ein weiteres hilfreiches Attribut namens "**checked**", das **ausschließlich bei Radio-Buttons sowie Checkboxes** angewendet werden kann.

Das Attribut "checked" sorgt dafür, dass Auswahlmöglichkeiten vorselektiert werden. Lädt man eine Web Site wird der Radio-Button, welcher mit dem Attribut "checked" gekennzeichnet ist, vorausgewählt.

Im unteren Beispiel ist dies der Radio-Button mit dem Namen "Frau".



```
<label for="geschlecht_m">Mann</label>
<input type="radio" name="geschlecht" value="m" id="geschlecht_m">
<br>
<label for="geschlecht_w">Frau</label>
<input type="radio" name="geschlecht" value="w" id="geschlecht_w"
checked>
```

Abb. 26: Das Attribut „checked“ – HTML

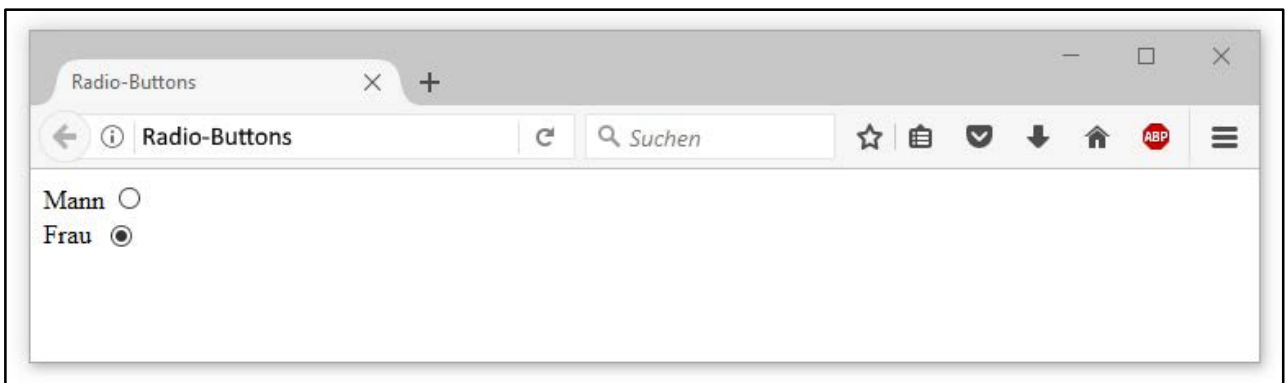


Abb. 27: Das Attribut „checked“ – Browseransicht

3.2.5 Übung zu Radio-Buttons

Lin W. Lan: „Nachdem Sie nun gelernt haben, dass es bei Formulareingabefeldern sinnvoll sein kann, Auswahlmöglichkeiten statt Texteingabefelder vorzugeben, möchte ich mit Ihnen eine kleine Übung durchführen.“

Bitte erstellen Sie folgende Geschlechterauswahl unter der Berücksichtigung der erlernten Attribute "value", "placeholder", "checked" und dem <label>-Tag:

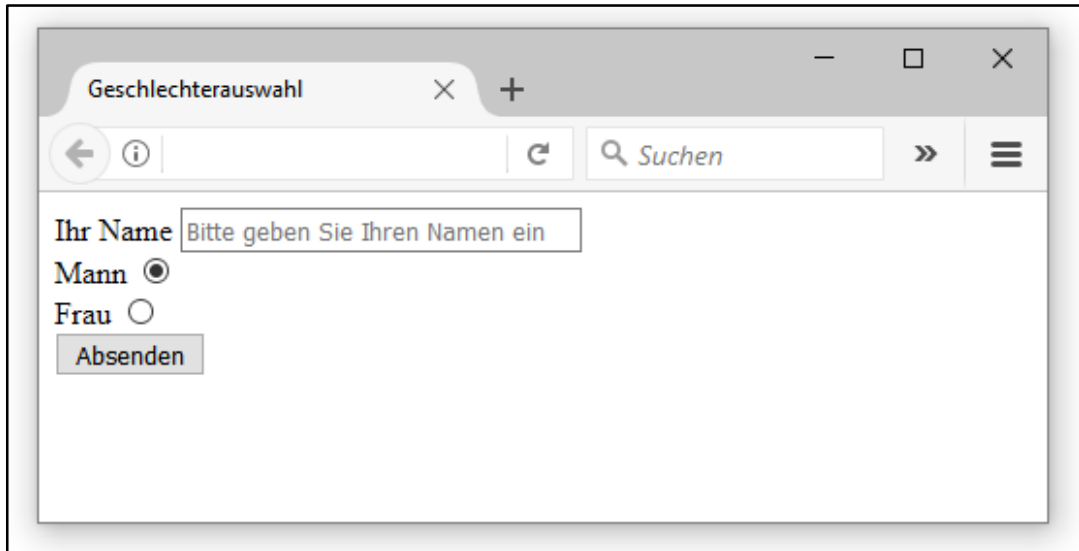


Abb. 28: Übung zu Radio-Buttons

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

3.3 Checkboxes

3.3.1 Checkboxes

Herr Neumann: „Wollen Sie in einem Formularfeld nicht zwischen mehreren Optionen auswählen, sondern nur eine Option an- oder ausschalten, um beispielsweise einen Newsletter zu bestellen, gibt es eine bessere Möglichkeit als die Radio-Buttons: die sogenannte **Checkbox**.“

Wichtiges zu Checkboxes:

- Checkboxes sind ankreuzbare Kontrollfelder
- Anwender können **keine, eine** oder **mehrere** dieser Kontrollfelder auswählen
- Die **Werte** (value) von ausgewählten Checkboxes werden beim Absenden des Formulars mit **übertragen**

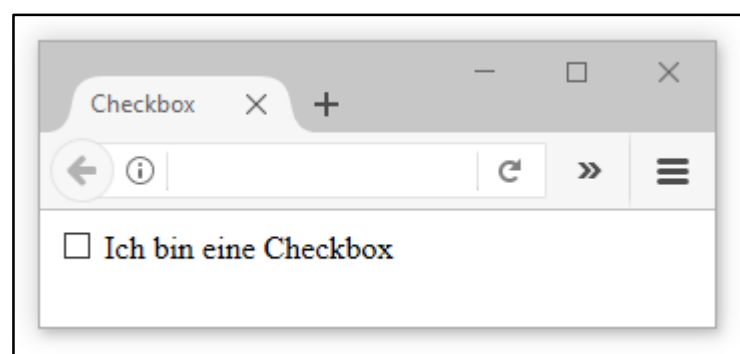


Abb. 29: Checkbox – Browseransicht

3.3.2 Der input-Typ „checkbox“

Im Gegensatz zu Radio-Buttons werden Checkboxes nicht zu Gruppen zusammengefasst, sondern stehen für sich alleine.

Wählt man eine Checkbox an, **ändert dies nichts am Zustand** der anderen Checkboxes, selbst wenn diese denselben Namen haben. Stattdessen schaltet man Checkboxes mit einem Klick an und mit einem weiteren Klick wieder aus.

Einsatzgebiet von Checkboxes:

Da bei Checkboxes mehrere Optionen einer Liste angewählt werden können, dienen sie für:

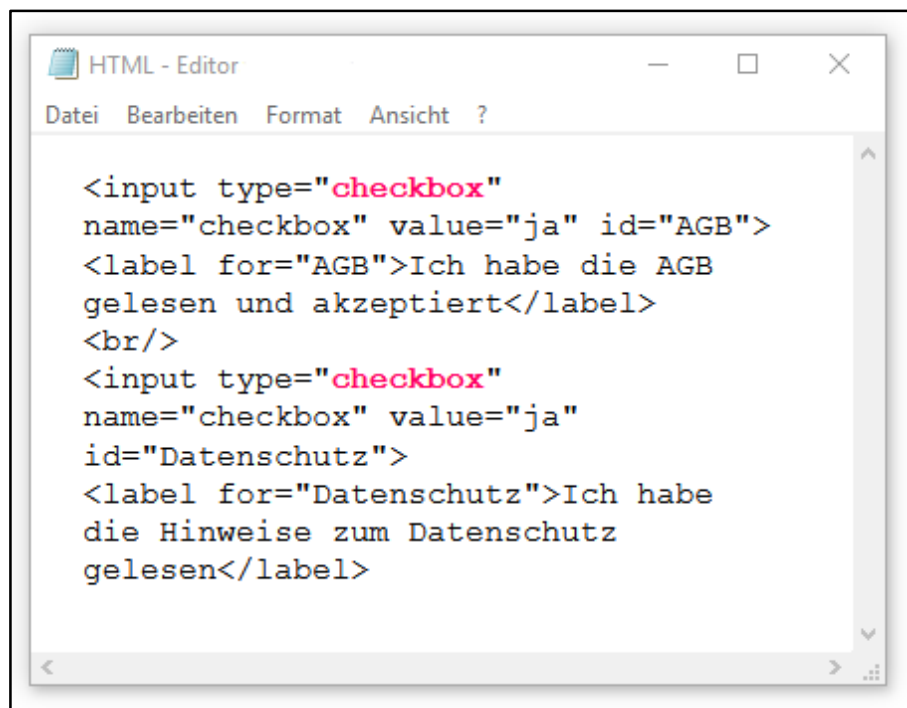
- den Aufbau interaktiver Checklisten und
- Auswahl- oder Fragelisten.

Dadurch können bei Formularen Kontrollfelder, wie

- Hinweise zum Datenschutz und
- der AGB

umgesetzt werden.

Bei Checkboxes macht es ebenfalls Sinn, den bereits bekannten <label>-Tag zu verwenden.



```
<input type="checkbox"
name="checkbox" value="ja" id="AGB">
<label for="AGB">Ich habe die AGB
gelesen und akzeptiert</label>
<br/>
<input type="checkbox"
name="checkbox" value="ja"
id="Datenschutz">
<label for="Datenschutz">Ich habe
die Hinweise zum Datenschutz
gelesen</label>
```

Abb. 30: Der input-Typ „checkbox“ – HTML

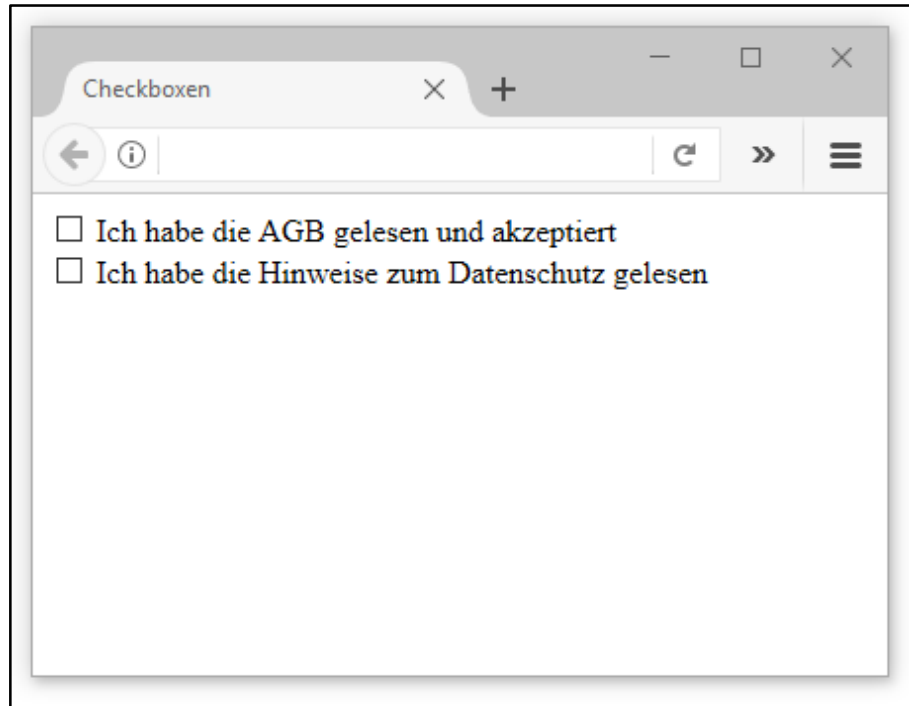


Abb. 31: Der input-Typ „checkbox“ – Browseransicht

3.3.3 Übungen zu Checkboxes

Herr Huber: „Fügen Sie nun bitte dem Formular aus der ersten Übungsaufgabe noch eine **Checkbox** mit dem Namen "mehr Komplimente" hinzu:“

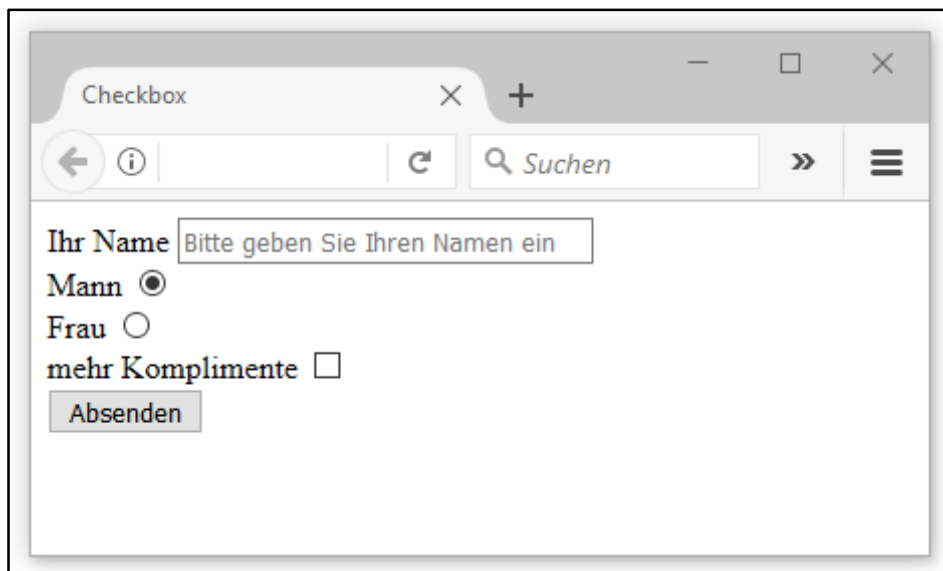


Abb. 32: Übung zu Checkboxes

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

3.4 Zusatz und Übung

3.4.1 Zusatz zu Radio-Buttons und Checkboxes – <fieldset>

Zur Strukturierung von Radio-Buttons und Checkboxes können diese durch einen Rahmen gruppiert werden.

Dafür gibt es das **<fieldset>-Element**:

- Mit dem **fieldset-Element** können Auswahlmöglichkeiten innerhalb von Formularen **gruppiert** werden
- Mit dem **legend-Element** kann zusätzlich eine **Überschrift** für das <fieldset>-Element definiert werden
- Das <fieldset>-Element **umschließt** dabei den <label>- und <input>-Tag der gewünschten Gruppe
- Zweck dieses Elements ist einzig die **optische Gruppierung**

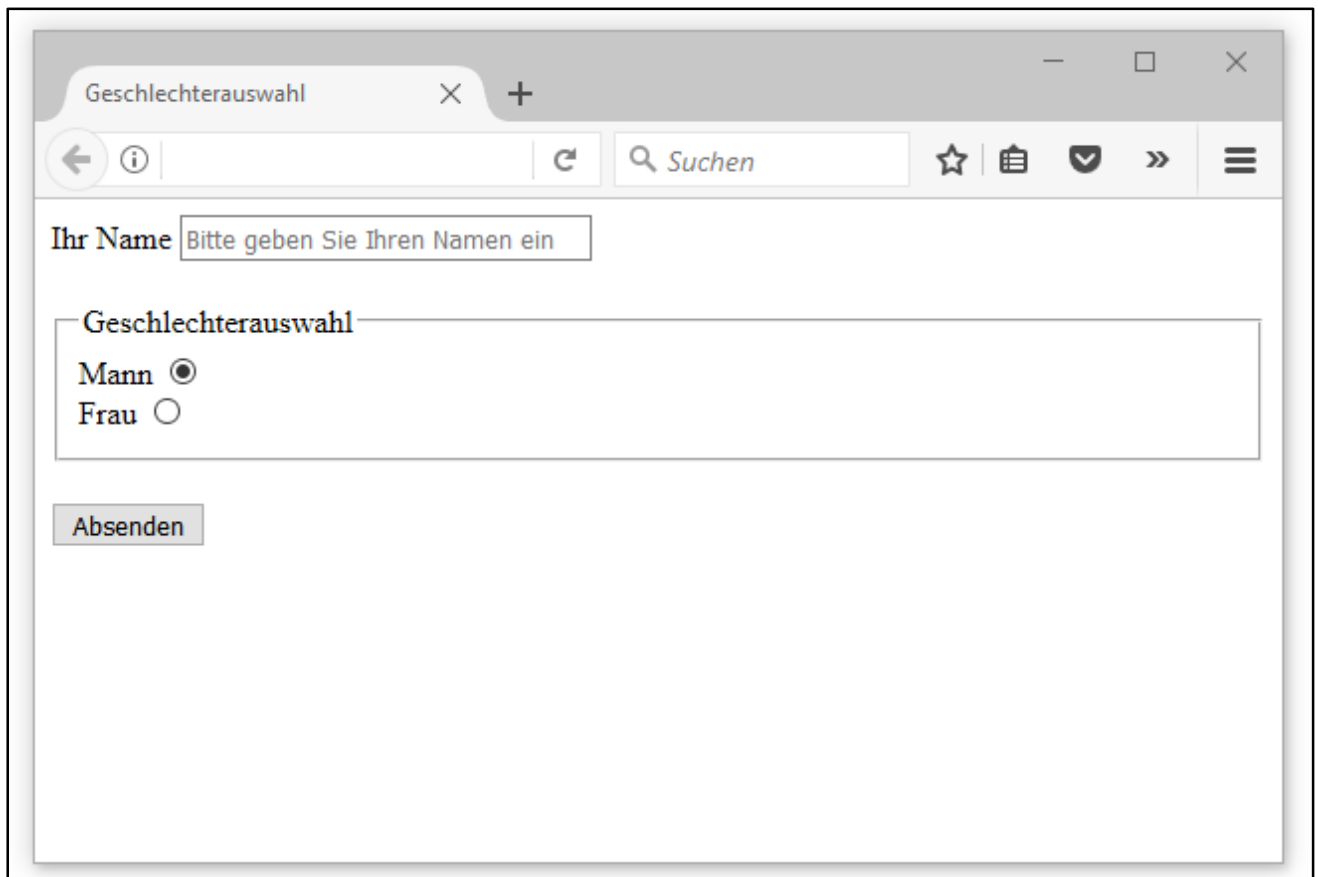
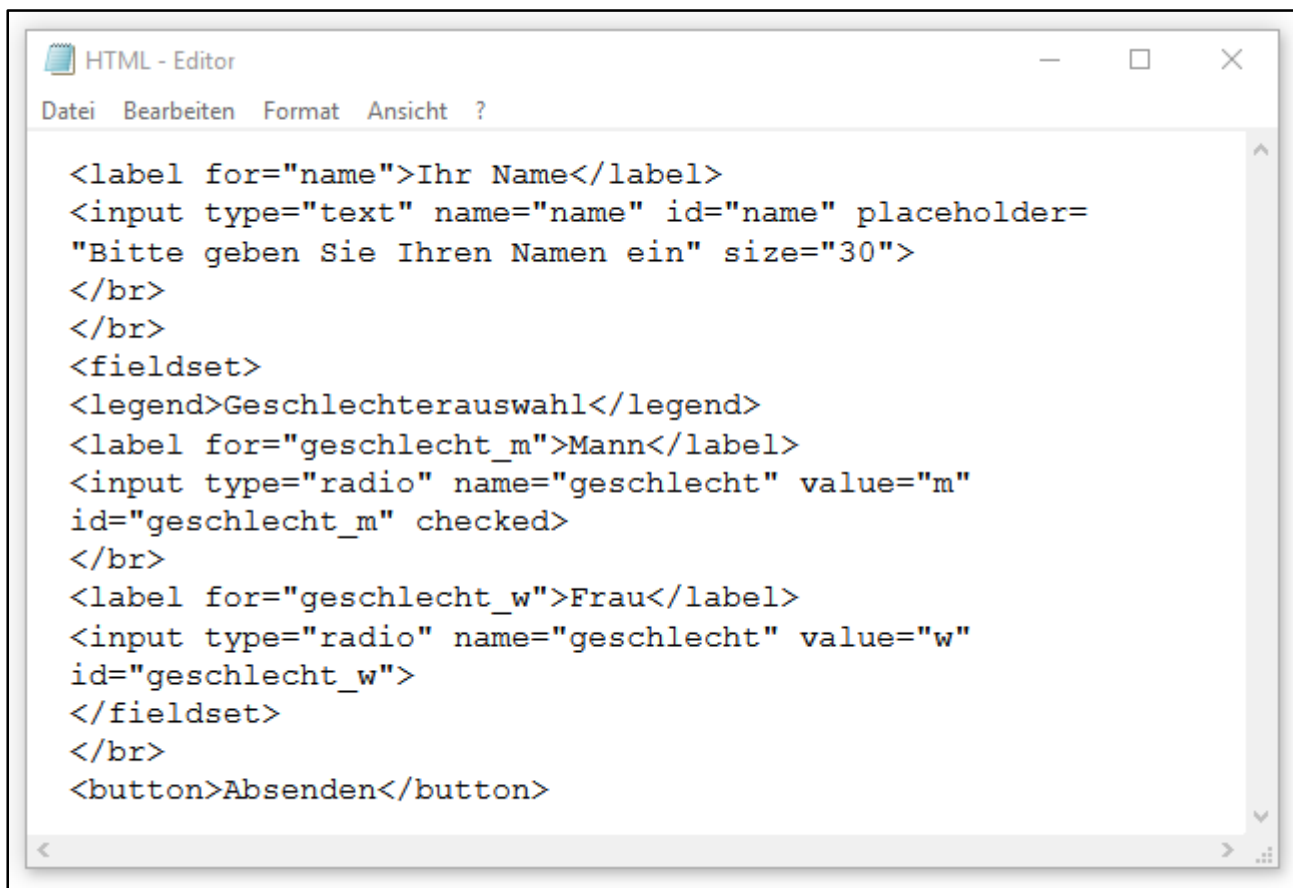


Abb. 33: Das <fieldset>-Element – Browseransicht



```
<label for="name">Ihr Name</label>
<input type="text" name="name" id="name" placeholder=
"Bitte geben Sie Ihren Namen ein" size="30">
</br>
</br>
<fieldset>
<legend>Geschlechterauswahl</legend>
<label for="geschlecht_m">Mann</label>
<input type="radio" name="geschlecht" value="m"
id="geschlecht_m" checked>
</br>
<label for="geschlecht_w">Frau</label>
<input type="radio" name="geschlecht" value="w"
id="geschlecht_w">
</fieldset>
</br>
<button>Absenden</button>
```

Abb. 34: Das `<fieldset>`-Element – HTML

3.4.2 Das Gelernte anwenden

Lin W. Lan: „In den letzten drei WBT haben Sie bereits eine Menge über die Erstellung von Formularen in HTML gelernt.

Sie können nun einfache **Texteingabefelder** sowie **Radio-Buttons** und **Checkboxes** in Formularen erstellen und kennen die wichtigsten Attribute, um typische Anforderungen an Formularfelder zu lösen.

Somit sind Sie bereit, die **ersten Anforderungen** von **Casarella** an deren Web Site **umzusetzen!**

Meine Kollegen und ich freuen uns darauf, auf der nächsten Seite diese Aufgabe mit Ihnen zu beginnen!“

3.4.3 Umsetzung der Formular-Anforderungen – Teil 1

Bitte erstellen Sie nebenstehendes Formular, welches einen ersten Teil der umgesetzten Anforderungen an die Web Site von Casarella zeigt.

Nutzen Sie hierfür den in WBT 01 bereits heruntergeladenen Quellcode der bisherigen Web Site von Casarella und fügen Sie dem Web-Site-Menü einen neuen **Reiter "Bestellungen"** hinzu.



The image shows a web form titled "Bestellungen" (Orders) within a navigation menu. The menu includes "Home", "Schmuck", "Dekoration", "Service", "Bestellungen", and "Impressum & Kontakt". The "Bestellungen" link is circled in purple, and a large purple arrow points upwards from the bottom of the circle towards the form's title. The form itself contains the following elements:

- Gender selection: "Herr Frau
- Required fields (marked with *):
 - VORNAME*
 - NACHNAME*
 - STRASSE*
 - HAUSNR.* (with a dropdown arrow)
 - PLZ*
 - ORT*
 - TELEFON
 - E-MAIL-ADRESSE*
- Optional checkboxes:
 - Es gelten die AGB von Casarella auch hinsichtlich des Widerrufsrechts
 - Ich habe die Hinweise zum Datenschutz gelesen
- Disclaimer: "Durch * gekennzeichnete Felder sind erforderlich."
- Submit button: "Bestellung abschicken"

Abb. 35: Formularanforderungen – Teil 1

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

3.5 Ausblick

Lin W. Lan: „In diesem WBT haben Sie zwei weitere Anzeigetypen von Formularfeldern kennen gelernt:

- die Radio-Buttons und
- Checkboxes.

Im Anschluss daran, haben Sie mit Ihrem bereits erlerntem Wissen die ersten Anforderungen an die Web Site von Casarella umgesetzt. Damit haben Sie in kürzester Zeit schon jede Menge Neues gelernt und angewandt.

Im nächsten WBT werden Sie lernen, wie Sie Listen mit festen Einträgen anbieten, aus der ein Eintrag ausgewählt werden kann.“

4 Select-Tag und Datalist-Tag

4.1 Einleitung

Lin W. Lan: „Hallo! Nachdem Sie im letzten WBT bereits die ersten Anpassungen an der Web Site von Casarella vorgenommen haben, führten Herr Neumann, Herr Huber und ich ein erstes Zwischengespräch mit Casarella.

Casarella ist mit Ihrer bisherigen Arbeit sehr zufrieden und freut sich bereits darauf, die neuen Formulare bald einsetzen zu können.

Damit wir mit der Umsetzung der Anforderungen an die Web Site fortfahren können, lernen Sie heute noch zwei wichtige HTML-Elemente kennen: die **Select-Box** und die **Datenliste**.

Im Anschluss kennen Sie alle HTML-Elemente, die Sie für die Erstellung der Formulare für Casarella benötigen und können den Auftrag abschließen!“

4.2 Select-Tag

4.2.1 Die Select-Box

Herr Huber: „Die **Select-Box**, im allg. Sprachgebrauch als Dropdown-Feld bekannt, ist ein weiteres HTML-Element, das sich Casarella für das neue Formular ihrer Web Site wünscht.

Die Funktionsweise ist nicht anders, als die von Radio-Buttons und Checkboxes. Zur Auswahl stehen mehrere Einträge einer Liste, von denen einer angewählt werden kann.

Dieses Design eignet sich vor allem bei großen Listen, da die Aufzählung der Einträge erst bei Klick auf die Select-Box gezeigt wird.“

1 - ARTIKELNUMMER*
1001

1 - ANZAHL DER ARTIKEL*

2 - ARTIKELNUMMER
1001

2 - ANZAHL DER ARTIKEL

RINGGROESSE DE (falls erforderlich)

Abb. 36: Die Select-Box – Browseransicht

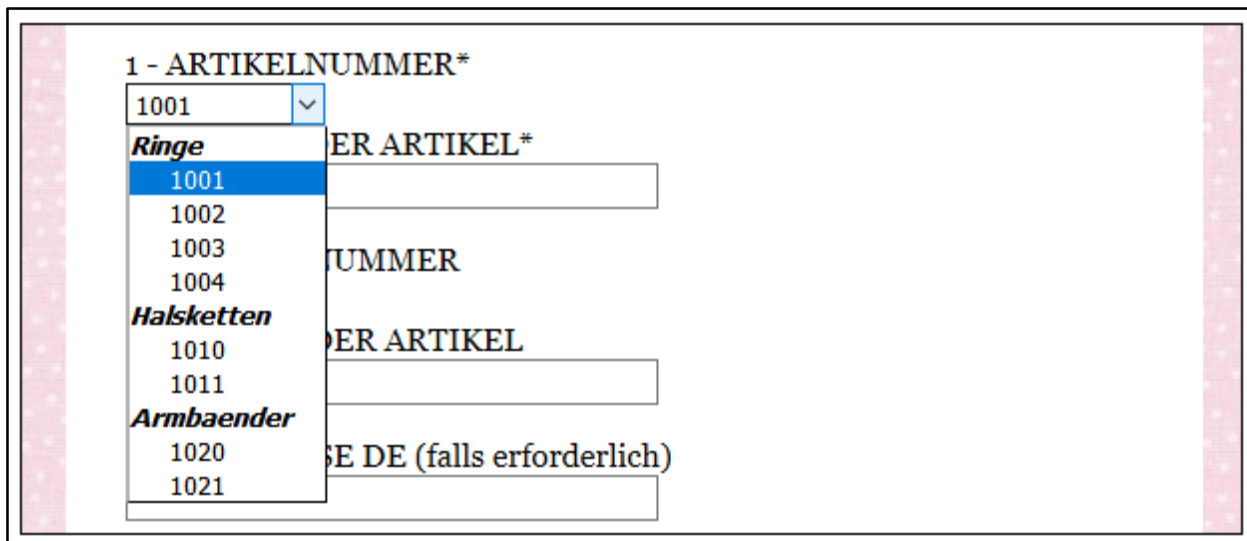


Abb. 37: Klick auf die Select-Box – Browseransicht

4.2.2 Der <select>-Tag

Select-Boxen werden in HTML mit dem **<select>-Tag** eingeleitet. Jede Select-Box sollte mit dem Attribut **"name"** einen internen Bezeichnernamen erhalten, damit ihr Wert von dem Ziel-Server verarbeitet werden kann.

Um Auswahlmöglichkeiten der Select-Box hinzuzufügen, benötigt man den **<option>-Tag**, der in den <select>-Tag verschachtelt wird.

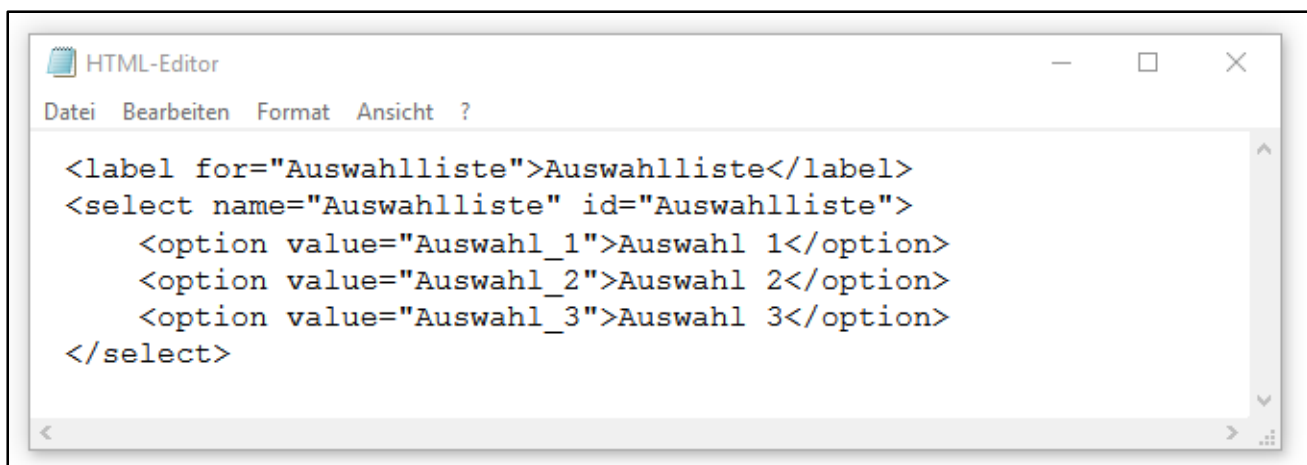


Abb. 38: Der <select>-Tag – HTML

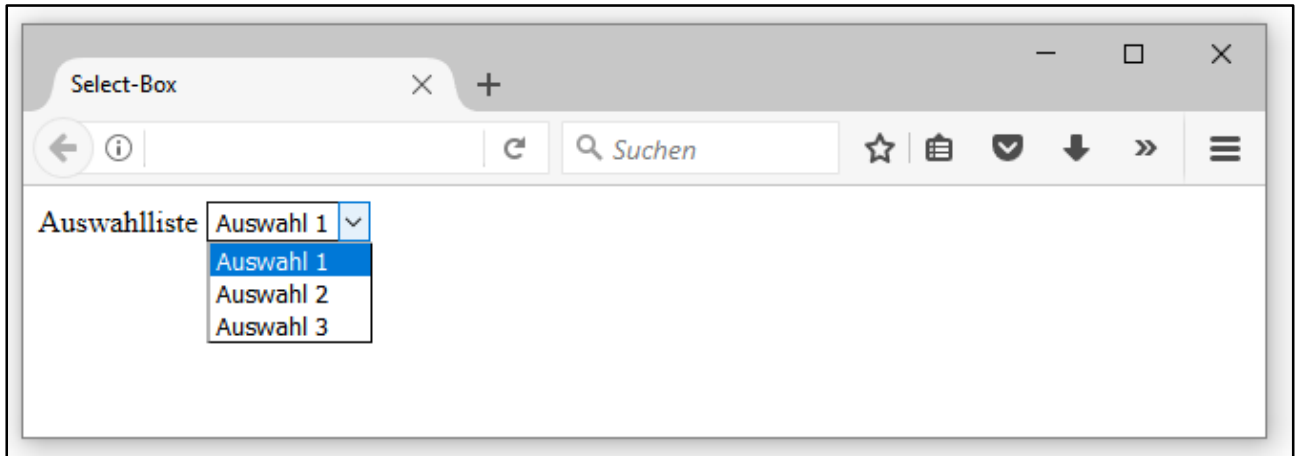


Abb. 39: Der <select>-Tag – Browseransicht

Hinweis: Das Attribut, um eine Option vorauszuwählen, lautet bei Auswahllisten nicht "checked", sondern "selected"!

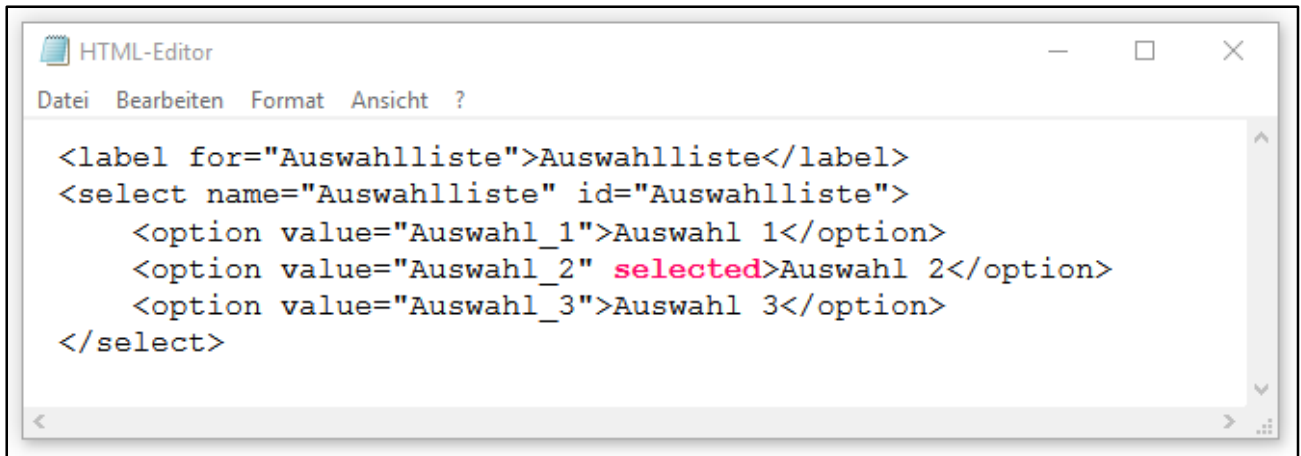


Abb. 40: Vorauswahl beim <select>-Tag – HTML

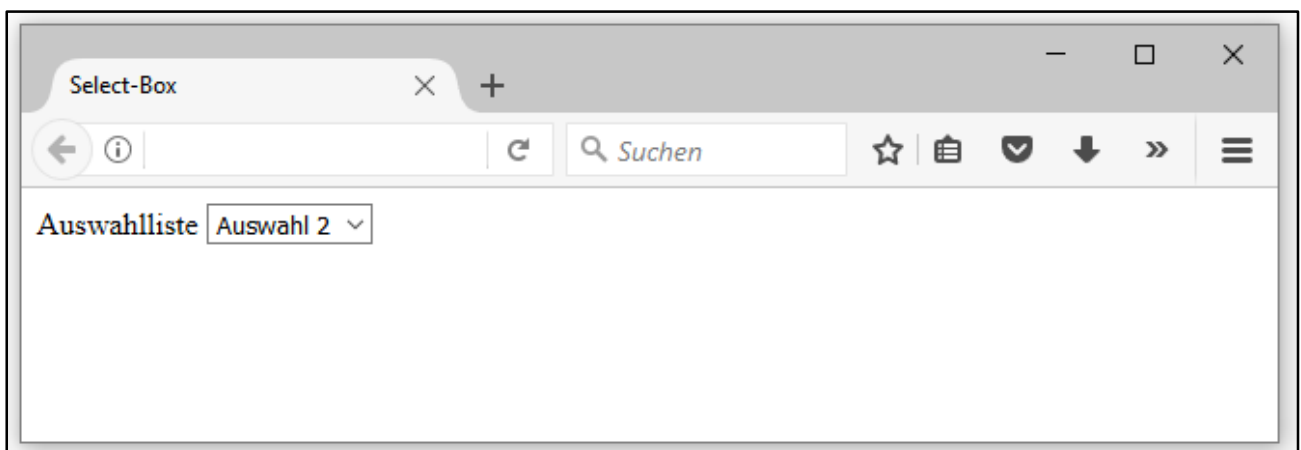


Abb. 41: Vorauswahl beim <select>-Tag – Browseransicht

4.2.3 Der <optgroup>-Tag

Lin W. Lan: „Casarella möchte in der Select-Box zur Auswahl ihrer Produkte, zusätzlich noch eine Gruppierung der Auswahlmöglichkeiten haben. Durch den **<optgroup>-Tag** lassen sich Optionen in einer Liste gruppieren und mit einer Überschrift versehen.

Die Überschriften für jede Gruppe stehen dabei im **<label>-Attribut**.“

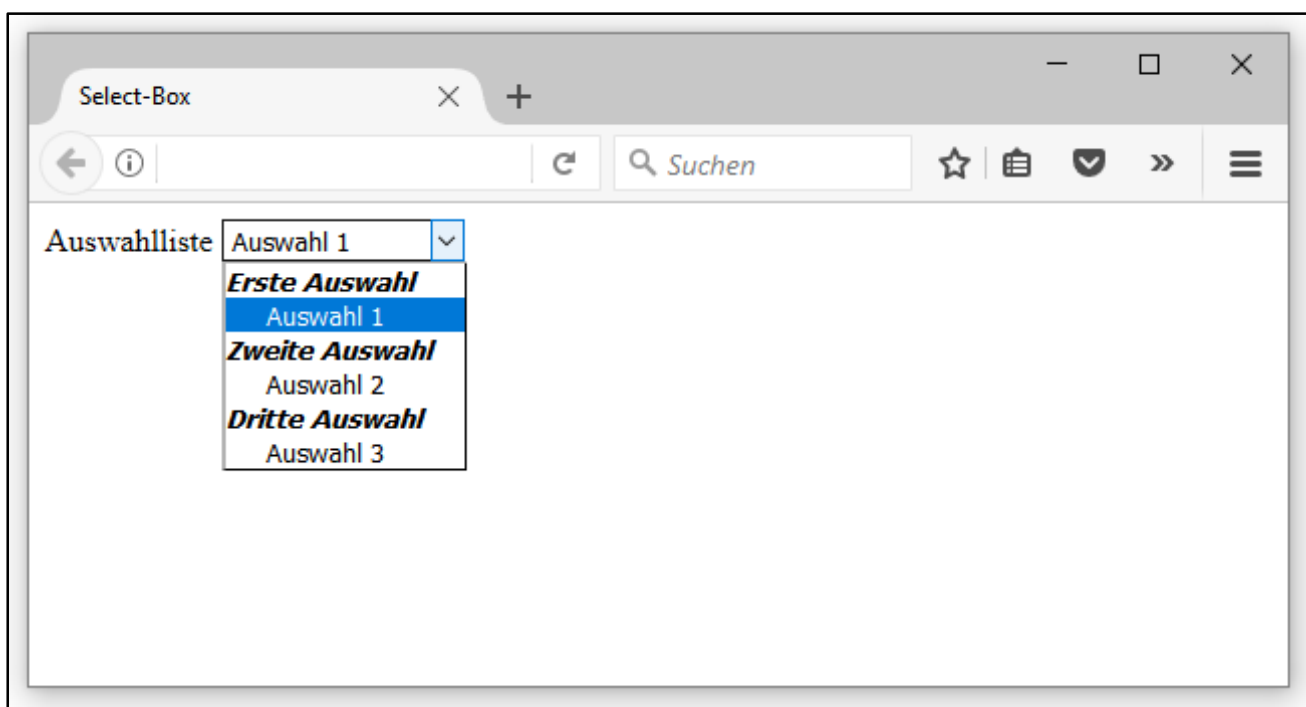


Abb. 42: Der <optgroup>-Tag – Browseransicht

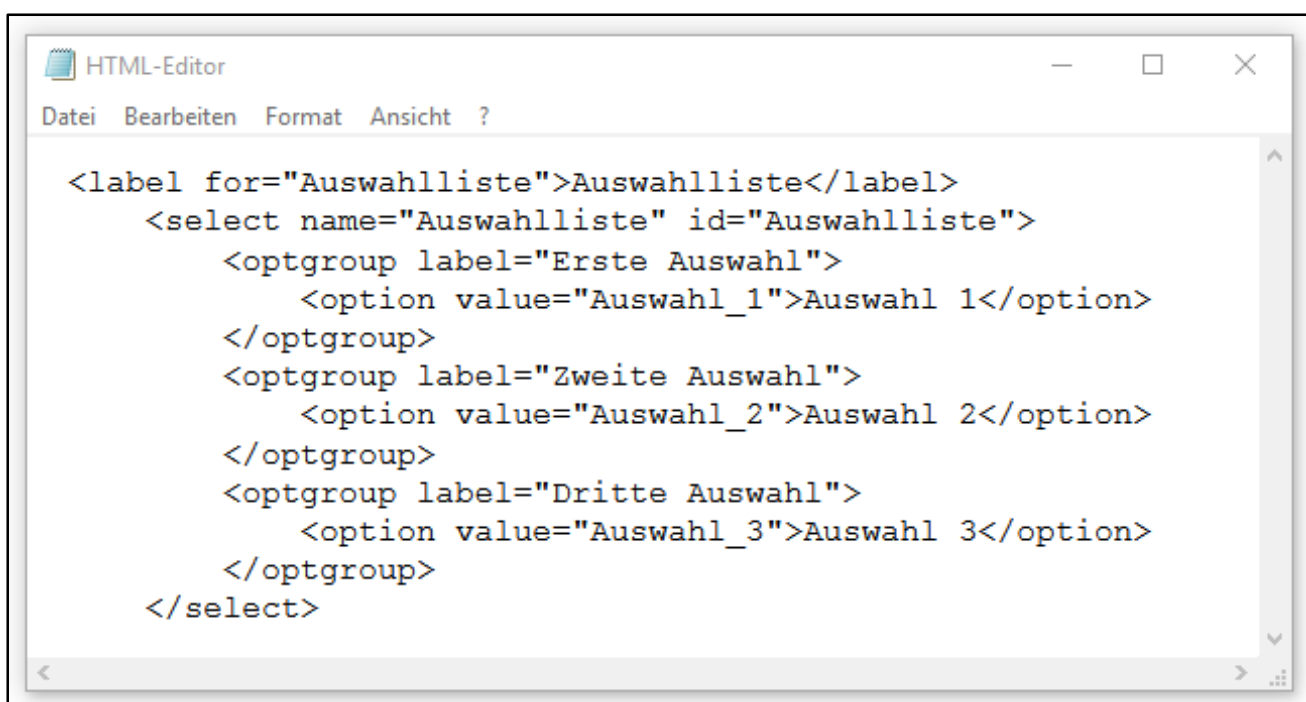


Abb. 43: Der <optgroup>-Tag – HTML

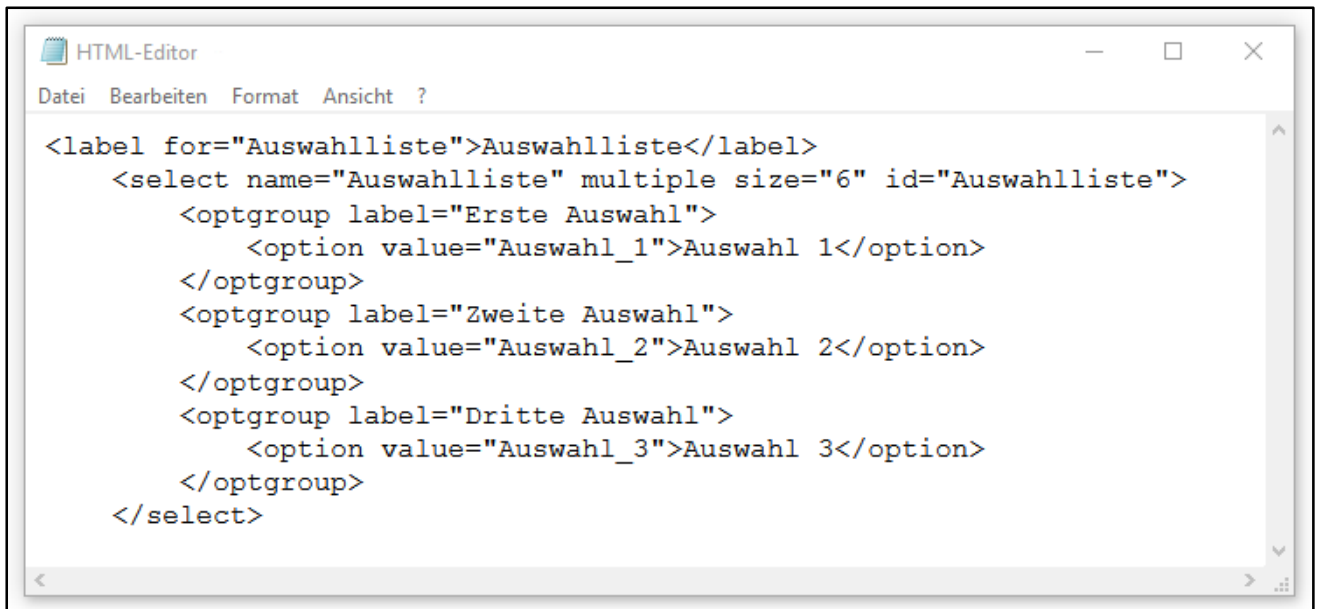
Was es bei den Überschriften des <optgroup>-Tag zu beachten gilt:

- Sie sind nicht auswählbar
- Sie dienen nur zur Gruppierung der Optionen
- Sie gehören ins <label>-Attribut und nicht in den Tag-Body
- Im Tag-Body stehen nur die Optionen, die zur jeweiligen Gruppe gehören

4.2.4 Attribute von Select-Boxen: „multiple“ und „size“

Der <select>-Tag bietet mit dem Attribut "**multiple**" eine Möglichkeit die Radio-Buttons nicht haben: es kann **mehrere Elemente einer Liste auswählen**.

Das Attribut "**size**" gibt dabei an, **wie viele Zeilen** in der Liste zu sehen sind, ohne dass dabei in der Liste gescrollt werden muss.



```
<label for="Auswahlliste">Auswahlliste</label>
  <select name="Auswahlliste" multiple size="6" id="Auswahlliste">
    <optgroup label="Erste Auswahl">
      <option value="Auswahl_1">Auswahl 1</option>
    </optgroup>
    <optgroup label="Zweite Auswahl">
      <option value="Auswahl_2">Auswahl 2</option>
    </optgroup>
    <optgroup label="Dritte Auswahl">
      <option value="Auswahl_3">Auswahl 3</option>
    </optgroup>
  </select>
```

Abb. 44: Die Attribute „multiple“ und „size“ – HTML

Hinweis: Das Attribut "multiple"

In einer Select-Box mit Mehrfachauswahl können mehrere Optionen ausgewählt werden, indem die Strg-Taste gedrückt gehalten wird und dabei die verschiedenen Werte angeklickt werden.

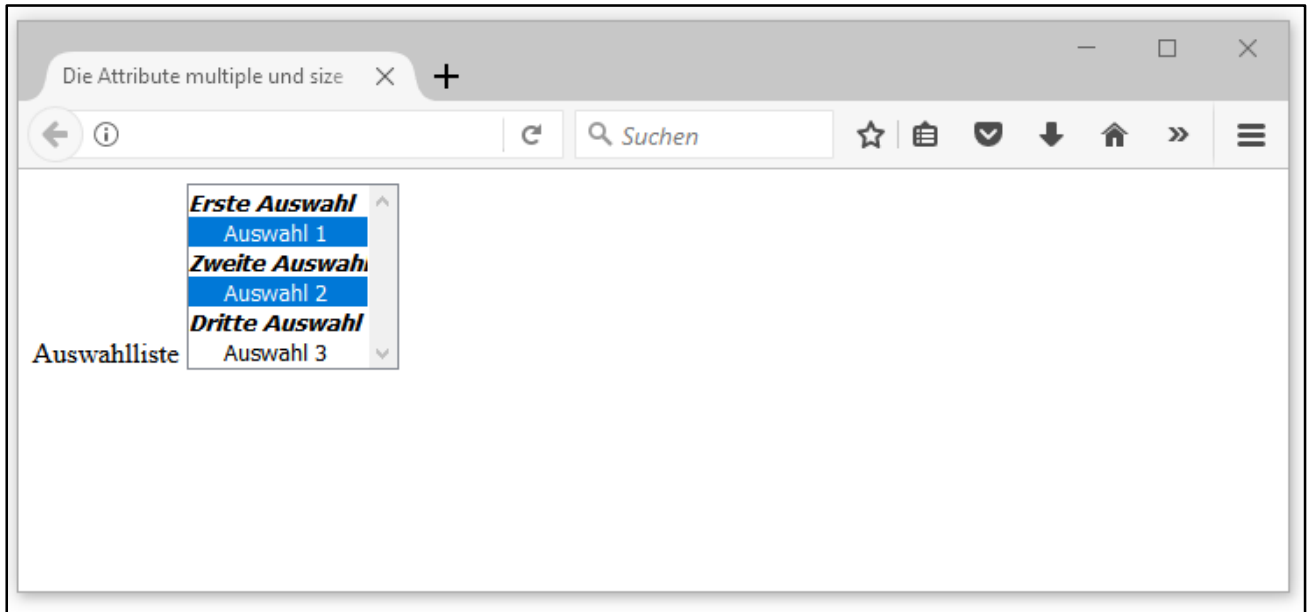


Abb. 45: Die Attribute „multiple“ und „size“ – Browseransicht

4.2.5 Übung zu Select-Boxen

Bitte erstellen Sie nun anhand des in diesem WBT erlernten Wissens zu Select-Boxen nebenstehendes Formular unter Berücksichtigung folgender Aufgabenstellung:

- Erstellen Sie eine Auswahlliste mit 8 Produkten
- Die 8 Produkte sollen in 3 Kategorien eingeteilt sein
- Jede Kategorie bekommt ihre eigene Überschrift
- In der Liste sollen mehrere Produkte anwählbar sein
- Angezeigt werden 5 Produkte, um den Rest zu sehen, soll gescrollt werden.

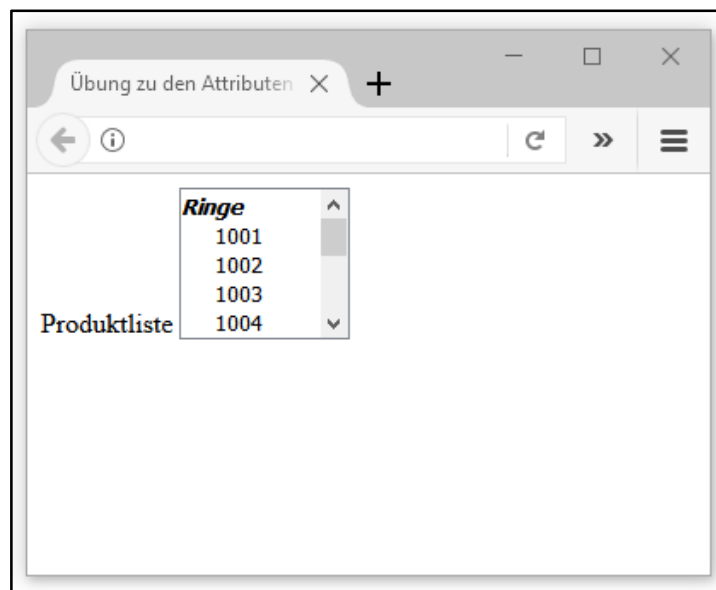
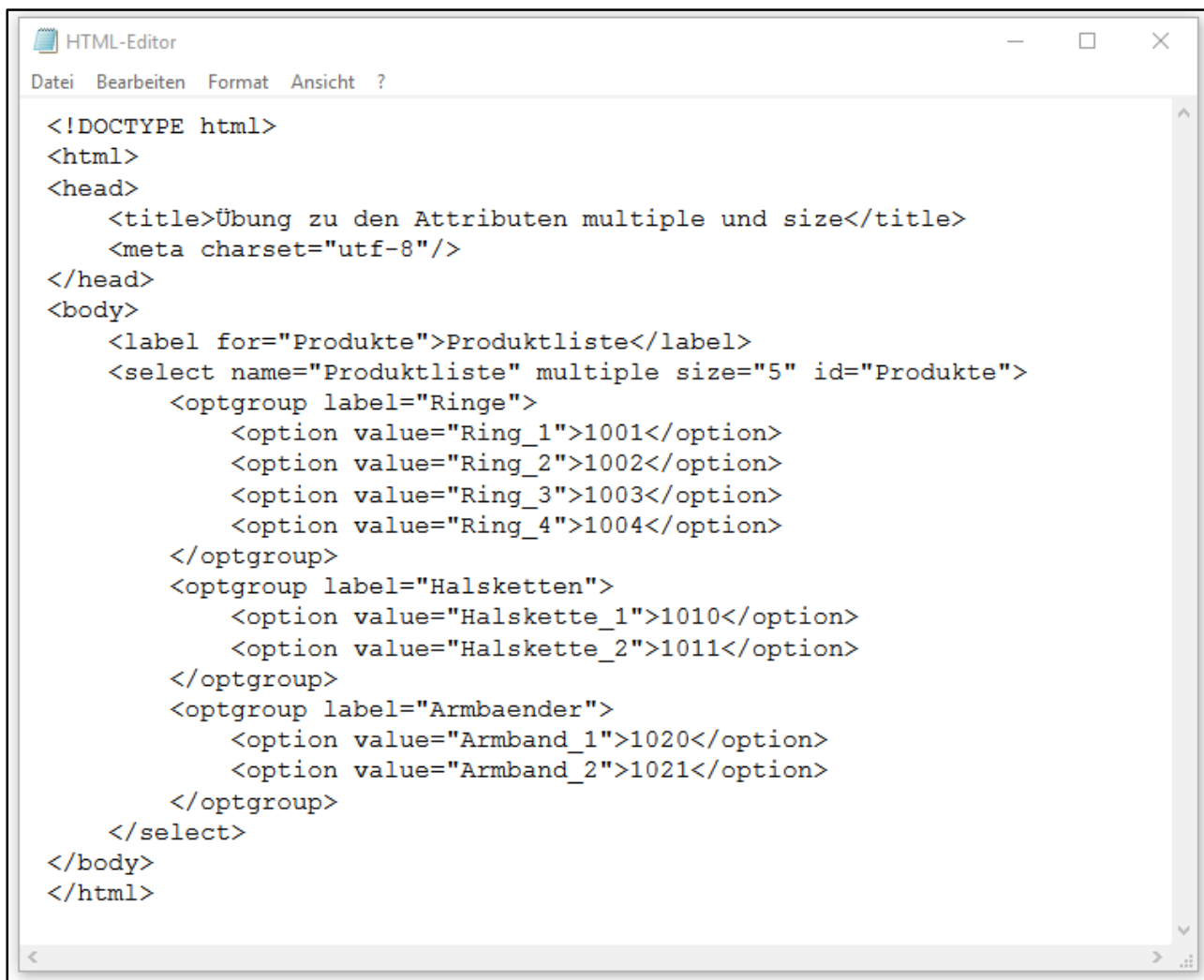


Abb. 46: Übung zu Select-Boxen



```
HTML-Editor
Datei Bearbeiten Format Ansicht ?

<!DOCTYPE html>
<html>
<head>
  <title>Übung zu den Attributen multiple und size</title>
  <meta charset="utf-8"/>
</head>
<body>
  <label for="Produkte">Produktliste</label>
  <select name="Produktliste" multiple size="5" id="Produkte">
    <optgroup label="Ringe">
      <option value="Ring_1">1001</option>
      <option value="Ring_2">1002</option>
      <option value="Ring_3">1003</option>
      <option value="Ring_4">1004</option>
    </optgroup>
    <optgroup label="Halsketten">
      <option value="Halskette_1">1010</option>
      <option value="Halskette_2">1011</option>
    </optgroup>
    <optgroup label="Armbaender">
      <option value="Armband_1">1020</option>
      <option value="Armband_2">1021</option>
    </optgroup>
  </select>
</body>
</html>
```

Abb. 47: Lösung zu Select-Boxen

4.3 Datalist-Tag

4.3.1 Die Datenliste

Herr Neumann: „Neben der Select-Box und den Radio-Buttons gibt es noch eine weitere Alternative, um Auswahllisten zu erstellen: die **Datenliste**.

Eine Datenliste ist eine Liste von **Einträgen**, die dem Benutzer **vorgeschlagen** werden, wenn er in ein Texteingabefeld tippt.

Der Benutzer kann diese Vorschläge annehmen, aber auch einen beliebigen anderen Text eingeben.“

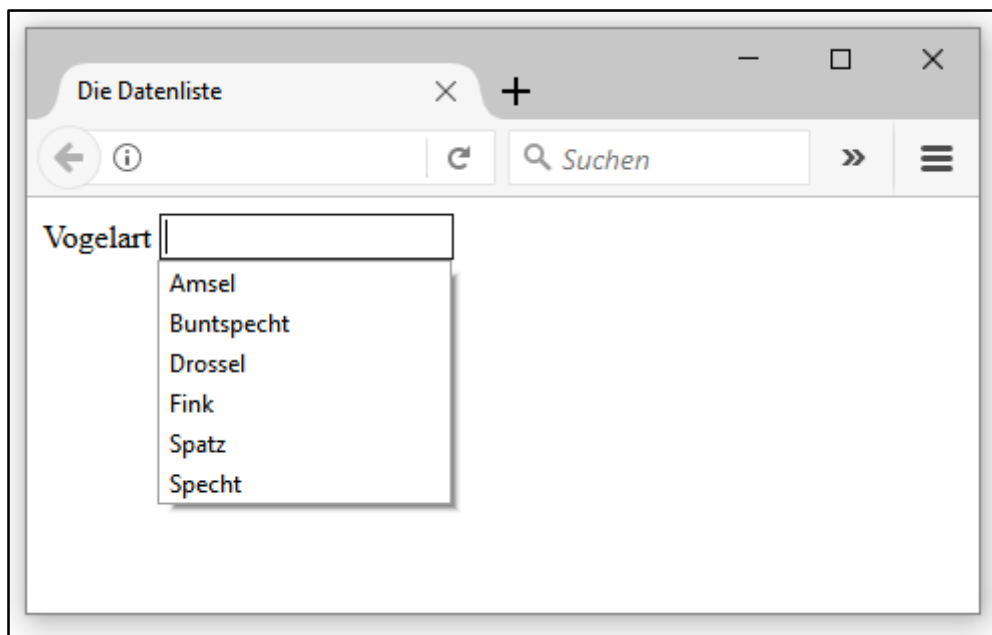


Abb. 48: Die Datenliste

4.3.2 Der <datalist>-Tag

Eine Datenliste wird in HTML durch den **<datalist>-Tag** eingeleitet. Wie auch beim <select>-Tag werden **<option>-Tags** in den <datalist>-Tag verschachtelt, um die vorzuschlagenden Einträge zu definieren.

Anders als beim <select>-Tag wird der Tag-Body des <option>-Tag hier ignoriert. Die angezeigten Einträge kommen einzig aus dem **value-Attribut**.

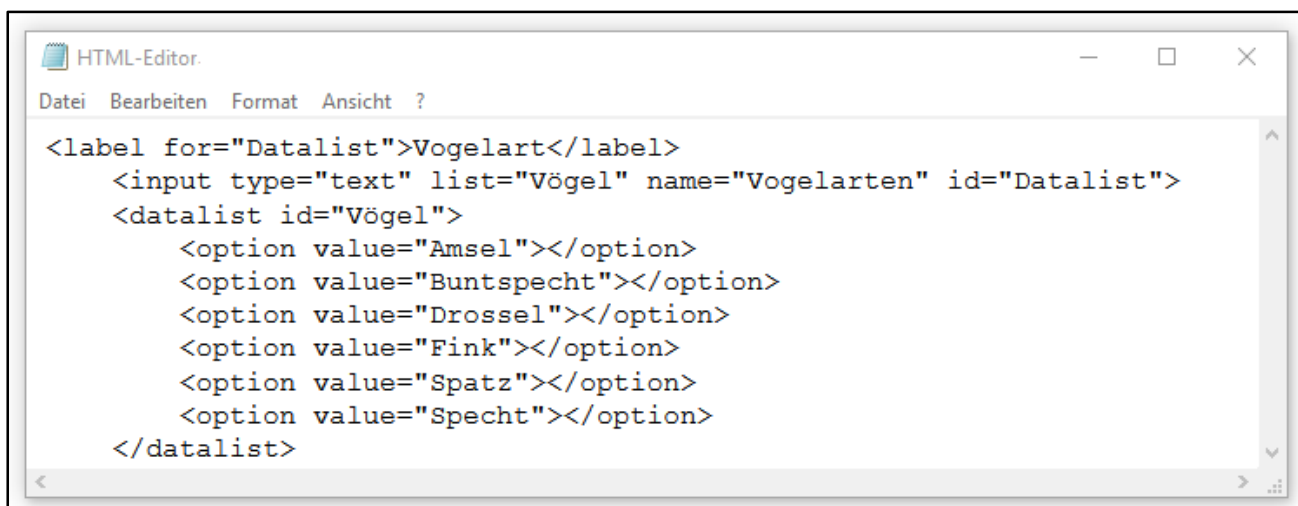


Abb. 49: Der <datalist>-Tag – HTML

Wichtig: Um die Datenliste mit dem Texteingabefeld zu verknüpfen, benötigt der <datalist>-Tag eine id. Zusätzlich muss ein Attribut "list" im <input>-Tag auf die id der Datalist verweisen!

4.3.3 Übung zu Datenlisten

Nr.	Frage	Richtig	Falsch
1	Eine Datenliste ist eine Liste von Einträgen, die dem Benutzer vorgeschlagen werden, wenn er in ein Texteingabefeld tippt. Eines der vorgeschlagenen Einträge muss gewählt werden.		
	Richtig		
	Falsch		
2	Der Tag-Body des <option>-Tags wird bei Datenlisten ignoriert. Die angezeigten Einträge kommen aus dem value-Attribut.		
	Richtig		
	Falsch		
3	Möglichkeiten eine Auswahlliste zu erstellen bieten:		
	Radio-Buttons		
	Checkboxen		
	Select-Boxen		
	Datenlisten		
4	Die angezeigten Einträge der Auswahlliste werden durch den Tag-Body des <option>-Tags erzeugt.		
	Richtig		
	Falsch		
5	Eine Datenliste wird in HTML durch den <option>-Tag eingeleitet.		
	Richtig		
	Falsch		
6	Die <option>-Tags werden in den <datalist>-Tag verschachtelt.		
	Richtig		
	Falsch		
7	Im list-Attribut wird auf die id der Datalist verwiesen.		
	Richtig		
	Falsch		

Tab. 2: Übung zu Datenlisten

4.4 Übung

4.4.1 Das Gelernte anwenden

Lin W. Lan: „Mit dem Durcharbeiten dieses WBT haben Sie alle nötigen Kenntnisse erlangt, um auch die letzten Anforderungen an die Web Site von Casarella umzusetzen.

Bitte beenden Sie den Auftrag von Casarella anhand der Aufgabenstellungen auf den nächsten beiden Seiten.“

4.4.2 Umsetzung der Formular-Anforderungen – Teil 2

Herr Huber: „Bitte ergänzen Sie das gewünschte Bestellformular von Casarella, mit dem in diesem WBT erlangtem Wissen, um die fehlenden Felder!

Nutzen Sie hierfür das angefangene Formular aus dem letzten WBT.“

Bestellungen

Herr Frau

VORNAME*

NACHNAME*

STRASSE*

HAUSNR.*

PLZ*

ORT*

TELEFON

E-MAIL-ADRESSE*

1 - ARTIKELNUMMER*

Ringe R ARTIKEL*

Halsketten R ARTIKEL

Armbaender

Es gelten die AGB von Casarella auch hinsichtlich des
Widerrufsrechts

Ich habe die Hinweise zum Datenschutz gelesen

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 50: Formularanforderungen – Teil 2

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

4.4.3 Umsetzung der Formularanforderungen – Teil 3

Herr Neumann: „Um den Auftrag von Casarella abzuschließen, fehlt nur noch das Kontaktformular auf der Impressumsseite!

Nutzen Sie dafür die bereits vorhandene Impressumsseite und ergänzen Sie die Formularfelder.

Hinweis: Für freie Textfelder benötigt man den <textarea>...</textarea>-Tag.“

Der <textarea>-Tag

- Die Spaltenanzahl wird mit dem Attribut "cols" im <textarea>-Tag angegeben
- Die Zeilenanzahl wird mit dem Attribut "rows" im <textarea>-Tag angegeben
- Mit dem Attribut "wrap" im <textarea>-Tag kann festgelegt werden, ob mehrzeilige Texteingaben umgebrochen werden sollen
 - wrap="soft": kein Zeilenumbruch
 - wrap="hard": Zeilenumbruch

Impressum & Kontakt

Casarella

Bella Perle
Am Schmuckweiher 14
35394 Gießen, Deutschland

Telefon: 0172/1239702
E-Mail: bella.perle@casarella.de

Alle auf der Web Site von Casarella angegebenen Preise sind Endpreise zzgl. Versandkosten. Aufgrund des Kleinunternehmertums gem. §19 UStG erheben wir keine Umsatzsteuer und weisen diese daher auch nicht aus.

Hinweise zum Versand: Versand innerhalb Deutschland möglich, per Deutsche Post Warensendung. Versandkosten 1,90 €.

Sie möchten direkt Kontakt mit uns aufnehmen? Gerne können Sie mir über das nachstehende Kontaktformular Ihre Nachricht zukommen lassen.

Name*

E-Mail-Adresse*

Ihre Nachricht*

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 51: Formularanforderungen – Teil 3

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

4.5 Ausblick

Lin W. Lan: „Herzlichen Glückwunsch! In diesem WBT haben Sie gelernt, was

- Select-Boxen und
- Datenlisten

sind.

Damit haben Sie alle wichtigen HTML-Elemente kennen gelernt, die wir Ihnen in dieser WBT-Serie näherbringen.

Sie haben außerdem den Auftrag von Casarella fertiggestellt und sind bereit, in dem nächsten und letzten HTML-WBT dieser Serie, einen ersten Einblick in das Thema "Media Queries" zu erlangen.

Wir freuen uns auf Sie!“

5 Seitenstruktur mit Media Queries

5.1 Einleitung

Lin W. Lan: „Herzlich Willkommen zum letzten WBT des HTML-Teils der WBT-Serie "HTML, CSS & JavaScript – Teil 2".

In den letzten vier WBT haben Sie alle HTML-Elemente kennengelernt, um den Auftrag von Casarella erfolgreich umsetzen zu können. Und das ist Ihnen auch gelungen!

Casarella ist begeistert von den neuen Formularen und der Leichtigkeit, mit der ab sofort eine Bestellung durchgeführt und Kontakt mit Casarella aufgenommen werden kann.

Zum Abschluss des HTML-Teils dieser Serie möchten wir Ihnen gerne noch einen ersten Einblick in das Thema **Responsive Webdesign** und **Media Queries** geben, um Ihnen zu zeigen, dass die Darstellung von Web Sites für verschiedene Ausgabemedien festgelegt werden kann.“

5.2 Web-Seiten auf verschiedenen Anzeigegeräten

5.2.1 Web-Seiten auf verschiedenen Anzeigegeräten – Teil 1

Herr Neumann: „Die Art und Weise, wie heute auf das Internet zugegriffen wird, ist sehr vielseitig geworden.

Betrachten wir bspw. die Web Site vom E-Campus auf anderen Geräten als Desktop-Bildschirmen, wie bspw. dem Smartphone, ergibt sich das Problem, dass diese andere Größen und Auflösungen aufweisen.“

5.2.2 Web-Seiten auf verschiedenen Anzeigegeräten – Teil 2

Herr Neumann: „Bei konventionell gestalteten Web Sites werden Elemente zu groß oder gar nicht angezeigt und um die gesamte Web Site zu sehen, muss viel gescrollt werden.

Selbiges gilt für die Web-Site-Anzeige auf dem Tablet oder der Smartwatch.

Um dieses Problem zu lösen, gibt es das so genannte **Responsive Webdesign**.“



Abb. 52: Eine Web Site auf verschiedenen Anzeigegeräten

5.2.3 Responsive Webdesign – Teil 1

Responsive Webdesign oder "**reagierendes Webdesign**" ist eine Technik, welche es mit Hilfe von **HTML** ermöglicht, das einheitliche Anzeigen von Inhalten auf einer Web Site zu gewährleisten und dabei die Eigenschaften des Endgeräts berücksichtigt.

Hierbei wird das Layout einer Web Site so flexibel gestaltet, dass dieses auf dem Desktop, Tablet und Smartphone eine gleichbleibende Benutzerfreundlichkeit bietet. Inhalts- und Navigationselemente sowie auch der strukturelle Aufbau einer Web Site **passen sich der Bildschirmauflösung des Endgeräts an**.

Die Web Site reagiert auf ihre Umgebung, indem sie Elemente dynamisch vergrößert und verkleinert. Elemente auf schmalen Smartphone-Bildschirmen werden so bspw. untereinander dargestellt, statt nebeneinander oder weniger wichtige Elemente, wie z. B. große Bilder als Design-Elemente, werden ganz ausgeblendet.



Abb. 53: Responsive Webdesign

Eine spezielle Form des Responsive Webdesign ist das Adaptive Webdesign (= anpassungsfähiges Webdesign). Adaptive Webdesign bezieht sich auf diskrete Zustände und somit auf definierte Größen von Endgeräten (siehe 5.2.5).

5.2.4 Responsive Webdesign – Teil 2

Responsive Webdesign arbeitet mit einem **flüssigen Gestaltungsraster**. Das Layout wird nicht gezielt für einen bestimmten Anzeigebereich einer Web Site optimiert, sondern das Design ist so entwickelt, dass der zur Verfügung stehende Platz immer optimal ausgenutzt wird.

Das Responsive Webdesign orientiert sich also am Layout und nicht an den Abmessungen eines bestimmten Displays. Es passt sich der Browsergröße an, unabhängig von dem Endgerät, von dem aus die Web Site besucht wird. Somit steht die **perfekte Informationsaufbereitung** im Vordergrund.

Umsetzungshinweise im Quellcode: Um eine Web Site mit all ihren Elementen und Abständen in ihrer Größe entsprechend an alle Ausgabegeräte anpassen zu können, sollten sie nicht nach absoluten Zahlen, wie festen Pixeln, sondern nach **relativen Größen**, wie %, ausgerichtet sein.

5.2.5 Adaptives Webdesign

Herr Huber: „Adaptive Webdesign ist ein für verschiedene **Displaygrößen** optimiertes Web-Layout. Kern des Adaptive Webdesign ist ein **starres Gestaltungsraster**.

Hierbei werden verschiedene Ansichten für exakte Anzeigebereiche einer Web Site entwickelt. Üblicherweise sind das eine Desktop-Ansicht, eine Tablet-Ansicht und eine Variante für Smartphones.

Die Abmessungen der verschiedenen Ansichten orientieren sich dabei an bestimmten Geräten.

Beim Adaptive Webdesign steht somit das **Ausgabegerät** im Vordergrund, denn je nachdem auf welchem Endgerät die Web Site aufgerufen wird, variiert die Darstellung.

Ein Gerät mit kleinem Display enthält so bspw. kleinere Bild-Dateien, Geräte mit hochauflösenden Displays erhalten hochauflösende Grafiken.“

5.3 Media Queries

5.3.1 Erkennung des Anzeigegerätes

Mit Medienabfragen, sogenannten **Media Queries**, ist eine Erkennung von Anzeigegeräten möglich.

Dafür werden in den HTML-Quellcode die Media Queries eingebaut, die das jeweilige Endgerät identifizieren, auf welchem die Web Site geöffnet wird.

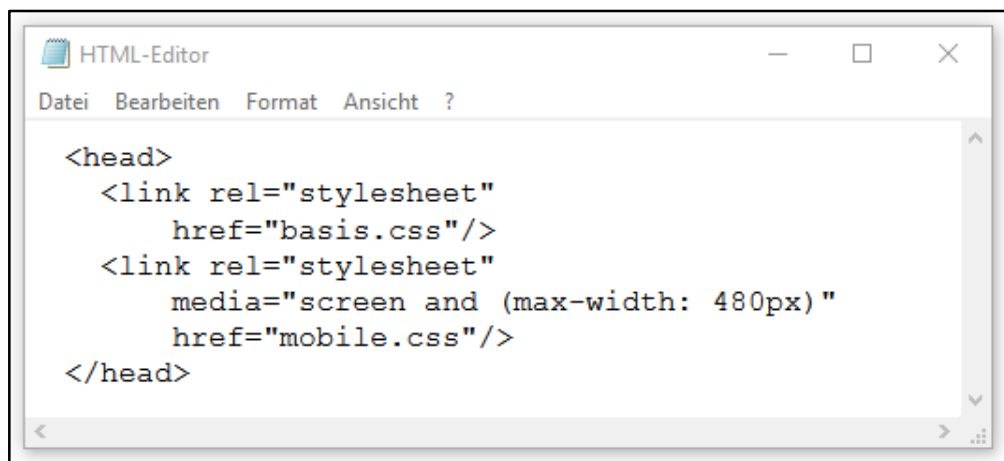
Wenn ermittelt wurde, um welches Endgerät es sich handelt, kann unter Berücksichtigung verschiedener Kriterien, wie der Größe des Endgeräts, der Bildschirmauflösung und der Orientierung (Hoch- oder Querformat) des Endgeräts, die Darstellung einer aufgerufenen Web Site festgelegt werden.

Erfüllt das verwendete Ausgabemedium alle Kriterien einer Medienabfrage, so wird die damit verknüpfte CSS-Ressource eingebunden, die ein entsprechendes Design für das Anzeigegerät vorhält.

5.3.2 Media Queries in HTML

Herr Huber: „Media Queries können auf verschiedene Arten eingebunden werden: in HTML oder in CSS. Die Einbindung in CSS lernen Sie in WBT 8 dieser Serie.

Die Verwendung einer **Medienabfrage in HTML** sieht wie folgt aus:“



```
<head>
  <link rel="stylesheet"
        href="basis.css"/>
  <link rel="stylesheet"
        media="screen and (max-width: 480px)"
        href="mobile.css"/>
</head>
```

Abb. 54: Media Queries in HTML

In diesem Beispiel wird eine Kombination aus Medientyp und Medienmerkmal verwendet.

"mobile.css" wird nur geladen, wenn die maximale Bildschirmbreite von 480 Pixeln nicht überschritten wird. Bei Geräten mit einer höheren Auflösung wird "basis.css" verwendet.

Achtung!

Die **Reihenfolge**, in der die Stylesheets angegeben werden, ist **wichtig**. Durch eine Media Query ändert sich die Priorität der CSS-Regeln nicht: Würde zuerst der Mobile-Stylesheet laden, könnte der Basis-Stylesheet dessen Regeln überschreiben.

So wie in diesem Beispiel überschreibt der Mobile-Stylesheet die Basis-Regeln.

5.3.3 Aufbau von Media Queries – Teil 1

Herr Neumann: „Da Sie nun wissen, wie Sie Media Queries in HTML einbinden, möchte ich Ihnen den **Aufbau** einer solchen Medienabfrage etwas genauer erklären.

Hierzu verwenden wir das Beispiel der vorherigen Seite.

Ein Media Query besteht in jedem Fall aus einem Medientyp (bzw. Ausgabegerät). Soll die Medienabfrage zusätzlich Medienmerkmale beachten, folgt auf den Medientypen die Verknüpfung "and". Innerhalb des Ausdrucks werden dann ein Medienmerkmal (oder auch eine Eigenschaft) und ein entsprechender Wert zwischen zwei runden Klammern notiert.“

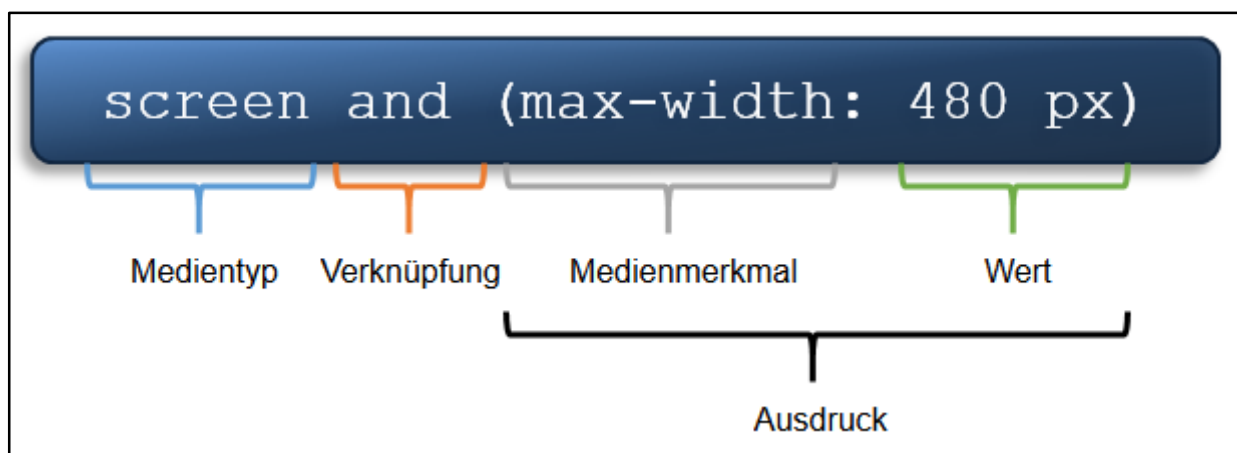


Abb. 55: Aufbau von Media Queries

5.3.4 Aufbau von Media Queries – Teil 2

Herr Neumann: „Schauen wir uns den Aufbau von Media Queries noch einmal genauer an:“

Medientyp:

Der Medientyp gibt an, für welches Ausgabemedium das Stylesheet festgelegt werden soll.

Dabei kann ein einziger Typ oder auch mehrere, getrennt durch ein Komma, angegeben werden.

Bezieht sich das Stylesheet auf bildschirmorientierte Ausgabegeräte wird bspw. "screen" verwendet.

Hinweis: Um eine Auflistung gängiger Ausgabegeräte zu erhalten, laden Sie sich bitte auf dieser Seite in dem WBT beigefügte Dokument herunter.

Verknüpfung:

Soll sich das Stylesheet nicht nur am Medientyp, sondern auch am Medienmerkmal orientieren, wird der Medientyp mit dem Schlüsselwort "and" verknüpft.

Dabei können auch mehrere and-Merkmale verknüpft und verarbeitet werden.

Ein Stylesheet wird nur dann verwendet, wenn alle mit "and" verbundenen Ausdrücke und Kriterien erfüllt werden.

Medienmerkmal:

Verschiedene Ausgabegeräte verfügen auch über viele verschiedene Merkmale. Das am häufigsten abgefragte Merkmal ist die **Minimal- und Maximalbreite der Anzeigefläche**.

Hinweis: Um eine Auflistung gängiger Ausgabegeräte zu erhalten, laden Sie sich bitte auf dieser Seite in dem WBT beigefügte Dokument herunter.

Wert

Der Wert bezieht sich auf das Medienmerkmal. In diesem Beispiel gibt er an, dass ein vorhergehendes Stylesheet nur geladen wird, wenn die maximale Bildschirmbreite von 480 Pixeln nicht überschritten wird.

5.4 Übung

5.4.1 Das Gelernte festigen

Herr Huber: „In diesem WBT haben Sie gelernt, was Responsive Webdesign ist und wie Media Queries verwendet werden.“

Bevor Sie den HTML-Teil dieser WBT-Serie abschließen, haben wir Ihnen noch einige Fragen zusammengestellt, um ihr erlerntes Wissen zur Seitenstruktur mit Media Queries zu festigen.

Viel Erfolg!“

5.4.2 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Gegenüber älteren Methoden bietet das responsive Webdesign den Vorteil, dass man eine angepasste Variante zusätzlich zu der originalen HTML-Datei hat.		
	Richtig		
	Falsch		
2	Die Web-Site-Anzeige auf der Smartwatch passt sich automatisch an.		
	Richtig		
	Falsch		
3	Beispiele für Medientypen sind:		
	screen		
	handheld		
	all		
4	Betrachtet man eine Web Site auf verschiedenen Ausgabegeräten, ergibt sich das Problem, dass die _____ und die _____ variieren.		
5	Das Responsive Webdesign sorgt dafür, dass die Web Site auf Eigenschaften des jeweils benutzten Endgeräts, vor allem Smartphones und Tablet-Computer, reagieren kann.		
	Richtig		
	Falsch		
6	Medienabfragen in HTML sehen wie folgt aus: @media screen		
	Richtig		
	Falsch		
7	Für die Technik des Responsive Webdesign benötigt man ausschließlich HTML.		
	Richtig		
	Falsch		
8	Media Queries sind der Kern des_____.		
9	Beispiele für Medienmerkmale sind:		
	height		

	high		
	width		
	monochrome		
10	Beim Responsive Webdesign passen sich die Inhalts- und Navigationselemente sowie auch der strukturelle Aufbau der Web Site der Bildschirmauflösung des mobilen Endgeräts an.		
	Richtig		
	Falsch		
11	Erfüllt das verwendete Ausgabemedium alle Kriterien einer Medienabfrage, so wird die damit verbundene CSS-Ressource eingebunden.		
	Richtig		
	Falsch		
12	Durch eine Media Query ändert sich die Priorität der CSS-Regeln nicht.		
	Richtig		
	Falsch		

Tab. 3: Abschlusstest WBT 05

5.4.3 Typische Klausurfrage

Erläutern Sie, warum Sie zur Darstellung von Elementen in HTML nur prozentuale und nicht absolute Werte nutzen sollten.

Hinweis: Einen Lösungsvorschlag finden Sie als ein zum Download bereitstehendes Dokument im WBT.

5.5 Ausblick

Lin W. Lan: „Mit dem Durcharbeiten dieses WBT haben Sie den HTML-Teil der WBT-Serie "HTML, CSS & JavaScript – Teil 2" abgeschlossen.

In WBT 03 "Media Queries in CSS" des CSS-Teils dieser Serie, werden Sie anknüpfend an dieses WBT einen tiefgehenden Einblick in das Thema **Media Queries** erhalten. Sie werden dabei lernen, dass Sie Media Queries, neben der Ihnen nun bekannten Weise, auch in CSS einbinden können.

Das nächste WBT "WBT 01 – CSS für Formulare" ist das erste WBT des CSS-Teils dieser Serie und führt in die **Gestaltung von Formularen mit Hilfe von CSS** ein.

Im Laufe des CSS-Teils werden wir Ihnen zeigen, wie verschiedene HTML-Elemente mit CSS ansprechender gestaltet werden können. Bis zum nächsten Mal. Wir freuen uns auf Sie!“

6 CSS für Formulare

6.1 Einleitung

Lin W. Lan: „Hallo! Willkommen zum CSS-Teil der WBT-Serie "HTML, CSS & JavaScript – Teil 2".

Im ersten Teil der WBT-Serie haben Sie bereits gelernt, was Formulare sind, wozu diese verwendet und wie diese in HTML erstellt werden können.

Durch Radio-Buttons, Checkboxes, den Select-Tag und Datalist-Tag haben Sie die Darstellung der Formulare anpassen und nach Ihren Bedürfnissen ausrichten können.

Mit den nachfolgenden WBT werden Sie nun lernen, wie die **Formulare mit CSS optisch gestaltet** werden können.

Dabei sollen die folgenden Ausführungen als kreative Anregungen verstanden werden, wie Sie dabei vorgehen können.“

6.2 Formulare mit CSS gestalten

6.2.1 Formulare mit CSS gestalten

Herr Huber: „Die Formulare, die Sie in den vergangenen WBT erstellt haben, waren optisch sehr einfach gestaltet. Schrifteigenschaften, Farben, Rahmen, all das was sie bereits in der ersten WBT-Serie "HTML, CSS & JavaScript – Teil 1" gelernt haben, ist jedoch auch bei Formularen anwendbar.

Aber es gibt auch CSS-Deklarationen, die speziell für Formulare da sind: die drei Pseudoklassen

- zu Pflichtfeldern
- zur Validierung und
- zur Fokussierung,

die wir Ihnen auf den nachfolgenden Seiten näherbringen.“

6.2.2 Pseudoklassen zur Gestaltung von Formularen

Bei Pseudoklassen handelt es sich um einfache Selektoren, die ein Element dann ansprechen, wenn es eine bestimmte Eigenschaft besitzt. Pseudoklassen beginnen im Stylesheet immer mit einem Doppelpunkt. Schriftarten und -farben werden, wie in der ersten WBT-Serie gelernt, angewandt.

Herr Neumann: „Casarella möchte, dass ihre Kunden leichter erkennen können, welche Formularfelder Pflichtfelder sind.

Um dies umzusetzen, kann so beispielsweise mit Hilfe der Pseudoklasse für Pflichtfelder, um das Eingabefeld des Pflichtfeldes ein dickerer Rahmen gesetzt werden, als bei den Feldern, bei denen keine Eingabe erforderlich ist.“

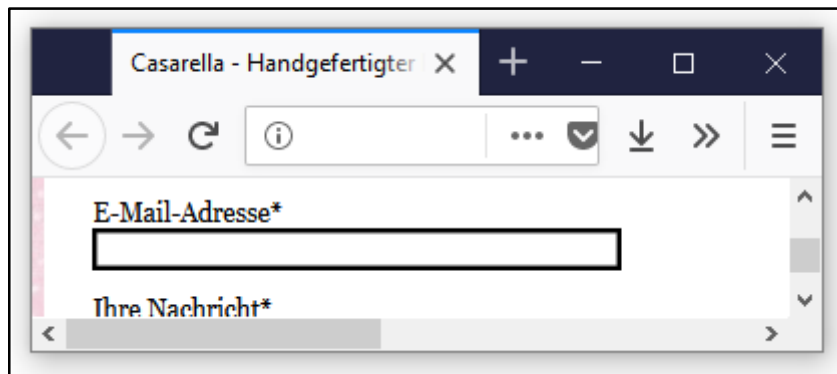


Abb. 56: Pseudoklasse zu Pflichtfeldern – Browseransicht

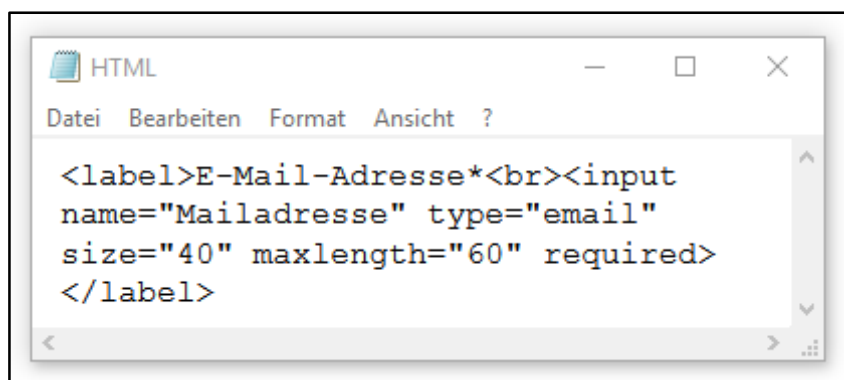


Abb. 57: Pseudoklasse zu Pflichtfeldern – HTML

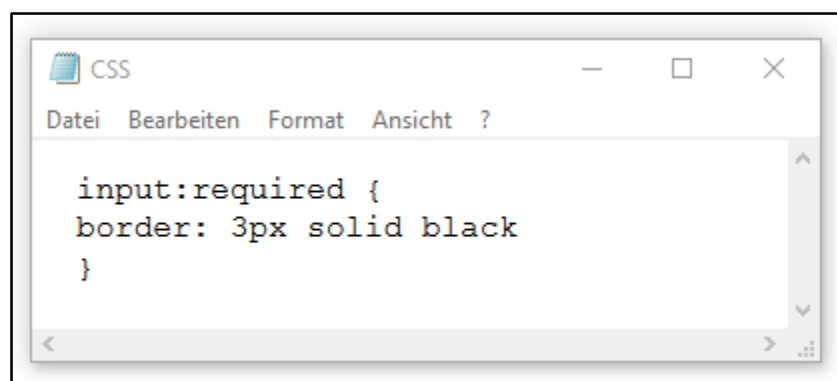


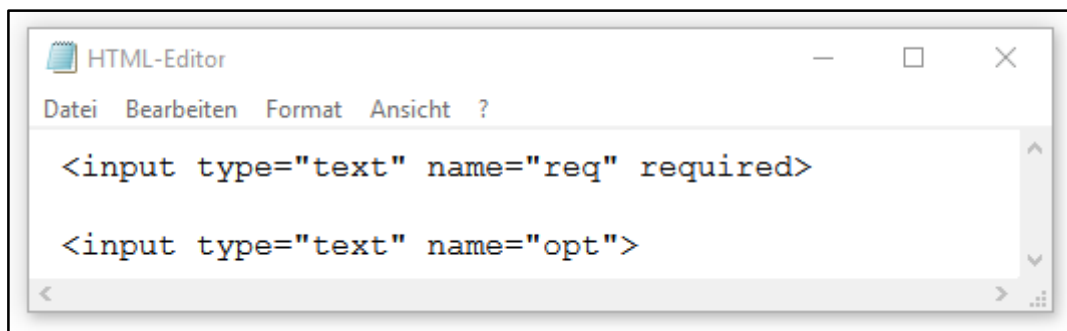
Abb. 58: Pseudoklasse zu Pflichtfeldern – CSS

6.2.3 Pflichtfelder in CSS gestalten

Die Pseudoklassen `:required` und `:optional` beziehen sich auf das `required`-Attribut, also auf die Pflichtfelder bei den Eingabefeldern eines Formulars.

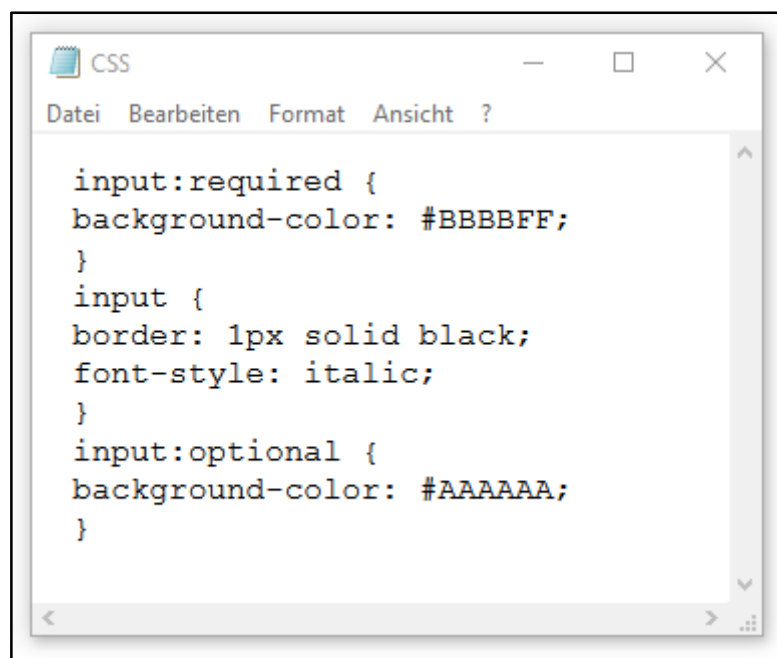
Setzt man in CSS die Style-Regel `:required`, spricht diese Regel das in dem HTML-Formular gesetzte `required`-Attribut an. `:optional` spricht hingegen die `<input>`-tags an, die kein `required`-Attribut gesetzt haben.

Wählt man weder `input:required`, noch `input:optional`, spricht die CSS-Regel `input`, jeden `<input>`-tag an.



```
HTML-Editor
Datei Bearbeiten Format Ansicht ?
<input type="text" name="req" required>
<input type="text" name="opt">
```

Abb. 59: `input:required` und `input:optional` – HTML



```
CSS
Datei Bearbeiten Format Ansicht ?
input:required {
background-color: #BBBBFF;
}
input {
border: 1px solid black;
font-style: italic;
}
input:optional {
background-color: #AAAAAA;
}
```

Abb. 60: `input:required` und `input:optional` – CSS

6.2.4 Fehlerhafte Eingaben

Lin W. Lan: „Um die Benutzerfreundlichkeit auf der Web Site noch weiter zu erhöhen, möchte Casarella zudem, dass bei der Eingabe eines Kunden in ein Formularfeld ein Hinweis erscheint, wenn diese Eingabe fehlerhaft ist. Aber was ist eigentlich eine fehlerhafte Eingabe?“

Fehlerhafte Eingaben in einem Formularfeld

Fehlerhafte Eingaben in einem Formularfeld liegen dann vor, wenn die gemachte Eingabe eines Kunden nicht mit den erforderlichen Zeichen des Eingabefeldes übereinstimmen.

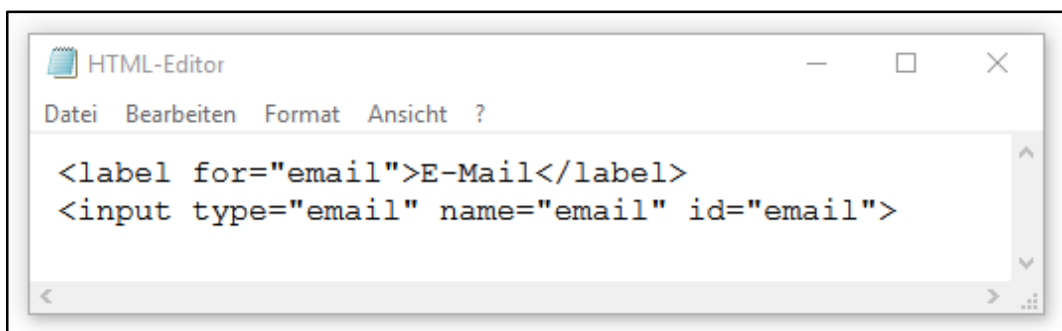
Dies kann beispielsweise sein:

- fehlendes "@"-Zeichen bei der Eingabe der E-Mail-Adresse
- fehlender "." bei der Eingabe der E-Mail-Adresse
- Texteingabe bei dem Eingabefeld der Postleitzahl
- Zahleneingabe bei einem Texteingabefeld
- Textangabe statt Datumsangabe
- etc.

6.2.5 Fehlerhafte Eingaben in Formularen hervorheben

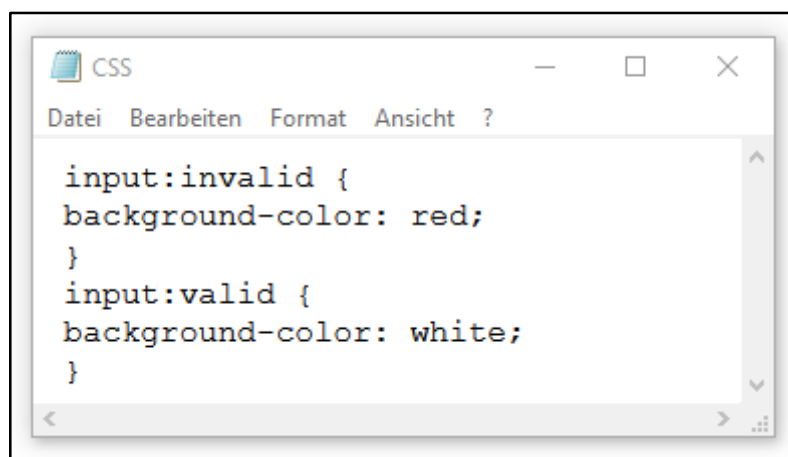
Möchte man durch eine farbliche Hervorhebung darauf aufmerksam machen, dass ein Eingabefeld eine inkorrekte Eingabe enthält, bieten sich die Pseudoklassen **:valid** und **:invalid** an.

Diese Pseudoklassen hängen mit dem Validierungsmechanismus zusammen, sodass ein Eingabefeld mit einer korrekten Eingabe die Pseudoklasse `:valid` hat und mit einer inkorrekten Eingabe `:invalid` erhält.



```
HTML-Editor
Datei Bearbeiten Format Ansicht ?
<label for="email">E-Mail</label>
<input type="email" name="email" id="email">
```

Abb. 61: Fehlerhafte Eingaben in Formularfelder hervorheben – HTML



```
CSS
Datei Bearbeiten Format Ansicht ?
input:invalid {
background-color: red;
}
input:valid {
background-color: white;
}
```

Abb. 62: Fehlerhafte Eingaben in Formularfelder hervorheben – CSS

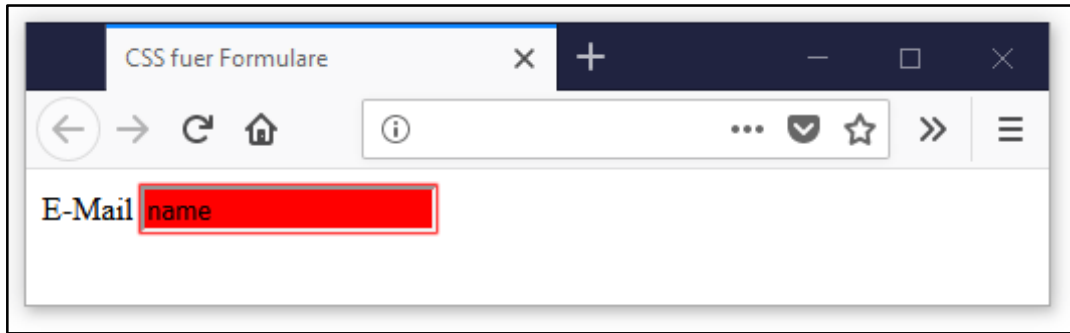


Abb. 63: Fehlerhafte Eingaben in Formularfelder hervorheben – HTML & CSS

6.2.6 Fokussierung von Formularfeldern

Herr Huber: „Sinnvoll ist es außerdem, Formularfelder hervorzuheben, die in dem Moment von dem Benutzer **fokussiert** werden.“

Das ist vor allem bei Formularen nützlich, die sehr viele Eingabefelder besitzen und der Benutzer schnell den Überblick verlieren kann, welches Eingabefeld er gerade ausfüllt.

Casarella hat uns beauftragt, diese Pseudoklassen zur Fokussierung bei ihrem Bestellformular anzuwenden.

Dabei kann zwischen zwei Fokussierungen unterschieden werden, die Ihnen auf den nächsten beiden Seiten vorgestellt werden:

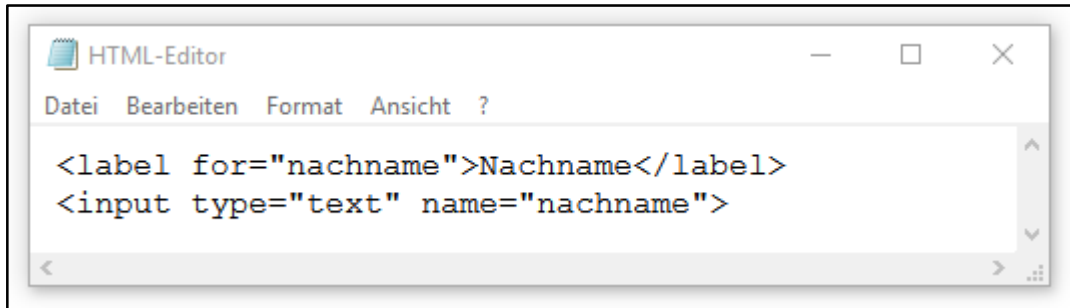
- Fokussierung der Formularfelder, die **in Bearbeitung** sind,
- Fokussierung der Formularfelder, die per **Mausover anvisiert** werden.“

6.2.7 Formularfelder hervorheben, die in Bearbeitung sind

Die letzten beiden Pseudoklassen zur Fokussierung gibt es zwar nicht nur bei Eingabefeldern von Formularen, jedoch sind sie die am häufigsten verwendeten: die Pseudoklassen **:focus** und **:hover**.

Die Pseudoklasse **:focus** bekommt das Element, das gerade vom Benutzer bearbeitet wird. Das kann ein Eingabefeld sein, in das er tippt, aber auch ein Link, auf den gerade geklickt wird oder der mit der Tab-Taste ausgewählt wurde. Eingabefeldern einen **:focus**-Style zu geben, gibt dem Benutzer ein besseres Feedback, in welchem Feld er sich gerade befindet.

Herr Huber: „Hier ein Beispiel: Bei der Fokussierung des Felds "Nachname" wird das Eingabefeld grün hinterlegt.“

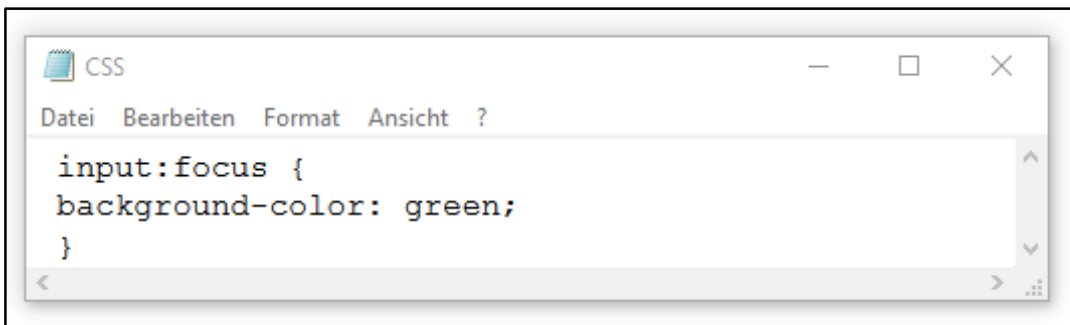


```

HTML-Editor
Datei Bearbeiten Format Ansicht ?
<label for="nachname">Nachname</label>
<input type="text" name="nachname">

```

Abb. 64: :focus – HTML



```

CSS
Datei Bearbeiten Format Ansicht ?
input:focus {
background-color: green;
}

```

Abb. 65: :focus – CSS

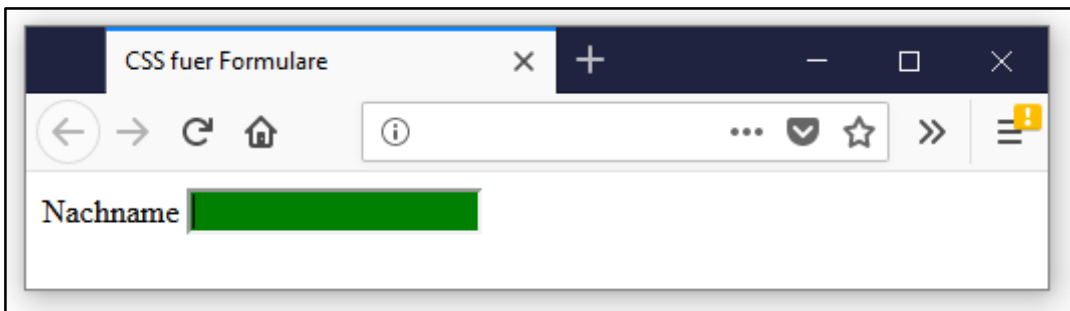
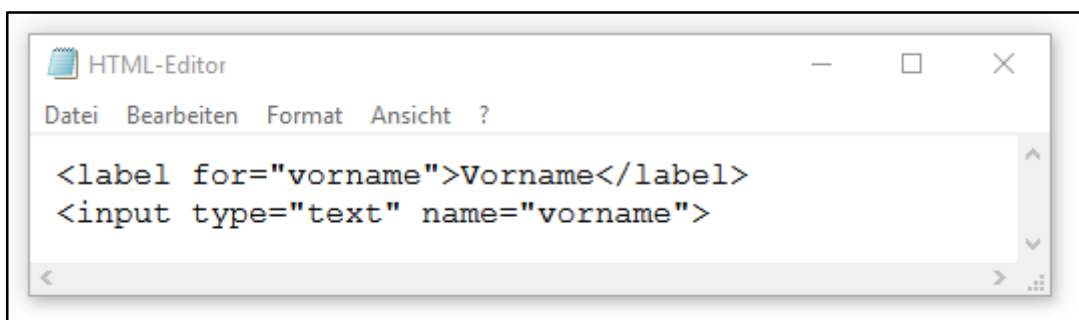


Abb. 66: :focus – HTML & CSS

6.2.8 Formularfelder bei Mausover gestalten

Herr Neumann: „Mit der Pseudoklasse :hover hebt man das Element hervor, dass vom Benutzer mit dem Mauszeiger berührt wird.“

Achtung: Bei der Benutzung von Geräten mit Touchscreen, funktioniert diese Pseudoklasse nicht.



```

HTML-Editor
Datei Bearbeiten Format Ansicht ?
<label for="vorname">Vorname</label>
<input type="text" name="vorname">

```

Abb. 67: :hover – HTML



```
input:hover {  
border-color: yellow;  
}
```

Abb. 68: :hover – CSS

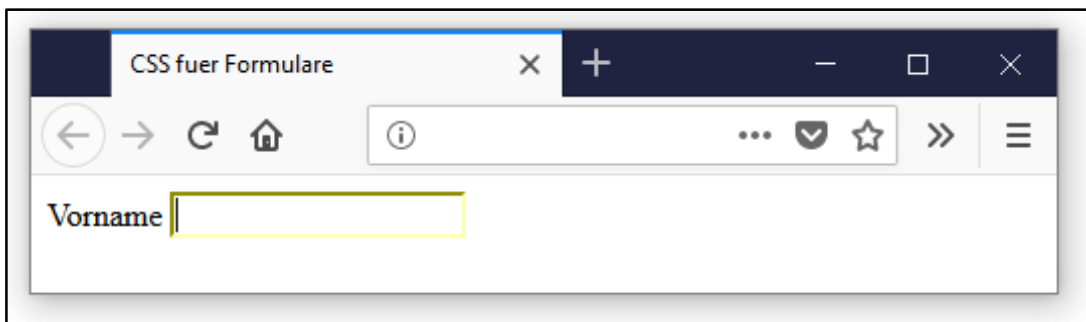


Abb. 69: :hover – HTML & CSS

6.3 Das Wissen vertiefen

6.3.1 Anforderungen an die Web Site umsetzen

Lin W. Lan: „Bitte nehmen Sie nun die von Casarella gewünschten Anpassungen am nebenstehenden Bestellformular vor.“

Nutzen Sie hierfür das in WBT 04 "Select-Tag und Datalist-Tag" fertiggestellte Formular.“

Bestellungen

Herr Frau

VORNAME*

NACHNAME*

STRASSE*

HAUSNR.*

PLZ*

ORT*

TELEFON

E-MAIL-ADRESSE*

1 - ARTIKELNUMMER*
 ▼

- Ringe**
 - 1001 R.ARTIKEL*
 - 1002
 - 1003 MMER
 - 1004
- Halsketten**
 - 1010 R.ARTIKEL
 - 1011
- Armbänder**
 - 1020 DE (falls erforderlich)
 - 1021

Es gelten die AGB von Casarella auch hinsichtlich des Widerrufsrechts

Ich habe die Hinweise zum Datenschutz gelesen

Durch * gekennzeichnete Felder sind erforderlich.

Abb. 70: Formularanforderungen – Teil 4

Hinweis: An dieser Stelle im WBT finden Sie einen zum Download bereitstehenden Ordner, indem die detaillierte Aufgabenstellung sowie die Lösung enthalten sind.

6.3.2 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Pseudoklassen sprechen immer ein Element an.		
	Richtig		
	Falsch		
2	Die Hintergrundfarbe des Eingabefeldes einer korrekten Eingabe kann durch die Pseudoklasse :valid geändert werden.		
	Richtig		
	Falsch		
3	Mit der Pseudoklasse :hover hebt man das Element hervor, das vom Benutzer mit dem Mauszeiger angeklickt wird.		
	Richtig		
	Falsch		
4	Inkorrekte Eingabefelder können durch die Pseudoklasse :focus hervorgehoben werden.		
	Richtig		
	Falsch		
5	:optional spricht die <input>-tags an, die ein required-Attribut gesetzt haben.		
	Richtig		
	Falsch		
6	Die Pseudoklassen :required und :optional beziehen sich auf das _____-Attribut bei den Eingabefeldern eines Formulars.		
7	Pseudoklassen beginnen im Stylesheet immer mit einem _____.		
8	Bei Formularen auch verwendbar:		
	Schrifteigenschaft		
	Farbe		
	Rahmen		
9	Die Pseudoklasse :focus bekommt das Element, das gerade vom Benutzer den Fokus hat. Das kann ein Eingabefeld sein, in das er tippt, aber auch ein Link, auf den gerade geklickt wird.		
	Richtig		

	0Falsch		
10	:focus und :hover werden selten für Formulare verwendet.		
	Richtig		
	Falsch		

Tab. 4: Abschlusstest WBT 06

6.3.3 Typische Klausurfrage

Beschreiben Sie, mit welchen Effekten ein Web-Formular mit den Pseudoklassen ausgestaltet werden kann,

Hinweis: Einen Lösungsvorschlag finden Sie als ein zum Download bereitstehendes Dokument im WBT.

6.4 Ausblick

Lin W. Lan: „In diesem WBT haben Sie gelernt, wie Sie

- Formulare mit CSS optisch gestalten können,
- Elemente mit Pseudoklassen ansprechen und
- Pseudoklassen zu Pflichtfeldern, zur Validierung und zur Fokussierung erstellen.

In dem nächsten WBT werden Sie lernen, wie Sie digitale Gestaltungsraster, das so genannte CSS-Grid-Layout, erstellen können.

Wir freuen uns auf Sie!“

7 CSS-Grid-Layout

7.1 Einleitung

Lin W. Lan: „Hallo! Willkommen zurück! Ich bin gestern auf Suche nach neuen Entwicklungen im Bereich "Web Sites" und "HTML, CSS und Java-Script" auf etwas Spannendes gestoßen. Es heißt **CSS Grid** und soll die Gestaltung von Seitenstrukturen stark vereinfachen. Das klingt ja super! Ich frage mal unseren Web-Designer Herrn Huber, ob er auch schon davon gehört hat.“

7.2 Seitenstruktur mit CSS Grid

Herr Huber: „Hallo Lin W. Lan, ja von dem neuen CSS Grid habe ich gestern auch gelesen und heute Morgen bereits die relevanten Informationen dazu gesammelt. Ich sage es Ihnen direkt, CSS Grid ist eine geniale Neuigkeit in der Seitengestaltung von Web Sites.“

Je komplexer Web Sites und ihre Layouts gestaltet sind, umso schwerer wird es die Layouts float-basiert umzusetzen.

CSS Grid ermöglicht nun die Seitengestaltung mit Rastern und so werden auch komplexe Layouts perfekt umgesetzt. Und das auch noch viel flexibler und übersichtlicher als zuvor mit float-basierten Layouts. Ich zeige Ihnen gerne, wie das funktioniert.“

7.3 Gestaltung der Seitenstruktur mit dem CSS-Grid-Layout

Herr Huber: „Okay, los geht's!

Zuerst wird ein zweidimensionales Raster benötigt, welches die Bereiche unserer Web Site eingrenzt. Das kennen Sie bereits aus der Web-Site-Planung, dort haben wir die Bereiche der Web Site in einem **Wireframe** grafisch dargestellt.

Für CSS Grid werden nun alle Inhaltsbereiche z. B. des Bodys in ein solches Raster geschrieben. Ein einfaches **CSS-Grid- Raster** sieht beispielsweise so aus.“

Wireframe:

Herr Huber: „Erinnern Sie sich, mit diesem Wireframe haben wir die grundlegende Struktur der Web Site von Casarella geplant.“

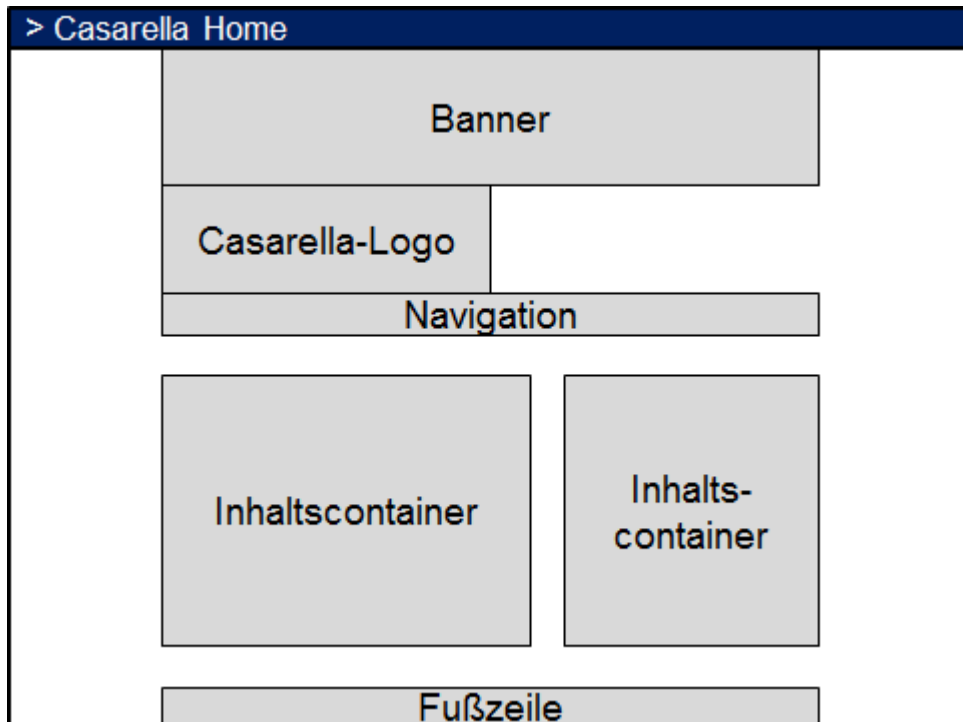


Abb. 71: Wireframe der Casarella-Web-Site

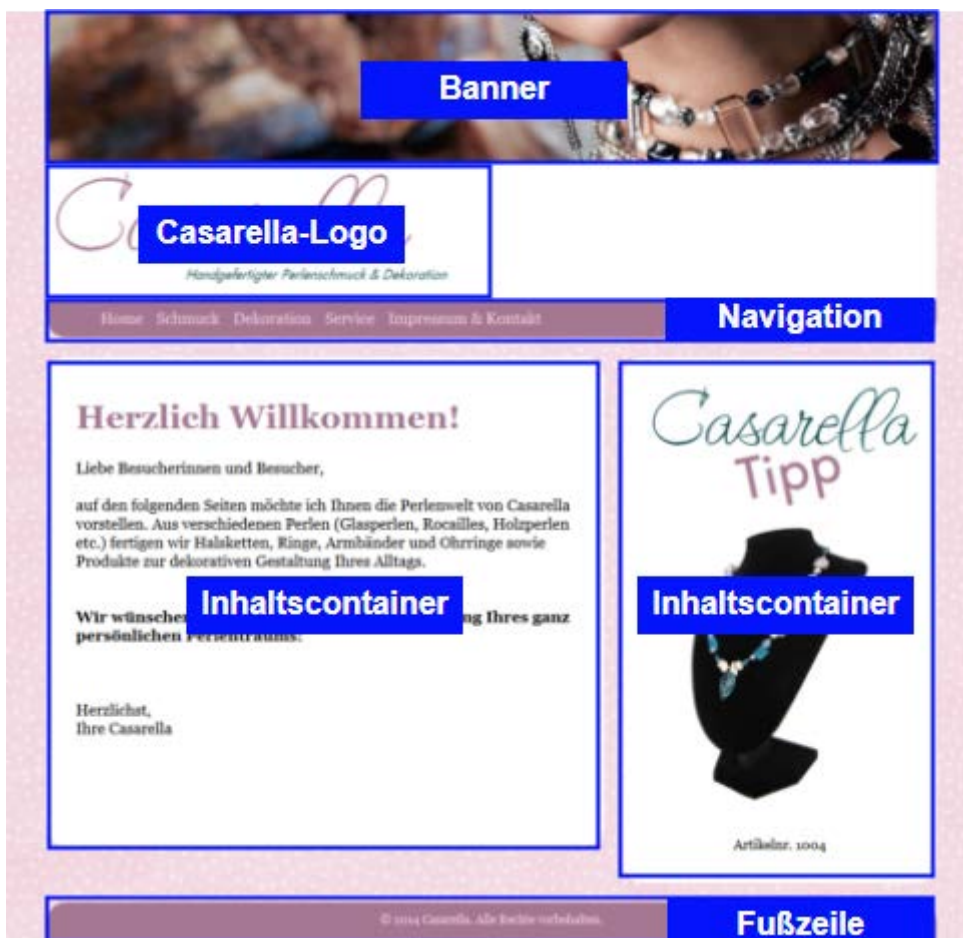


Abb. 72: Struktur der Casarella-Web-Site

CSS-Grid- Raster:

Abb. 73: CSS-Grid-Raster

```
css
Datei Bearbeiten Format Ansicht ?
.container{
display:grid;
grid-template-areas:
"top1 top2 top3"
"content1 content2 sidebar"
"foot1 foot2 foot3"
}
```

Abb. 74: CSS-Grid-template-areas

7.4 Die Raster-Bereiche im CSS-Grid-Layout

Herr Huber: „Schauen wir uns den CSS-Quelltext doch einmal genauer an:

Dem Element, welchem die Klasse "container" zugewiesen wurde, wird mit Hilfe der Angabe `display:grid;` mitgeteilt, dass CSS-Grid genutzt werden soll.

Die einzelnen Bereiche des CSS-Grids können mit Hilfe des Befehls `grid-template-areas` bei der Definition des Rasters im Elternelement benannt werden. Dazu erhält in jeder Zeile jede einzelne Zelle einen Namen.

Die einzelnen Namen werden mit Leerzeichen voneinander getrennt. Einzelne Zeilen stehen in Anführungsstrichen.

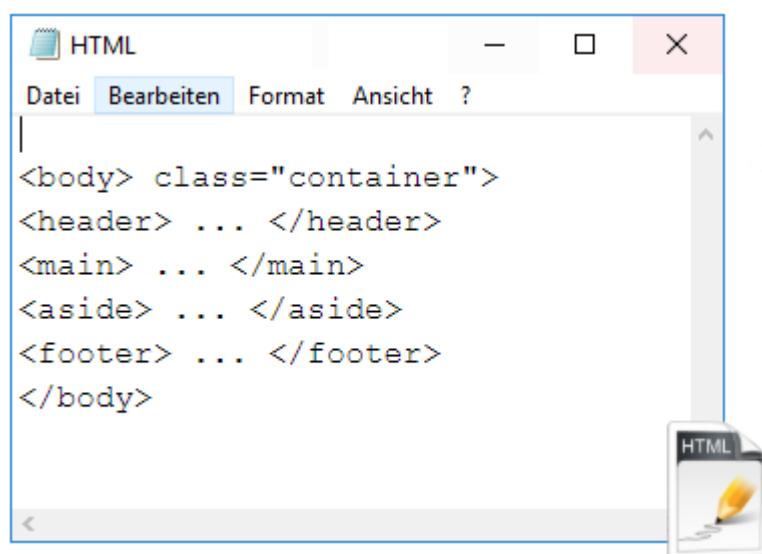
Durch eine sinnvolle Strukturierung im Code, kann so der Layout-Aufbau im Stylesheets »sichtbar« gemacht werden.“

7.5 CSS Grid verknüpfen mit dem HTML-Quelltext

Lin W. Lan: „Wow, das ist ja wirklich übersichtlich.“

Wie man CSS-Grid nun mit dem HTML-Quelltext verbindet, ist mir auch schon klar. Ich weise dem HTML-Container, welcher die Struktur erhalten soll, einfach die Klasse "container" zu.

Das teste ich einfach mal in einem neuen HTML-Dokument und weise dem <body> die Klasse container zu.



```
HTML
Datei Bearbeiten Format Ansicht ?
<body> class="container">
<header> ... </header>
<main> ... </main>
<aside> ... </aside>
<footer> ... </footer>
</body>
```

Abb. 75: HTML-Dokument mit der Klasse „container“

Lin W. Lan: „Hier ist ein Ausschnitt meines HTML-Dokuments. Herr Huber hat mir den Tipp gegeben, die geplanten Strukturbereiche im HTML-Elemente durch semantische div-Container zu hinterlegen.“

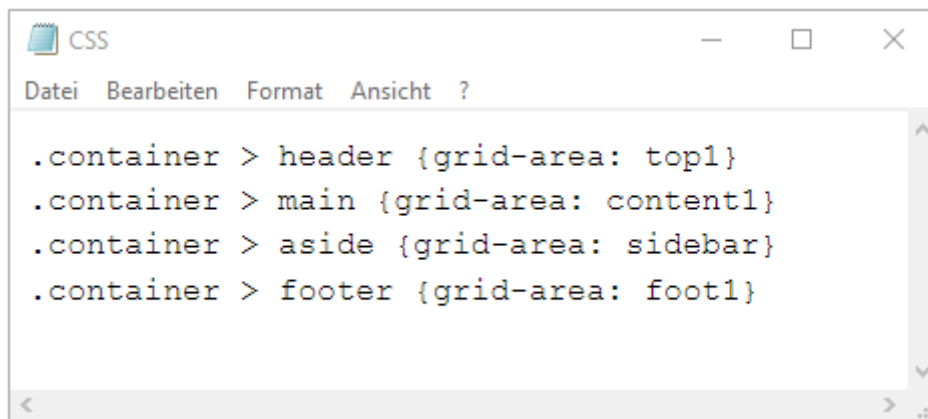
Klassische HTML-Seitenstrukturierung verbindet inhaltlich zusammengehörende Elemente durch <div>-Container. Moderne Seitenstruktur mit HTML 5 kennt semantische Container, also Container die entsprechend ihrer Bedeutung benannt sind. <header> und <footer> strukturieren die Elemente des Web-Seiten-Kopfes bzw. -Fußes, wie Logo, Titel, Navigation oder Impressum und Kontakt. Der Hauptinhalt einer Web-Seite wird <main> genannt, Randnotizen <aside>.

Das ist aber noch längst nicht alles. [Hier](#)– sind weitere semantische Container zur HTML5-Seitenstrukturierung beschrieben.

Lin W. Lan: „Ich vermute, die Container werden im nächsten Schritt mit den CSS-Grid-Areas verknüpft.“

7.6 CSS-Grid-Areas den Bereichen in HTML zuweisen

Herr Huber: „Ganz genau. Der HTML-Quelltext ist jetzt korrekt mit dem Stylesheet verbunden. Nun müssen noch die einzelnen CSS-Grid Areas mit den **Bereichen der Web-Seite** (header, main, aside und footer) verbunden werden. Die Zuweisung wird in CSS vorgenommen.“



```
.container > header {grid-area: top1}
.container > main {grid-area: content1}
.container > aside {grid-area: sidebar}
.container > footer {grid-area: foot1}
```

Abb. 76: Zuweisung der Web-Seiten-Bereiche

7.7 Zeilen und Spalten in CSS-Grid gestalten

Lin W. Lan: „Wow, das ist ja total einfach.“

Und jetzt kann ich jederzeit die Seitenstruktur durch neue Zuweisungen anpassen. Das Austauschen von Inhalten und Positionen war in CSS mit dem float-Befehl doch immer recht mühsam. Wie kann ich nun die einzelnen Bereiche meinen Wünschen entsprechend gestalten, also z. B. die Größe einer Zeile angeben?“

Herr Huber: „Die Größe von Zeilen und Spalten werden mit Hilfe der CSS-Eigenschaften **grid-template-columns** bzw. **grid-template-rows** gestaltet.“

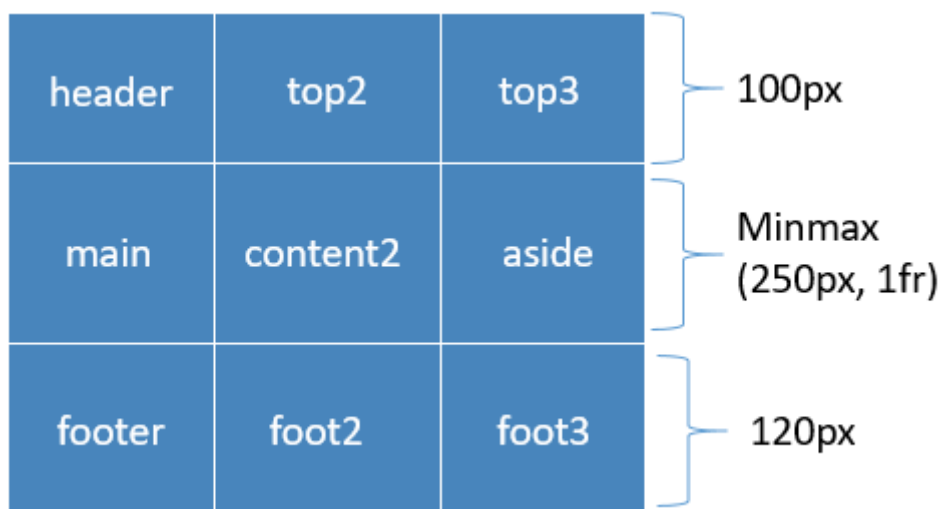


Abb. 77: Grid-template-rows

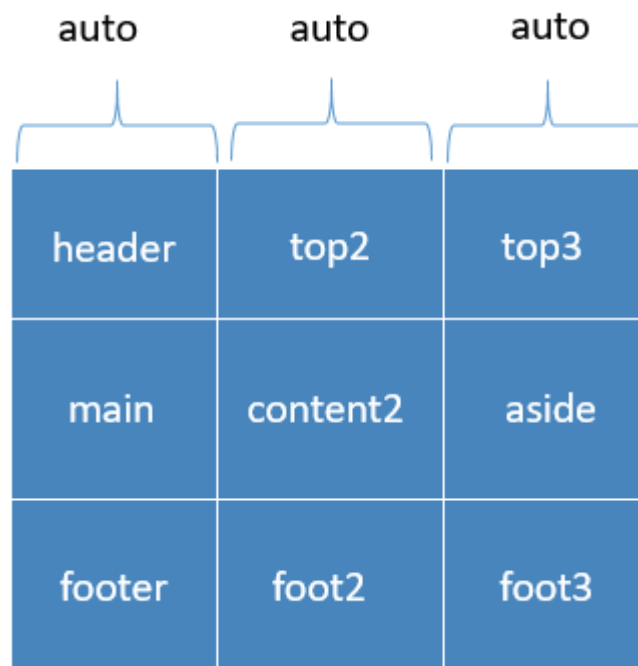


Abb. 78: Grid-template-columns

7.8 Relative Größenangaben mit „fraction“

Lin W. Lan: „Stopp!!! Das ging mir jetzt zu schnell.“

Was ist denn das für eine Größenangabe der zweiten Zeile? Den Befehl Minmax kenne ich schon, in den dahinterstehenden Klammern wird der Größenbereich angegeben. Also muss die zweite Zeile mindestens 250 Pixel hoch sein. Die maximale Größenabgabe von 1fr, die kenne ich aber noch nicht.

Bitte helfen Sie mir, Herr Huber.“

Herr Huber: „Ah gut, dass Sie es ansprechen. Noch so eine geniale Neuheit.“

Die Abkürzung "fr" steht für "fraction", zu Deutsch "Bruchteil". Diese Einheit ermöglicht die Füllung des restlichen verfügbaren Platzes auf der Web-Seite. Dieser hängt davon ab, über welches Endgerät bzw. mit welcher Auflösung die Web-Seite betrachtet wird.“



Monitor: 5K-Auflösung: 5.120 x 2.880 Pixel



Monitor: High Definition-Auflösung: 1.280 x 720 Pixel

Abb. 79: Monitorauflösung und die Einheit „fraction“

7.9 CSS-Grid-Layout

Herr Huber: „Der zugehörige Quelltext wird einfach im Stylesheet oberhalb der CSS-Grid-Areas eingetragen.“

Die Größe der Zeilen und Spalten ist beschrieben. Die Abstände zwischen den Zeilen und Spalten können an dieser Stelle ebenso festgelegt werden.“

```

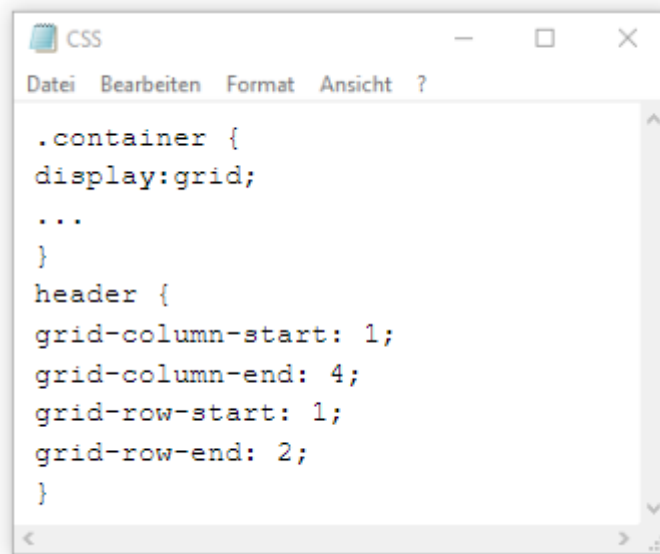
CSS
Datei Bearbeiten Format Ansicht ?
.container {
display:grid;
grid-template-columns:
auto auto auto;
grid-template-rows:
100px minmax(250px, 1fr) 120px;
grid-gap: 20px 10px;
grid-template-areas:
...
}

```

Abb. 80: Zeilen, Spalten und Abstände in CSS-Grid gestalten

7.10 Elemente im CSS-Grid positionieren

Lin W. Lan: „Das erstellte Raster hat noch einige inhaltsleere Bereiche. Header und footer sollen beispielsweise über die gesamte Breite der Web-Seite zu sehen sein. top2, top3, foot2 und foot3 werden also gar nicht benötigt.“



```
css
Datei Bearbeiten Format Ansicht ?
.container {
display:grid;
...
}
header {
grid-column-start: 1;
grid-column-end: 4;
grid-row-start: 1;
grid-row-end: 2;
}
```

Abb. 81: Elemente in CSS-Grid positionieren

Mit CSS-Grid ist es möglich, Elemente völlig frei im Raster zu positionieren. Mit Hilfe der Eigenschaften `grid-column` und `grid-row` wird einem Element mitgeteilt, an welcher Rasterposition es sich befinden soll.

Achtung: Die Positionierung erfolgt nicht über die Anzahl der Raster-Spalten (hier 3) sondern über die Rasterlinien (hier 4).

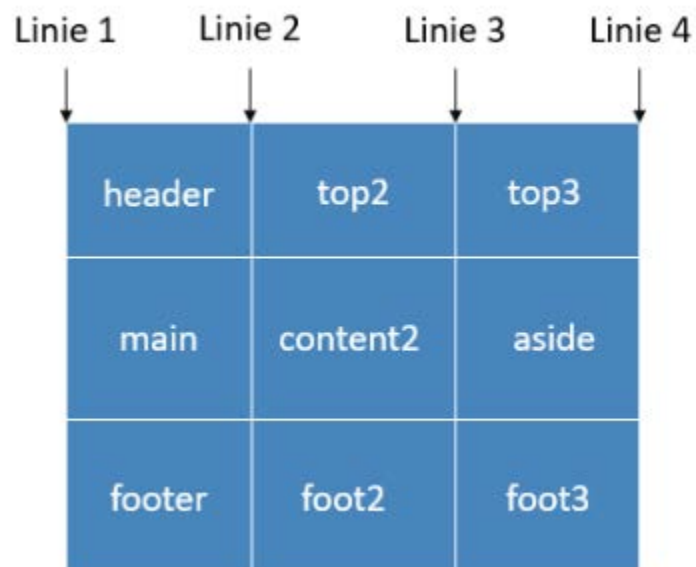
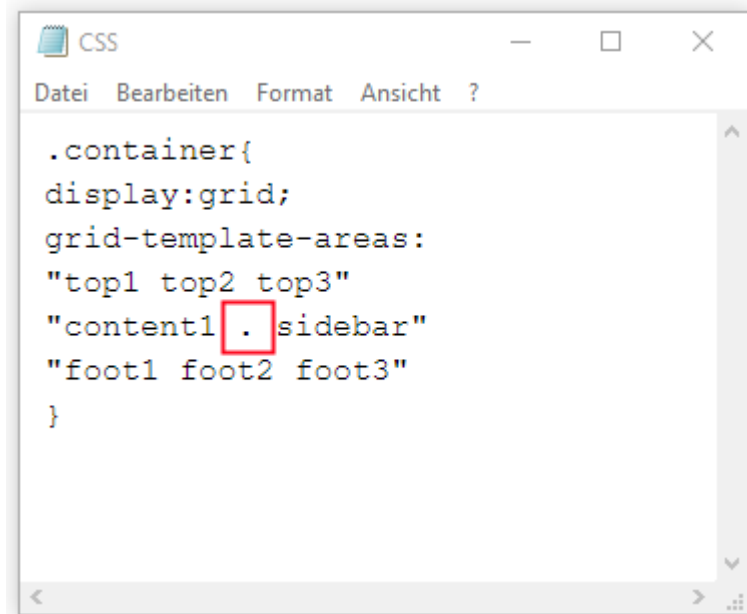


Abb. 82: Rasterlinien

7.11 Leere Bereiche im Raster kennzeichnen

Herr Huber: „Ebenso wie zuvor der header wird auch der footer positioniert. Zuletzt muss nur noch der überflüssige Bereich content2 entfernt werden.“

Wenn Bereiche im Raster leer bleiben sollen, wird dafür ein Punkt als Platzhalter gesetzt.“



```
.container{
display:grid;
grid-template-areas:
"top1 top2 top3"
"content1 . sidebar"
"foot1 foot2 foot3"
}
```

Abb. 83: Leere Bereiche in CSS-Grid

7.12 Abschluss

Lin W. Lan: „So, jetzt haben wir uns einen umfangreichen Überblick über CSS-Grids verschafft. Eine wirklich tolle neue Entwicklung.“

Mit dem Durcharbeiten dieses WBT haben Sie den CSS-Teil der WBT-Serie "HTML, CSS & JavaScript – Teil 2" abgeschlossen.

Weiter geht es im Folgenden mit Java Script, dort werden Sie lernen, wie Sie selber Java-Script-Quellcode entwickeln können.“

8 Rechnen mit JavaScript

8.1 Casarella braucht JavaScript

Lin W. Lan: „Hallo! Schön, dass Sie wieder bei uns sind, um Casarella bei dem Ausbau ihrer Web Site zu helfen.“

Casarella möchte, dass dem Kunden die Summe einer Bestellung in einem separaten Fenster angezeigt wird, bevor er die Bestellung abschließt. Er muss die Summe also nicht selbst errechnen.

Um dieser Bitte nachzukommen, brauchen wir JavaScript. Aber bevor wir richtig starten, erinnern wir uns daran, was JavaScript überhaupt ist.“

8.2 JavaScript – Wir erinnern uns

Lin W. Lan: „Wir haben die Web Site "Casarella" mit Hilfe von CSS schon deutlich attraktiver gestalten können. Um die Site noch interaktiver für den Nutzer gestalten zu können, verwenden wir JavaScript.“

JavaScript ist eine vollwertige Programmiersprache, das heißt, wir wenden uns wieder dem Programmieren zu um damit unser Web Design zu verbessern.

Bevor wir damit starten, schauen wir uns an, wie wir eine JavaScript-Datei in das vorhandene HTML-Dokument der Casarella Web Site einbinden können.“

8.3 JavaScript direkt in das HTML-Dokument schreiben

Der JavaScript-Quellcode wird direkt in das HTML-Dokument geschrieben. Eingeleitet wird JavaScript immer mit dem HTML-Tag: `<script>`.

Da statische Elemente schneller geladen werden als dynamische, sollte man JavaScript am Ende des `<body>` einbinden, so kann der Nutzer die dargestellten Inhalte schneller sehen.

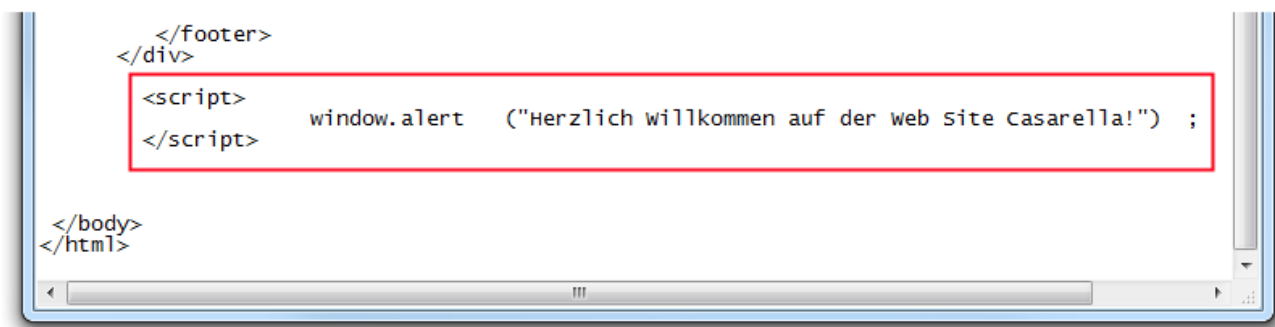


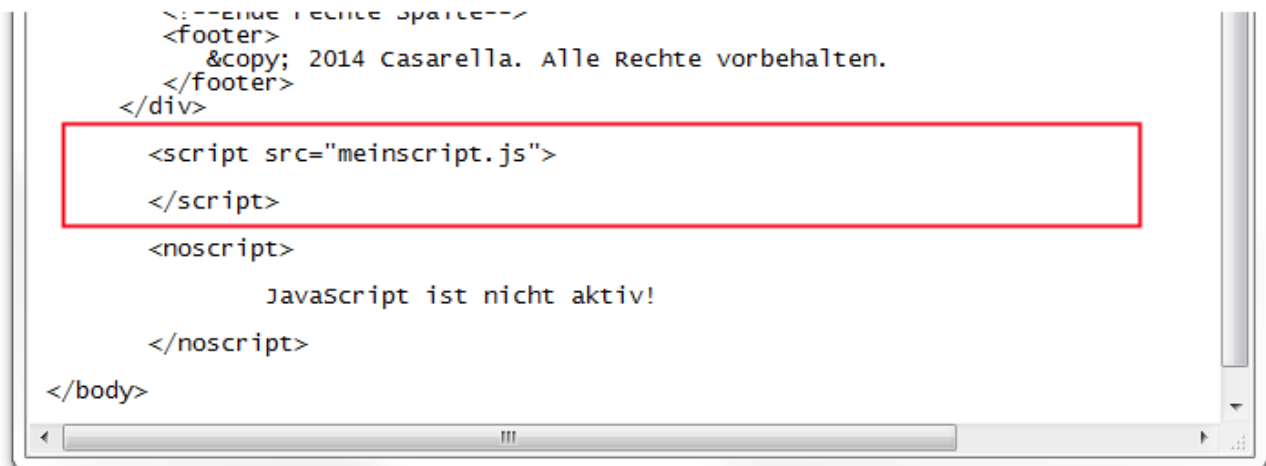
Abb. 84: Der HTML-Tag `<script>`

Hier wird JavaScript direkt in das HTML-Dokument eingebunden. Wie zuvor empfohlen, wird der Code ans Ende des HTML-Quellcodes positioniert.

8.4 JavaScript-Dateien in HTML-Dokumente einbinden

Lin W. Lan: „Natürlich kann auch eine externe JavaScript-Datei in das HTML-Dokument eingebunden werden. Dazu wird in den Script-Tag die Quelle der JavaScript-Datei geschrieben, wie unten im Beispiel "meinscript.js".

Da Casarella wünscht, die JavaScript Befehle direkt im HTML-Dokument auszuführen, wird diese Variante hier keine weitere Anwendung finden.“



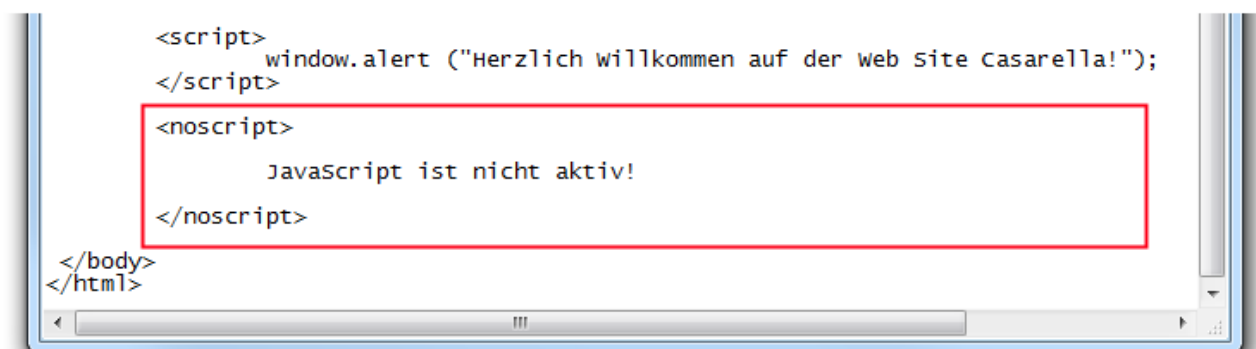
```
<!-- keine leere spalte -->
<footer>
  &copy; 2014 Casarella. Alle Rechte vorbehalten.
</footer>
</div>
<script src="meinscript.js">
</script>
<noscript>
  JavaScript ist nicht aktiv!
</noscript>
</body>
```

Abb. 85: Einbindung von JavaScript in HTML-Quelltext

8.5 JavaScript – Noscript

Lin W. Lan: „Damit im Browser des Nutzers die JavaScript-Elemente richtig ausgeführt werden, muss JavaScript im Browser aktiviert sein.

Wenn JavaScript nicht aktiviert ist, soll dem Nutzer eine entsprechende Meldung angezeigt werden. Dazu wird das noscript-Tag verwendet. Meistens steht im noscript-Tag nur: "JavaScript ist nicht aktiv".“



```
<script>
  window.alert ("Herzlich willkommen auf der web site Casarella!");
</script>
<noscript>
  JavaScript ist nicht aktiv!
</noscript>
</body>
</html>
```

Abb. 86: Der HTML-Tag <noscript>

8.6 Objekte und Methoden – Befehle in JavaScript

Peter Lan: „Sie denken sicher: "Das ist ja alles schön und gut, aber wie schreibe ich denn jetzt überhaupt einen Befehl in JavaScript?"

Hier sehen Sie wichtige und grundlegende Regeln von Javascript:“

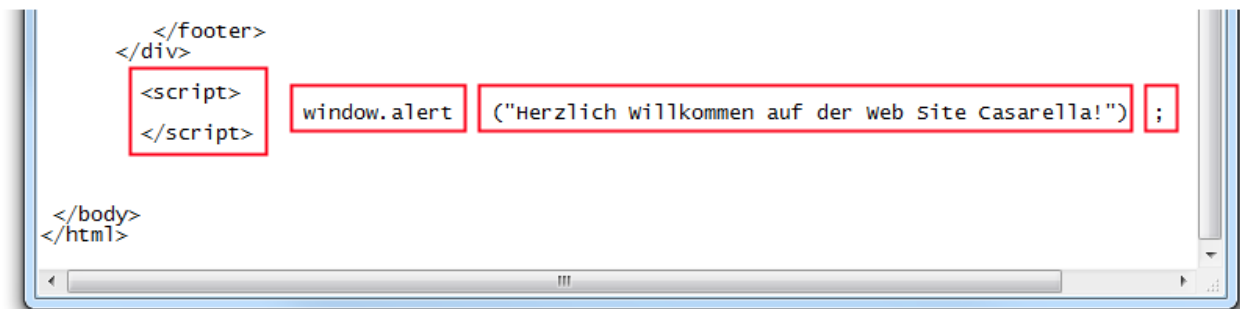


Abb. 87: Bestandteile eines JavaScript-Befehls

`<script> </script>`:

Hier sehen Sie einen kurzen JavaScript-Befehl. Durch das Script-Tag wird JavaScript in das HTML-Dokument eingebunden. Achten Sie darauf, das Script-Tag immer wieder zu schließen.

`window.alert`:

Im Script-Tag steht immer ein sogenanntes Objekt, (hier "window") welchem eine Methode folgt (hier "alert"). Objekt und Methode werden durch einen Punkt voneinander getrennt. "alert" öffnet im Browser ein Dialogfenster, welches eine Interaktion des Nutzers erfordert. Solange der Nutzer in diesem Dialogfenster nicht auf "OK" klickt, werden keine weiteren Befehle auf der Seite ausgeführt.

(„Herzlich Willkommen auf der Web Site Casarella!“)

In den Klammern hinter der Methode stehen die Parameter. In diesem Fall der Text des Dialogfensters. Anführungszeichen sind ebenfalls wichtig. Sie zeigen, dass es sich um einen "string", also eine Zeichenkette handelt. Zahlen und Variablen werden ohne Anführungszeichen in die Klammern geschrieben.

Semikolon (;)

Jeder JavaScript-Befehl wird durch ein Semikolon abgeschlossen.

8.7 Objekte und Methode in JavaScript – "window.alert"

Wie Sie auf der vorigen Seite schon gesehen haben, gibt es bei JavaScript die sogenannten Funktionsaufrufe. Hiermit erklärt man dem Browser, wie er die dargestellten Parameter anzeigen soll. Durch die Methode "alert" werden die Parameter in einem Dialogfenster angezeigt.



Abb. 88: Dialogfenster auf der Web Site Casarella

Dies ist das Dialogfenster auf der Web-Seite. Der Hintergrund wird ausgegraut und der Nutzer muss mit der Seite interagieren. Durch einen Klick auf "OK" kann der Nutzer die Seite weiter bedienen.

"window.alert" wird in das Script vor die Klammern des eigentlichen Befehls geschrieben. So weiß der Browser, dass er JavaScript interpretieren muss. Für den Nutzer der Web-Seite ist diese Information aber nicht zu sehen.

8.8 Objekte und Methode in JavaScript – "document.write"

Eine andere Form der Informationsdarstellung auf einer Web Site, ist, die Parameter einer Methode direkt auf der Seite selbst anzuzeigen.



Abb. 89: Inhalte dargestellt im Browser durch document.write

Durch "document.write" wird der JavaScript-Befehl direkt als Text auf der Web-Seite dargestellt.

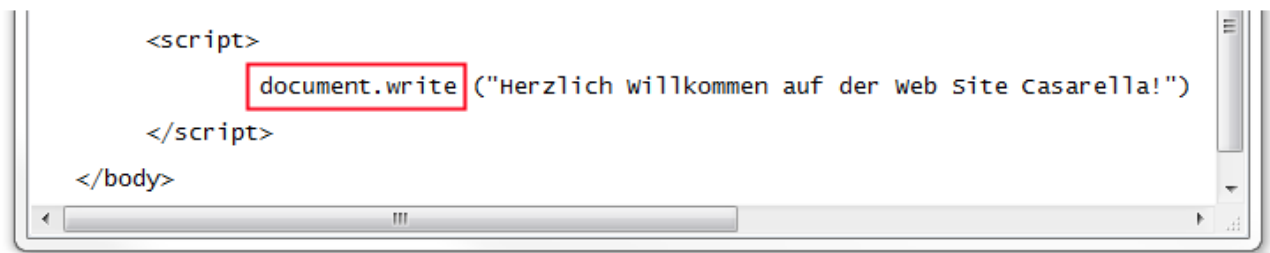


Abb. 90: Document.write im Quelltext

"document.write" wird gefolgt von einem Anzeigetext. Im Gegensatz zu "window.alert" wird keine Interaktion des Users gefordert.

8.9 Datentypen

Beate Ether-Net, Datenbeauftragte: „Für die Darstellung von Texten wie "Herzlichen Willkommen auf der Web Site Casarella" haben wir zwar JavaScript verwendet, üblicherweise nimmt man dafür aber HTML.

JavaScript kennt aber dafür neben Texten (genannt: Strings oder Zeichenfolgen) weitere Datentypen:“

Strings:

Strings sind Folgen von beliebigen Buchstaben, Ziffern und Sonderzeichen. Alle Zeichen werden so dargestellt, wie vom Programmierer geschrieben.

Strings werden auch "Zeichenfolgen" genannt und werden umgeben von Anführungszeichen, wie "Hello" oder 'World'. Ein Beispiel für Strings ist: 'JavaScript ist super!'

Numbers:

Der Datentyp numbers kann nur Zahlen aufnehmen. Außerdem benötigt man hier keine Anführungszeichen. Achten Sie darauf, dass JavaScript die internationale Punktation verwendet. Daher sind Dezimalzahlen mit Punkt und nicht wie im Deutschen mit Komma darzustellen.

Beispiele: 3, 2017 und 70.999

Booleans:

Boolean ist ein Datentyp, der nur aus den Zuständen wahr oder falsch besteht. Dafür werden als Parameter die englischen Begriffe true und false verwendet.

Auch sie werden ohne Anführungszeichen geschrieben und finden beispielsweise bei Bedingungen Anwendung.

Zum Beispiel: if "Temperatur > 20°C", then...ergibt im Sommer häufig den Boolean "true" und im Winter meist "false".

Null:

"Null" ist in JavaScript ein festgelegter, leerer Wert. Es wird immer dann verwendet, wenn zum Beispiel Variablen leer sind und das auch sein sollen. Es wird also nicht "undefined" angezeigt, sondern eben "null".

Sobald einer Variable dann ein Wert zugeordnet wird, ändert sich der Datentyp entsprechend.

Null wird im Code auch ohne Anführungszeichen verwendet, da es ein festgelegter Begriff in JavaScript ist.

8.10 JavaScript kann rechnen

Lin W. Lan: „Ein Vorteil von JavaScript ist: Sie können damit rechnen! JavaScript beherrscht die vier Grundrechenarten und dies ist auf der Web Site von Casarella nützlich, um den Rechnungsbetrag eines Kunden zu ermitteln.

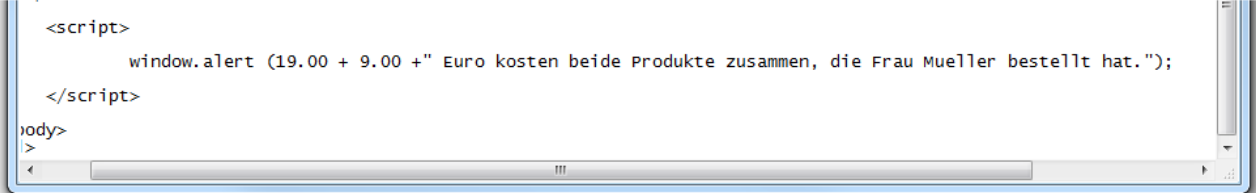
Jetzt wo wir wissen, wie JavaScript eingebunden wird, können wir starten die gewünschten Rechenoperationen auf der Web Site Casarella einzurichten. Um Rechnen mit JavaScript kennen zu lernen, starten wir mit einem einfachen Beispiel.

Danach sind Sie dran - Los geht's!“

8.11 Rechnen mit JavaScript

Die Kundin Müller hat sich eine Kette und ein Armband bei Casarella ausgesucht.

Mit der Alert-Methode können Sie in JavaScript ganz einfach rechnen. Die Rechenoperation kann direkt hinter die Alert-Methode geschrieben werden. Das Ergebnis wird dem Nutzer dann mit einem Dialogfenster im Browser mitgeteilt.



```
<script>
    window.alert (19.00 + 9.00 +" Euro kosten beide Produkte zusammen, die Frau Mueller bestellt hat.");
</script>
body>
>
```

Abb. 91: Window.alert im Texteditor

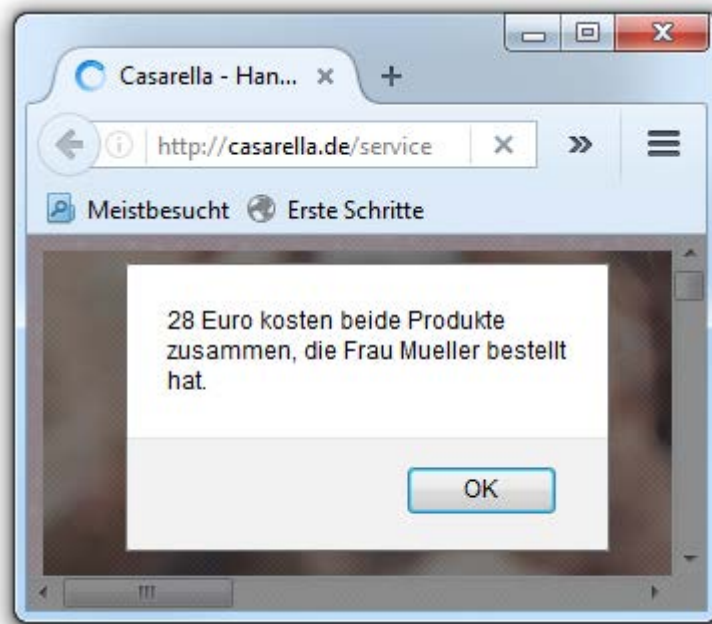


Abb. 92: Window.alert im Browser

8.12 Rechnen mit JavaScript – Teil 2

Lin W. Lan: „Es wäre sinnvoller, wenn das Ergebnis der Berechnung nicht als Dialogfenster, sondern direkt im Text erscheinen würde. Dies ermöglicht "document.write". Daher ersetzen wir nun "alert" durch "document.write". Der Befehl wird dadurch direkt im Dokument ausgeführt und es öffnet sich kein Dialogfenster.“

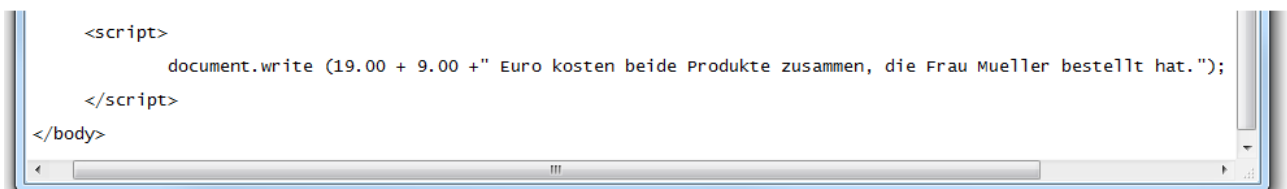


Abb. 93: Document.write im Texteditor

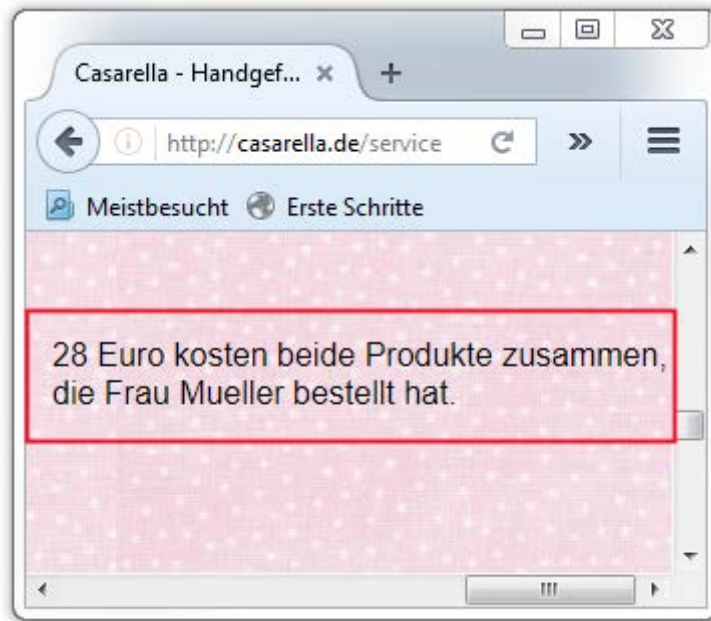


Abb. 94: Document.write im Browser

8.13 Übung – Rechnen mit JavaScript

Lin W. Lan: „Nun sind Sie an der Reihe! Frau Meier hat jetzt 20 Ringe à 7,50€ gekauft. Hinzu kommen Versandkosten von 4,50€ und da sie per Direktüberweisung zahlt, erhält sie 2% Skonto auf den Preis inklusive Versandkosten.“

Casarella benötigt Ihre Hilfe bei der Berechnung mit JavaScript. Klicken Sie auf die Diskette, um den Quellcode der Web Site von Casarella zu downloaden. Die Berechnungen können Sie direkt im Skript ausführen. Viel Erfolg!“

Lösung:

```
<script>
document.write (0.98 *(20*7.5)+4.5+" Euro lautet die Gesamtsumme
der bestellten Produkte von Frau Meier.");
</script>
```


8.14 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Der Aktionsbefehl _____ führt die JavaScript-Befehle direkt auf der Web-Seite aus.		
2	JavaScript als Programmiersprache ist zuständig für die farbliche Gestaltung einer Web Site.		
	Richtig		
	Falsch		
3	JavaScript kennt die vier Grundrechenarten.		
	Richtig		
	Falsch		
4	Durch die "alert"-Funktion wird ein Dialogfenster geöffnet, auf welches der Nutzer reagieren muss. Solange dies nicht geschieht, lädt die Seite keine weiteren Inhalte.		
	Richtig		
	Falsch		
5	Wählen Sie die richtige Antwort aus.		
	Die Anführungszeichen in der Klammer eines JavaScript Befehls zeigen den Verlauf einer Zeichenkette.		
	Ein Befehl wird immer mit einem Semikolon beendet.		
	Der noscript-Tag kommt bei JavaScript nur selten zur Anwendung.		
6	Bei dem Befehl "document.write" wird das Ergebnis einer Funktion direkt im Text der Seite angezeigt.		
	Richtig		
	Falsch		
7	JavaScript...		
	..ist eine vollwertige Programmiersprache.		
	...wird zur Programmierung komplexer Programme angewendet.		
	...muss direkt im HTML-Dokument ausgeführt werden.		

Tab. 5: Abschlusstest WBT 08

9 Variablen in JavaScript

9.1 Casarella möchte Größentabellen erweitern

Lin W. Lan: „Hallo! Schön, dass Sie wieder bei uns sind.

Casarella hat für heute eine Bitte an uns: Sie möchte internationalen Kunden die Ringgrößen in Inches bereitstellen. Die Umrechnung jeder einzelnen Ringgröße ist aufwendig. Der Einsatz von JavaScript mit Variablen erleichtert uns die Umrechnung der einzelnen Ringgrößen.

Heute schauen wir uns daher an, wie man in JavaScript mit Variablen umgeht.“

9.2 Was sind Variablen in JavaScript?

Lin W. Lan: „Variablen bieten die Möglichkeit, sie mit einem bestimmten Inhalt zu befüllen. Durch die Angabe des Variablennamens kann man jederzeit auf die Inhalte der Variable zugreifen.

Diese Wiederverwendbarkeit erspart beim Programmieren eine Menge Schreibarbeit.

Hier finden Sie einige Beispiele für Variablen in JavaScript. Keine Panik, ich erkläre Ihnen noch alles, was Sie dazu wissen müssen.“

```
<script>

var skonto = 0.03;
var preisRing1 = 14.25*skonto
var preisKette2 = 40.99*skonto

</script>

//Preis ring1 = 13.83
//Preis Kette2 = 39.79
```

Abb. 95: Variablen in JavaScript

Variablen erhalten einen **Namen** (in Abb. 96: var skonto) der die Variable eindeutig benennt. Durch die **Angabe dieses Namens** (z. B. skonto innerhalb der Variablenbezeichnung var preisRing1 = 14,25*skonto) wird auf den Inhalt der Variable zugegriffen. Sie werden drei Arten kennen lernen, Variablen zu kennzeichnen: "var", "let" und "const".

9.3 Die const-Variable

„const“ (kurz für constant) ist ein Schlüsselwort in JavaScript. „const“ steht also für eine Konstante. In einem JavaScript-Programm kann einer const-Variablen genau einmal ein Wert zugeordnet werden. Diese Variable soll sich im gesamten Programm nicht verändern.

Beispiele dafür sind Werte, die auch in der realen Welt konstant sind: Die Zahl Pi, der Umrechnungsfaktor von Fahrenheit in Celsius und die Anzahl der Tage pro Woche.

```
<script>
  const myProject = `Casarella`;
</script>
```

Const

Mit der Einleitung "const" weiß JavaScript, dass nun eine Variable mit konstantem Inhalt folgt. Anschließend folgt der Name der Variablen mit dem zugeordneten Wert.

myProject

Beachten Sie, dass im Namen der Variablen keine Leerzeichen stehen. Außerdem fängt der Name einer Variablen mit einem kleinen Buchstaben an, dafür wird jedes neue Wort mit einem Großbuchstaben begonnen. Diese Schreibweise ist eine Konvention, die man „camelCase“ nennt, weil die Klein- und Großbuchstaben Kamelhöckern ähneln. Sie ist keine Pflicht und Sie können beispielsweise auch mit underscore arbeiten.

Gleichheitszeichen (=)

Das Gleichheitszeichen stellt den Zuweisungsoperator dar. Er weist der Variablen „myProject“ den Wert „Casarella“ zu.

`Casarella`

Einer Variablen können beliebige Zeichenfolgen (strings) oder Zahlen (ohne Anführungszeichen) oder Zustände (booleans) zugeordnet werden:

```
const myAge = 24;
const iLoveJavaScript = true;
```

9.4 Was sind Variablen in JavaScript?

Lin W. Lan: „Die zweite Art, in JavaScript eine Variable zu kennzeichnen, ist „let“. Im Gegensatz zu einer const-Variablen kann die let-Variable innerhalb eines JavaScript-Programms verändert werden.“

Let-Variablen sind also immer dann zu verwenden, wenn im Laufe eines JavaScript-Programms der Wert veränderbar sein soll.“

Beachten Sie: Wenn Sie einer Variable keinen Wert zuordnen, gibt JavaScript automatisch den Wert “undefined” für diese Variable aus.

9.5 Die var-Variable

Die Variable, die mit „var“ gekennzeichnet wird, funktioniert etwas anders, als die beiden vorherigen. „var“ kann sozusagen als Allzweckwerkzeug verstanden werden, da JavaScript hier nicht unterscheidet, ob die Variable innerhalb des Programms nochmal verändert wird, oder konstant bleiben soll.

Die var-Variable wird von jedem aktuellen Web Browser unterstützt.

```
<script>
    Var DAYS_OF_THE_WEEK = "7";
    ...
    Var myJewellery = "necklace";
</script>
```

Um dem Nachteil der geringeren Übersichtlichkeit entgegenzuwirken, wird bei der var-Variablen mit konstantem Wert (DAYS_OF_THE_WEEK) nicht der camelCase verwendet.

Stattdessen werden alle Buchstaben großgeschrieben und die Wörter durch Unterstriche voneinander getrennt. Auch dies ist lediglich eine Übereinkunft, um den Quellcode verständlicher zu gestalten.

Bei einer var-Variable, dessen Inhalt im Laufe des Programms noch verändert werden können soll, wird wie gewohnt die camelCase-Konvention verwendet.

9.6 Die richtige Variable für Casarella

Lin W. Lan: „Welche ist nun die richtige Variable für die Web Site von Casarella?“

Um Ablaufprobleme von JavaScript-Programmen zu vermeiden, verwenden wir generell die var-Variable.“

9.7 Variablen können auch mit Zahlen befüllt werden

Lin W. Lan: „Natürlich können wir mit Variablen in JavaScript auch Rechenoperationen durchführen. Dazu teilen wir einer Variable keine Zeichenfolge als Wert zu, sondern Zahlen.“

Da bei Rechenoperationen häufig Kommazahlen benötigt werden, ist es von Vorteil, dass JavaScript auch mit Dezimalzahlen rechnen kann. Um das Ergebnis einer Rechenoperation zu runden wird der Befehl toFixed(n) genutzt.

```
<script>
    var x = 30023525;
```

```
Document.write(x*4);  
</script>
```

Im Web Browser wird angezeigt: 120094100

Beachten Sie: Zahlen in JavaScript lassen sich nur in dem Wertebereich von -9.007.199.254.740.992 bis 9.007.199.254.740.992 darstellen. Außerdem ist die Anzahl der Nachkommastellen begrenzt. Beide Eigenschaften werden Sie in der Praxis kaum betreffen, trotzdem sollten Sie sie einmal gehört haben.

```
<script>  
var y = 4.2344533*3;  
Document.write (y.toFixed(2));  
</script>
```

Im Web Browser wird angezeigt: 12.70

Wir erinnern uns: Diese Schreibweise mit Objekt und Methode kennen wir schon aus dem WBT "Rechnen mit JavaScript".

Die Zwei in der Klammer bedeutet, dass das Ergebnis auf zwei Stellen gerundet wird.

9.8 Übung – Variable schreiben

Lin W. Lan: „Erstellen Sie die Variable `favoriteMeal` und geben Sie ihr als Inhalt Ihr Lieblingsessen. Nutzen Sie anschließend ein passendes Objekt mit dazugehöriger Methode, um in die JavaScript-Datei zu schreiben: 'Mein Lieblingsessen: Sushi', aber ersetzen Sie 'Sushi' mit Ihrer Variablen.

Ordnen Sie Ihrer Variablen anschließend noch drei andere Werte zu und testen Sie Ihre Ausgabe jeweils im Web Browser.

```
<script>  
var favoriteMeal = "sushi";  
document.write('My favorite meal:' + favoriteMeal + '.');  
</script>
```

Im Web Browser wird angezeigt: My favorite meal: sushi.

9.9 Übung – Umrechnung der Ringgrößen

Erstellen Sie bitte Variable für Casarella, die durch alle Browser interpretiert werden kann. Sie soll den Namen `cmToInch` tragen und sie soll Casarella ermöglichen, jede cm-Angabe im Handumdrehen in eine inch-Angabe zu verwandeln.

Suchen Sie dafür den Faktor, den man benötigt, um 1 cm in Inch umzurechnen. Rechnen Sie anschließend mit Ihrer Variable 5cm in Inch um. Achten Sie bitte darauf, dass das Ergebnis auf zwei Nachkommastellen gerundet wird.

```
<script>
var cmToInch = 0.393701
document.write(cmToInch*5)
var cmToInches5 = (cmToInch*5)
document.write(cmToInches5.toFixed(2));
</script>
```

**Im Web Browser wird angezeigt: 1.97
(Ohne Runden: 1.9685050000000002)**

9.10 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Wiederholen Sie die Übung der vorigen Seite, aber rechnen Sie nun 7.9 cm in Inch um. Welches Ergebnis, zeigt Ihnen Ihr Web Browser gerundet auf zwei Nachkommastellen an?		
	4.9		
	3.11		
	3.42		
2	Die Variable _____ ist von jedem Browser einwandfrei interpretierbar.		
3	JavaScript kann nur mit ganzen Zahlen rechnen.		
	Richtig		
	Falsch		
4	Der Einsatz von Variablen in einer Programmiersprache dient dazu, dem Programmierer unnötige Arbeit zu ersparen, da man auf Variablen immer wieder zugreifen kann.		
	Richtig		
	Falsch		
5	Variablen können nur mit "strings" befüllt werden, nicht aber mit "numbers".		
	Richtig		
	Falsch		
6	Wählen Sie die richtige Antwort aus.		
	"const" kennzeichnet eine konstante Variable.		
	Für Casarella eignet sich die Variable "var" am Besten.		
	Die Nachkommastellen sind in JavaScript begrenzt.		
7	Eine Schreibweise, mit der man Variablen kennzeichnen kann, heißt "camelCase".		
	Richtig		
	Falsch		

Tab. 6: Abschlusstest WBT 09

10 Interaktivität und Fehleranalyse

10.1 Einleitung – Funktionen für die Casarella Web-Site

Lin W. Lan: „Hallo! Schön, dass Sie wieder bei uns sind. Heute schauen wir uns in JavaScript an, was Funktionen sind und wie sie funktionieren. Sie werden lernen, wie Funktionen geschrieben werden, um zum Beispiel den Rechnungsbetrag des aktuellen Warenkorbs im Web Shop von JavaScript berechnen zu lassen. Wir beginnen auf der nächsten Seite mit einer Anwendung einer Funktion in JavaScript in Form eines Videos. Viel Spaß!“

[Es folgt ein Video über ein Beispiel einer JavaScript-Funktion anhand eines Warenkorbs auf der Web Site der Zalando GmbH.]

10.2 Funktionen in JavaScript

- **1. Funktion schreiben:** Der Programmierer schreibt eine Funktion, die den Perlenbedarf des Jahres 2018 berechnet.
- **2. Funktion ausführen:** JavaScript führt die Berechnung der Funktion durch.
- **3. Funktionsergebnis darstellen:** JavaScript stellt das Ergebnis mit Hilfe von HTML-Quelltext im Browser dar.

10.3 Ging das zu schnell?

Lin W. Lan: „Ich habe das Gefühl das ging Ihnen etwas zu schnell?! Kein Problem! Es ist noch kein Meister vom Himmel gefallen. Auf den nächsten Seiten schauen wir uns die drei Schritte noch einmal genauer an. Lassen Sie uns mit dem Aufbau einer Funktion in JavaScript beginnen.“

10.4 Aufbau einer Funktion in JavaScript

Casarella beobachtet seit einer Woche die Kundenbestellungen. Darauf basierend soll der Perlenbedarf für das Jahr 2018 ermittelt werden. Dies erfolgt über eine Berechnung im Intranet des Betriebs.


```

<script>
function bedarfPerlen2018 ()
{
var TAGE_IM_JAHR = 270;
var perlenProKette = 50;
var perlenProArmband = 25;
var kettenProTag = 2;
var armbaenderProTag = 4;
var bedarfPerlenProTag = kettenProTag *
perlenProKette + armbaenderProTag *
perlenProArmband
var perlen2018 = bedarfPerlenProTag * TAGE_IM_JAHR;
}

bedarfPerlen2018 ();
</script>

```

Abb. 96: Aufbau einer Funktion in JavaScript

- 1.: Das Schlüsselwort `function` leitet eine Funktion ein. Die Funktion wird benannt (hier mit `bedarfPerlen2018`). Es folgt ein Klammerspaar und eine geöffnete geschwungene Klammer. Die geschwungene Klammer öffnet einen „Code-Block“. Hierbei spielt es keine Rolle, in welcher Zeile die geschwungene Klammer steht. Alles, was innerhalb des Code-Block steht, gehört zur Funktion `bedarfPerlen2018`.
- 2.: Innerhalb der Funktion werden Variablen definiert. Mit diesen Variablen werden mit den vier Grundrechenarten Berechnungen durchgeführt. Durch die geschlossene geschwungene Klammer weiß JavaScript, dass der Code-Block, und damit auch die Funktion, zu Ende ist.
- 3.: Wenn Sie die Funktion geschrieben haben, kann die Funktion durch Nennung ihres Namens ausgeführt werden. Wie das Ergebnis der Funktion im Browser dargestellt wird, lernen Sie auf der nächsten Seite. ACHTUNG: Dass die Funktion ausgeführt wird, heißt nur, dass JavaScript sie berechnet. Dem Nutzer der Web Site wird hier noch nichts dargestellt.

10.5 Welches Verfahren eignet sich zur Darstellung der Funktion?

Der Code unserer Funktion ist für den Anwender im Browser nicht sichtbar. Wir kennen bisher die JavaScript-Darstellungsbefehle `"window.alert"` und `"doc-write"`. Lassen Sie uns zunächst einmal prüfen, ob wir diese Befehle zur Darstellung des Funktionsergebnisses verwenden können.

- `window.alert`: Dem Objekt **"window"** folgt die Methode **"alert"**. Objekt und Methode werden durch einen Punkt voneinander getrennt. `"alert"` öffnet im Browser ein Dialogfenster, welches eine Interaktion des Nutzers erfordert. Dieser Befehl kann zwar das Funktionsergebnis darstellen, ist aber nicht benutzerfreundlich. Eine bessere Möglichkeit der Darstellung bieten JavaScript IDs.
- `document.write`: `"document.write"` schreibt man direkt in den HTML-Quelltext. Im Gegensatz zu `"window.alert"` wird keine Interaktion des Users gefordert. Der geschriebene Code wird direkt an Ort des Aufführens dargestellt und kann keine Funktionsergebnisse ausgeben. Dieses Verfahren ist somit ebenfalls nicht geeignet.

- Wie Sie sehen, sind beide uns bekannten Verfahren der Code-Darstellung hier unpassend. Auf der nächsten Seite lernen Sie die geeignete Lösung kennen.

10.6 Darstellung von Funktionen durch IDs

Die Darstellung einer Funktion im Browser erfolgt mit IDs. Das Objekt, welches wir verwenden ist document. Die passende Methode lautet .getElementById(). Hierfür ist es nötig, dass dem darzustellenden Element im Vorhinein eine genaue ID zugeordnet wird. Die Funktion bedarfPerlen2018() berechnet den Perlenbedarf für das Jahr 2018. Ergebnis der Funktion steht in der Variable perlen2018. Der Wert der Variable soll auf der Web Site dargestellt werden.

Hier ein Beispiel mit unserer Funktion bedarfPerlen2018():

```
<script>
function bedarfPerlen2018 () {
...
document.getElementById("ausgabe").innerHTML=perlen2018;
}

bedarfPerlen2018();
</script>

//Jetzt den Perlenbedarf mit einer ID darstellen:
<span id="ausgabe"></span>
```

Abb. 97: Darstellung von Funktionen

10.7 Ergebnisdarstellung der JavaScript-Funktion

Lin W. Lan: „Super! Wir haben es geschafft und unser Funktionsergebnis wird im Browser angezeigt. Casarella muss also für 2018 mit einem Perlenbedarf von 54.000Stück rechnen. Da diese Funktion im Intranet dargestellt wird, ist es nicht nötig, das Funktionsergebnis mit CSS ansehnlicher zu gestalten. Ein Beispiel für eine JavaScript-Funktion auf einer umsatzrelevanten Web Site haben Sie bereits in unserem kurzen Video kennengelernt.“

10.8 Übung – Funktion schreiben

Erstellen Sie die Funktion `bedarfEdelsteine2018`. Nutzen Sie die bekannten Angaben der Funktion `bedarfPerlen2018`. Lediglich die Anzahl der Edelsteine pro Kette und Armband weichen ab.

Rechnen Sie mit:

- a) 2 Edelsteinen pro Kette und 1 Edelstein pro Armband und
- b) mit 4 Edelsteinen pro Kette und 2 Edelsteinen pro Armband.

Lassen Sie sich Ihre Funktion wie gelernt im Browser darstellen.

Lösung:

```
<script>
function bedarfEdelsteine2018 ()
{
  var TAGE_IM_JAHR = 270;
  var edelsteineProKette = 2(4);
  var edelsteineProArmband = 1(2);
  var kettenProTag = 2;
  var armbaenderProTag = 4;
  var bedarfEdelsteineProTag =kettenProTag*
  edelsteineProKette + armbaenderProTag *
  edelsteineProArmband
  var edelsteine2018 = bedarfEdelsteineProTag *
  TAGE_IM_JAhr;
}

bedarfEdelsteine2018();
</script>
```

Abb. 98: Lösung der Übung

Im Web Browser wird angezeigt:

- a) „2160“
- b) „4320“

10.9 Fehleranalyse in JavaScript

Peter Lan: „Fehler im Quelltext von JavaScript werden im Browser nicht angezeigt. Der Programmierer muss selbst aktiv werden. Die sogenannte console (oder Konsole) im Browser unterstützt jedoch bei der Fehleranalyse, indem sie Fehlermeldungen erzeugt. Bei gängigen Browsern wie Firefox oder Google Chrome ist die console unter den Entwicklertools zu finden.“

Folgender Quelltext wird auf den nächsten beiden Seiten in Chrome und Firefox analysiert:

```

<script>
document.write("Casarella braucht 2018 viele Perlen.")

function bedarfPerlen2018 ()
{
var TAGE_IM_JAHR = 270;
var perlenProKette = 50;
var perlenProArmband = 25;
var kettenProTag = 2;
var armbaenderProTag = 4;
var bedarfPerlenProTag = kettenProMonat*perlenProKette + armbaenderProTag * perlenProArmband
var perlen2018 = bedarfPerlenProTag * 270;
}

bedarfPerlen2018 ();

function bedarfEdelsteine2018 (){
var TAGE_IM_JAHR = 270;
var edelsteineProKette = 2(4);
var edelsteineProArmband = 1(2);
var kettenProTag = 2;
var armbaenderProTag = 4;
var bedarfEdelsteineProTag =kettenProTag*
edelsteineProKette + armbaenderProTag * edelsteineProArmband
var edelsteine2018 = bedarfEdelsteineProTag * 270;
}

bedarfEdelsteine2018 ();
document.getElementById("ausgabe")

</script>

```

Abb. 99: Quelltext zur Fehleranalyse

10.10 Fehleranalyse mit Google Chrome

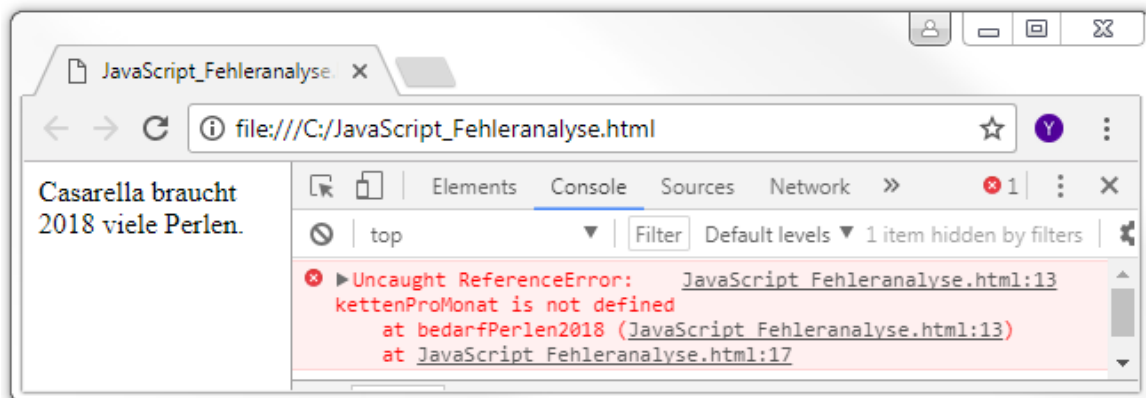


Abb. 100: Fehleranalyse mit Google Chrome

- „x“: Hier sieht der Programmierer, dass ein neuer Fehler beginnt. In der Symbolleiste sieht man neben dem roten Kreuz, wie viele Fehler das JavaScript-Dokument insgesamt aufweist.
- „Uncaught Reference Error“: Hier zeigt die console die Art des Fehlers: Uncaught Reference bedeutet immer, dass eine Variable gar nicht oder nicht richtig deklariert wurde.
- „kettenProMonat is not defined“: kettenProMonat ist die Variable, die nicht richtig oder gar nicht deklariert wurde. "Not defined" meint, dass die Variable gar nicht deklariert wurde.

- **„bedarfPerlen2018“:** Hier wird der fehlerhafte Code-Block aufgezeigt. Im vorliegenden Beispiel ist es die Funktion "bedarfPerlen2018". Dahinter wird erneut das Dokument und die Zeile aufgeführt, in denen der Fehler liegt.
- **„JavaScript Fehleranalyse.html:13“:** Dieser Abschnitt zeigt das Dokument, indem der Fehler vorliegt. Die Zahl hinter dem Doppelpunkt meint die Zeile, in der der Fehler auftritt.
- **„JavaScript Fehleranalyse.html:17“:** Die Fehleranalyse ermittelt hier auch noch Zeile 17 als fehlerhaft, da die Funktion in der Zeile ausgeführt werden soll. Dies ist mit der nicht-deklarierten Variable allerdings nicht möglich.

10.11 Fehleranalyse mit Firefox

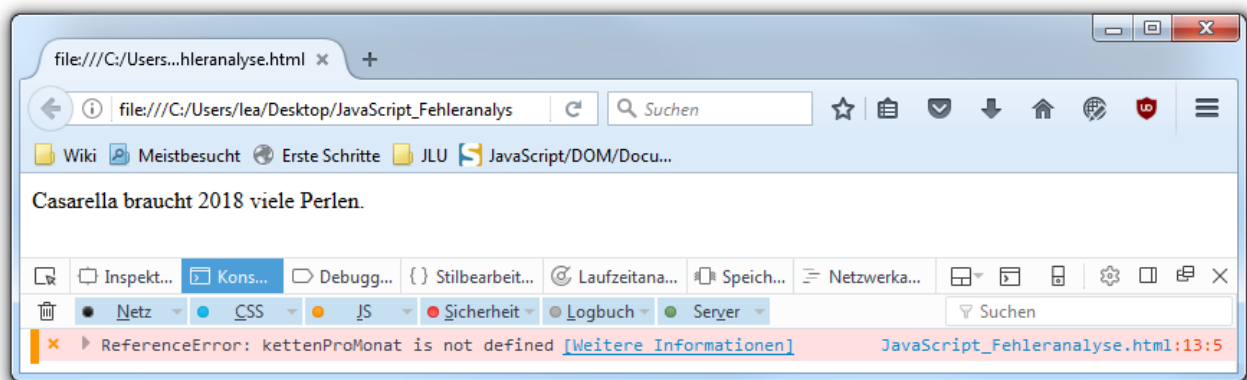


Abb. 101: Fehleranalyse mit Firefox

- Die Konsole in Mozilla Firefox ist sehr ähnlich zu der von Google Chrome. Statt an der rechten Seite wird die Konsole hier an der unteren Seite des Browser-Fensters geöffnet. In der blau markierten Leiste sehen Sie, welche Programmiersprachen im Browser markiert sind.
- Das Feld "Weitere Informationen" leitet Sie zur Online-Fehleranalyse weiter.

10.12 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Die Fehleranalyse eines JavaScript-Codes erfolgt in der _____.		
2	Das Schlüsselwort _____ leitet eine Funktion in JavaScript ein.		
3	Ein rundes Klammernpaar rahmt in JavaScript einen Code-Block ein.		
	Richtig		
	Falsch		
4	Das Verfahren zur Darstellung einer JavaScript-Funktion im Browser ist eine Kombination aus HTML- und JavaScript-Elementen.		
	Richtig		
	Falsch		
5	Funktionen in JavaScript sind nichts anderes als Funktionen aus der Mathematik.		
	Richtig		
	Falsch		
6	Die Darstellung einer JavaScript-Funktion im Browser muss aktiv programmiert werden. Durch ein reines "ausführen" der Funktion ist für den Anwender im Browser nichts zu sehen.		
	Richtig		
	Falsch		

Tab. 7: Abschlusstest WBT 10

11 Zeichenketten und Eingabefelder

11.1 Zeichentheorie

11.1.1 Einführung

Lin W. Lan: „Hallo! Schön, dass Sie wieder bei uns sind. Heute schauen wir uns in JavaScript an, was unter Zeichentheorie zu verstehen ist. Dazu lernen wir Methoden und deren Funktionsweisen kennen. Danach lernen wir, wie man Eingabefelder mit JavaScript erstellt. Viel Spaß!“

11.1.2 Strings

Beate D. Lan: „Wie Sie bereits in einem vorigen WBT gelernt haben, gibt es in JavaScript nicht nur Zahlen, sondern auch sogenannte „strings“. Das sind Zeichenketten, die im JavaScript-Code mit Anführungszeichen gekennzeichnet werden. Variablen können neben Zahlen auch mit Strings befüllt werden. Durch den Aufruf der Variablen wird dann der hinterlegte Text angezeigt. Dies bietet sich bei einem Standard-Text an, der auf der Web Site öfter vorkommt. Zum Beispiel:“

```
var welcome = „Willkommen auf der Web Site Casarella!“
```

11.1.3 Verkettung von Strings

Peter Lan: „Durch ein Pluszeichen (+) können nicht nur mehrere Strings miteinander verkettet werden, sondern auch Variablen werden zu größeren Zeichenketten verbunden. Das heißt dann Stringkonkatenation (lat. Verkettung). JavaScript erkennt durch das Pluszeichen, dass die Kette zusammengehört und kann sie ausgeben. Zum Beispiel:“

Texteditor:

```
<script>
var Lieblingsfarbe = "Blau";
var satzLieblingsfarbe = "Hallo. "+Lieblingsfarbe+
" ist meine Lieblingsfarbe.";
document.write(satzLieblingsfarbe);
</script>
```

Web-Browser:

Abb. 102: Verkettung von Strings im Browser

11.1.4 Eigenschaften von Strings

Peter Lan: „Strings haben nur eine Eigenschaft: length. Wie man an dem Namen erkennen kann, gibt diese Eigenschaft die Länge der Zeichenkette an. Außerdem kann der Programmierer an der Eigenschaft nichts ändern, da sie eine Read-Only-Eigenschaft ist. Bei der Länge einer Zeichenkette werden Leerzeichen und Satzzeichen mitgezählt.“

Die Zeichenkette: "Hallo Welt!" hat also 11 Zeichen. Length muss nicht explizit geschrieben werden, JavaScript zählt die Zeichenkette automatisch.“

11.1.5 Methoden von Strings

Peter Lan: „Die Eigenschaft length hat viele Methoden. Diese Methoden sind Werkzeuge, um **einzelne Zeichen in einer Zeichenkette zu lokalisieren**. Diese Methoden werden für gewöhnlich nicht im Browser abgebildet, sie dienen Programmierern als Hilfestellung beim Programmieren. JavaScript hat 18 Methoden. Wir betrachten hier **drei Methoden** als Beispiele.“

Wie Sie es von Methoden gewohnt sind, werden sie mit einem Punkt eingeleitet. Davor steht das Objekt, in dem Fall also die Zeichenkette in Anführungszeichen. Auf den nächsten Seiten betrachten wir die Methoden genauer.

11.1.6 Methode charAt()

Stefan Ethernet: „Diese Methode identifiziert das Zeichen an der n-ten Stelle der Zeichenkette. Beachten Sie bitte, dass Zeichenketten in JavaScript immer mit der Stelle "0" beginnen. Die Position der n-ten Stelle wird in Klammern hinter die Methode gesetzt. Auf dem Laptop sehen Sie ein Beispiel:

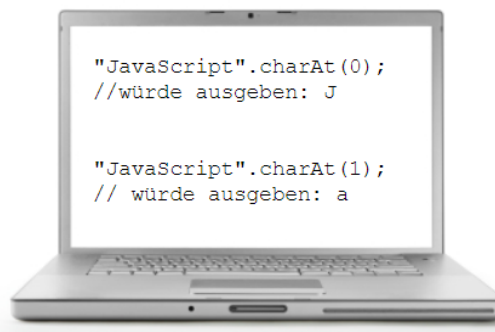


Abb. 103: Beispiele der Methode `charAt()`

Bitte beachten Sie: Die Indizes haben nichts mit der Eigenschaft `"length"` der Zeichenkette zu tun. Die Länge von `JavaScript` entspricht 10 Zeichen. An Stelle 0 steht das Zeichen `"J"` und an Stelle 1 das Zeichen `"A"`.

11.1.7 Methoden `indexOf()` und `lastIndexOf()`

Diese Methoden untersuchen, **wo ein bestimmtes Zeichen in der Zeichenkette zu finden ist**. Hierbei sucht `indexOf()` das Zeichen beginnend bei der ersten Stelle der Zeichenkette. `LastIndexOf()` beginnt die Suche bei der letzten Stelle der Zeichenkette.

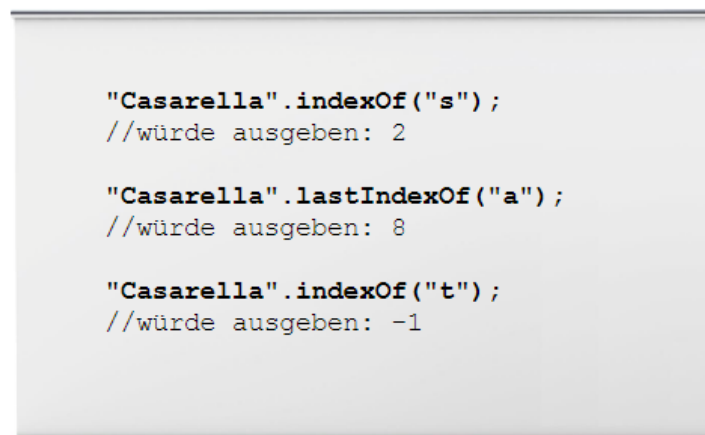


Abb. 104: Beispiele von Methoden in JavaScript

- Bitte achten Sie darauf, die gesuchten Strings immer in Anführungszeichen zu setzen.
- Die Suche beginnt JavaScript bei `.lastIndexOf()` an der letzten Stelle der Zeichenkette. Das erste `"a"`, welches JavaScript von hinten findet, steht an Stelle 8.
- Wenn ein String in der gesuchten Zeichenkette nicht vorkommt, gibt JavaScript immer `-1` aus.

11.1.8 Methoden `indexOf()` und `lastIndexOf()` – Teil 2

Lin W. Lan: „Eine zusätzliche Funktion dieser Methoden ist, dass hinter dem gesuchten String in den Klammern ein Parameter angegeben werden kann, der festlegt, **an welcher Stelle JavaScript anfängt zu suchen**.

```
"Casarella".indexOf("a",2);  
//würde ausgeben: 3  
  
"Casarella".lastIndexOf("a", 2);  
//würde ausgeben: 1
```

Abb. 105: Beispiele von Methoden in JavaScript – Teil 2

- **Zum 1. Beispiel:** Hier werden die ersten zwei Zeichen übersprungen (C,a) und die Suche beginnt bei Index 2, also "s". Das "a" welches nach dem kleinen "s" folgt, hat den Index 3.
- **Zum 2. Beispiel:** Die beiden ersten Indizes von hinten werden übersprungen ("a","l") und die Suche beginnt bei "l". Das nächste "a" trägt den Index 1.

11.1.9 Methoden – Übung

Lin W. Lan: „Nun sind Sie gefragt. Verwenden Sie den String: "Der Umgang mit Methoden in JavaScript macht Spaß!" Wenden Sie bitte folgende Methoden an:

- indexOf("g",7)
- lastIndexOf("a")
- CharAt(25)

Lösung:

- 9
- 46
- Leerzeichen

11.2 Eingabefelder mit JavaScript

11.2.1 Eingabefelder

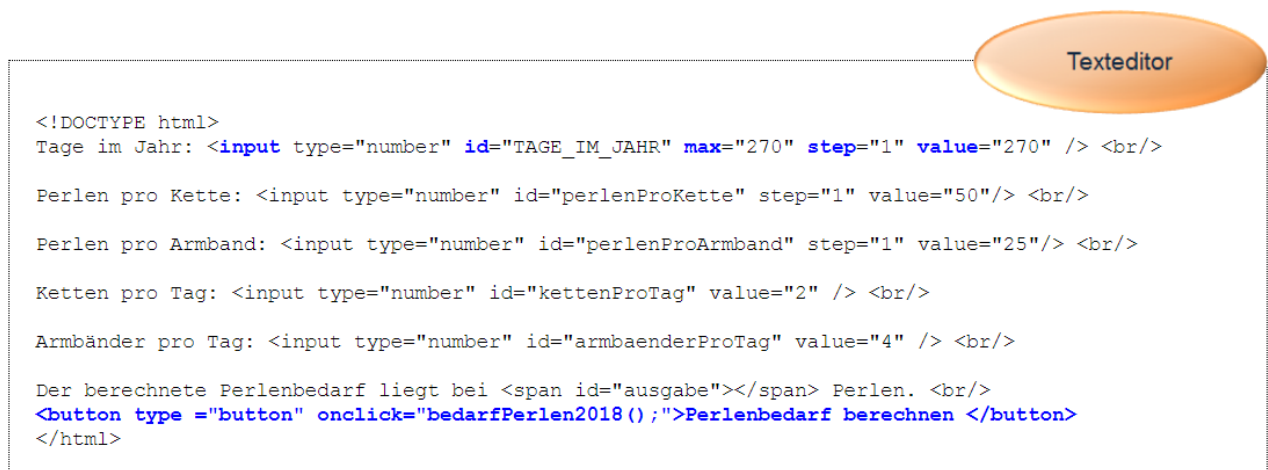
Peter Lan: „Die Betreiber der Casarella-Web-Site wünschen sich für die Perlenbedarf-Funktion aus dem letzten WBT eine Änderung.

Sie möchten den Perlenbedarf im Intranet-Bereich selbst berechnen, haben allerdings keine Programmierkenntnisse. Dafür werden wir Eingabefelder bereitstellen. Eingabefelder bestehen aus HTML-Elementen, die mit einer JavaScript-Funktion zusammen agieren. Mit den Eingabefeldern kann der

Perlenbedarf individuell im Browser berechnet werden. Auf der nächsten Seite sehen Sie, welche Eingabefelder auf der Casarella Web Site angelegt werden.

11.2.2 Eingabefelder des Typs "number" – HTML-Elemente

Da für die Funktion `perlenBedarf2018` nur Zahlen in die Eingabefelder eingegeben werden müssen, verwenden wir den Eingabefeld-Typ "number". So gehen wir sicher, dass ausschließlich Zahlen eingegeben werden. Für Texteingaben kennt JavaScript den input type "text".



The image shows a screenshot of a text editor window titled "Texteditor". The editor contains the following HTML code:

```
<!DOCTYPE html>
Tage im Jahr: <input type="number" id="TAGE_IM_JAHR" max="270" step="1" value="270" /> <br/>

Perlen pro Kette: <input type="number" id="perlenProKette" step="1" value="50"/> <br/>

Perlen pro Armband: <input type="number" id="perlenProArmband" step="1" value="25"/> <br/>

Ketten pro Tag: <input type="number" id="kettenProTag" value="2" /> <br/>

Armbänder pro Tag: <input type="number" id="armbaenderProTag" value="4" /> <br/>

Der berechnete Perlenbedarf liegt bei <span id="ausgabe"></span> Perlen. <br/>
<button type="button" onclick="bedarfPerlen2018();" >Perlenbedarf berechnen </button>
</html>
```

Abb. 106: HTML-Elemente eines Eingabefeldes

- **input:** Für die Variablen `TAGE_IM_JAHR` usw. werden Eingabefelder erzeugt.
- **id:** Jedes Eingabefeld wird mit einer ID versehen, auf die JavaScript später für die Berechnung der Funktion zurückgreift.
- **max:** Die Eigenschaft `max` minimiert die Eingabefelder. Die maximale Anzahl an Arbeitstagen im Jahr ist 270.
- **step:** Das HTML-Attribut `step` meint, in welchen Stufen die Werte (durch eine Scrollbar) eingegeben werden dürfen. Hier können also nur ganze Zahlen eingefügt werden.
- **value:** Die Werte hinter `Value` sind Platzhalter. Mit ihnen wird gerechnet, falls der Nutzer Eingabefelder frei lässt.
- **<button type="button" onclick="bedarfPerlen2018();" >Perlenbedarf berechnen </button>:** Der hier eingefügte Button führt bei Klick die Funktion `bedarfPerlen2018` aus.

11.2.3 Eingabefelder des Typs "number" – JavaScript-Elemente

Lin W. Lan: Die Durchführung der Berechnung führt JavaScript aus. Die eingegebenen Elemente werden durch den JavaScript-Befehl `document.getElementById()` in den JavaScript-Quellcode inte-

griert. Berechnung und Ausgabe der Funktion erfolgen über JavaScript. Die Methode `.value` veranlasst, dass JavaScript mit dem Wert rechnet, der sich aktuell im Eingabefeld befindet. Auf der nächsten Seite sehen Sie, wie die Funktion im Browser dargestellt wird.

Texteditor

```
<script>
function bedarfPerlen2018() {
    var TAGE_IM_JAHR = document.getElementById ("TAGE_IM_JAHR").value;
    var perlenProKette = document.getElementById ("perlenProKette").value;
    var perlenProArmband = document.getElementById ("perlenProArmband").value;
    var kettenProTag = document.getElementById ("kettenProTag").value;
    var armbaenderProTag = document.getElementById ("armbaenderProTag").value;
    var bedarfPerlenProTag = kettenProTag * perlenProKette + armbaenderProTag*perlenProArmband;
    var perlen2018 = bedarfPerlenProTag*TAGE_IM_JAHR;
    document.getElementById("ausgabe").innerHTML=perlen2018;}
bedarfPerlen2018 ();
</script>
```

Abb. 107: JavaScript-Elemente eines Eingabefeldes

11.2.4 Eingabefelder – Browser-Ansicht

Peter Lan: „Hier sehen Sie die Darstellung des Quellcodes in Firefox. Da Casarella die Eingabefelder nur im Intranet benötigt, wurde hier von der Gestaltung mit CSS abgesehen.“

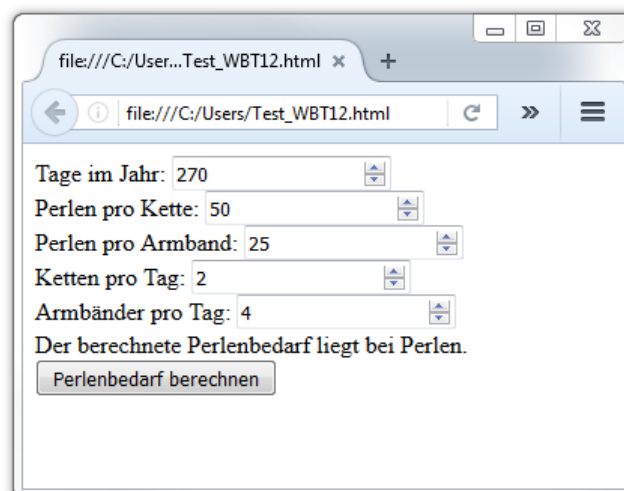


Abb. 108: Eingabefelder im Browser

11.2.5 Eingabefelder – Übung

Erstellen Sie Eingabefelder für folgende JavaScript-Funktion:

```
<script>
function bedarfEdelsteine2018 ()
{
var TAGE_IM_JAHR = 270;
```

```

var edelsteineProKette = 4;
var edelsteineProArmband = 2;
var kettenProTag = 2;
var armbaenderProTag = 4;
var bedarfEdelsteineProTag =kettenProTag*
edelsteineProKette + armbaenderProTag * edelsteineProArmband
var edelsteine2018 = bedarfEdelsteineProTag * TAGE_IM_JAhr;
}
bedarfEdelsteine2018();
</script>

```

Für Lösungshinweise schauen Sie sich einfach die vorherigen Folien an und verwenden Sie den dort gegebenen Programmiercode als Vorlage.

11.2.6 Abschlusstest

Nr.	Frage	Richtig	Falsch
1	Die Methode _____ sucht die Position eines Buchstabens in einer JavaScript-Zeichenkette..		
2	Die Verwendung der Methode charAt(5) bei dem String "Casarella" würde das Zeichen _____ ausgeben.		
3	Strings werden immer ohne Anführungszeichen aufgeführt.		
	Richtig		
	Falsch		
4	Das Verfahren zur Darstellung einer JavaScript-Funktion im Browser ist eine Kombination aus HTML- und JavaScript-Elementen.		
	Richtig		
	Falsch		
5	Strings haben nur eine Eigenschaft: width.		
	Richtig		
	Falsch		
6	Die Methoden lastIndexOf() und indexOf() unterscheiden sich nicht.		
	Richtig		
	Falsch		

Tab. 8: Abschlusstest WBT 11

12 Bedingungen in JavaScript

12.1 Willkommen zurück

Lin W. Lan: „Hallo! Schön, dass Sie wieder bei uns sind. Heute schauen wir uns in JavaScript an, wie man Funktionen an Bedingungen anknüpft. So können zum Beispiel User nur unter bestimmten Voraussetzungen eine Web Site besuchen. Um bedingte Funktionen mit JavaScript zu erstellen, lernen wir mit den Programmierern Beate D. Lan und Peter Lan if-else-Statements kennen. Viel Spaß!“

12.2 Bedingungen in JavaScript

Beate D. Lan: „Sie können die Ausführung von JavaScript-Quellcode von Bedingungen abhängig machen. Von Code-Blöcken wird dann nur derjenige durchlaufen, der eine zuvor gestellte Bedingung erfüllt. So könnte zum Beispiel die Casarella-Web-Site die Bedingung enthalten, dass Kunden mindestens 16 Jahre alt sein müssen, um eine Bestellung aufzugeben. Gibt der Kunde ein Alter an, welches kleiner als 16 ist, wird der Bestellvorgang abgebrochen. Andernfalls kann die Bestellung erfolgen.“

12.3 If-else-Statements

Bedingungen entstehen durch die Schlüsselwörter „if“ (=wenn) und „else“ (=dann/sonst). Mit „if“ wird eine Wenn-Dann-Bedingung eingeleitet. Dahinter folgt in Klammern die Formulierung der Bedingung mit Vergleichsoperatoren und Variablen. Hier sehen Sie den Aufbau einer if-else-Bedingung, über den Aufbau einer Bedingung lernen Sie auf der nächsten Seite.

Eine Bedingung in JavaScript ist eine Aussage, die der Computer mit Ja (=truthy) oder Nein (=falsy) beantworten kann. Eine Bedingung muss immer einen Operator enthalten.

```

if (Bedingung) {

    Code, der bei Erfüllung dieser
    Bedingung ausgeführt werden soll

} else {

    Code, der ausgeführt wird, wenn
    if-Bedingung nicht erfüllt wird.

}

if (hour < 18) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}

```

Abb. 109: Bedingungen in JavaScript

Operator	Bedeutung
>	größer als
>=	größer gleich
==	gleich
<=	kleiner gleich
<	kleiner
!=	ungleich
===	gleicher Wert und gleicher Typ
!==	Entweder gleicher Wert, aber unterschiedlicher Typ oder ungleicher Wert, aber gleicher Typ

Abb. 110: Operatoren in JavaScript

- **Beispiel "==="** : 3 === 3 ist true, aber 3 === '3' ist false.

12.4 Bedingungen in if-else-statements

Peter Lan: „Keine Sorge, wir gehen alles nacheinander durch. Am besten schauen wir uns das Beispiel von eben an: Was lesen wir aus dem Code? Bis 18 Uhr schreiben wir die Begrüßung "Guten Tag", danach "Guten Abend". Das if-else-Statement inklusive der Bedingung ist Teil einer JavaScript-Funktion. Eine solche schauen wir uns auf der nächsten Seite an.

```
if (hour < 18) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

Abb. 111: Beispielhaftes if-else-Statement

12.5 Bedingungen – Teil 2

Die Bedingung wird im Web Browser mit Hilfe eines HTML-Buttons dargestellt. Diese Form der Ausführung eines JavaScript-Codes kennen Sie ja bereits. Lediglich das if-else-Statement ist neu für Sie.

12.6 If...Else... – Interpretation von JavaScript

Um den richtigen Code-Block eines if-else-Statements auszuführen, muss JavaScript die aufgeführten Bedingungen interpretieren. Ergebnisse von Bedingungsprüfungen werden entweder als wahr (=truthy) oder falsch (=falsy) eingestuft. Es gibt verschiedene Ausprägungen von falsy-Werten:

Falsy-Werte sind:

- false
- 0 (Null als string ist truthy)
- leere Zeichenkette
- null
- NaN (not a number)
- Undefined

Truthy-Werte sind:

- true
- strings
- numbers und
- booleans

12.7 else-if-Statement

Beate D. Lan: „Natürlich kann man auch zwischen mehr als zwei unterschiedlichen Bedingungen differenzieren. Wie? Mit dem else-if-Statement! Es wird zwischen das if und das else eingeschoben.“

Man kann so viele else-if-Statements einschieben, wie man möchte. Dabei ist auf die logische Reihenfolge der Bedingungen zu achten, da JavaScript von oben nach unten gelesen wird. Ich habe das Ihnen bekannte Beispiel abgeändert und ein else-if-statement hinzugefügt. Nun wird man mit "Good Morning" begrüßt, wenn es früher als 10 Uhr ist. Wenn es früher als 20 Uhr ist, lautet die Begrüßung "Good Day" und wenn keine der beiden Bedingungen zutrifft, wird "Good Evening" ausgegeben.

12.8 Kombination von Bedingungen

Peter Lan: „JavaScript ermöglicht es, zwei Bedingungen miteinander zu kombinieren. Dafür gibt es zwei verschiedene Möglichkeiten:

1. "&&" : Beide Bedingungen müssen erfüllt sein, damit JavaScript die Bedingung als "truthy" (=wahr) interpretiert.
2. "||" --> Wenn eine der beiden Bedingungen erfüllt ist, erkennt JavaScript die Bedingung als wahr an.“

12.9 Negations-Bedingung

Beate D. Lan: „Der logische Operator "!" prüft, ob ein Ausdruck unwahr ist. Dies kann hilfreich sein, wenn beispielsweise Nutzer vergessen haben, Daten einzutragen. Ich habe ein entsprechendes Beispiel für Sie mitgebracht:“

```
<script>
var PLZ = 81000,
    x = 20,
    y = 8,
    name = "";

if (PLZ >= 80000 && PLZ <= 82000)
    alert("Sie wohnen wohl in München oder Umgebung!");

if (x > 100 || y == 0) {
    return;
} if (!name) {
    alert("Sie haben vergessen, einen Namen einzugeben!");
} else {
    alert("Hallo" + name);
}
</script>
```

Abb. 112: Beispiel der Negations-Bedingung

Das Ausrufezeichen vor der Bedingung dreht den Wahrheitsgehalt des Ausdrucks um. Also: Wenn die Variable name = false (also leer oder null) ist, wird der alert ausgegeben, dass der Name vergessen wurde.

12.10 If-Else – Übung

Erstellen Sie ein if-else-Statement: Verwenden Sie Ihr eigenes Alter als Variable. Wenn die Variable größer als 25 ist, geben sie das Statement "alt genug" aus. Für den Fall, dass jemand jünger als 16 ist, geben Sie "kein Studierender" aus. Geben Sie für alles andere das Statement "zu jung" aus. Kopieren Sie den Code-Block aus der Box fügen Sie ihn in einen Editor Ihrer Wahl ein. Fügen Sie das if-else-Statement hinzu.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Zeigen Sie hier das Ergebnis.</p>

<script>
var age = //Ihr Alter;

// Fügen Sie hier das if-else-statement ein.
</script>

</body>
</html>
```

Überprüfen Sie Ihre Lösungen hier:

```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Stellen Sie hier Ihr Ergebnis dar.</p>

<script>
var statement;
var age = 24;

if (age > 25 ) {
    statement = "Alt genug";
} else if (age < 16) {
    statement = "Kein Studierender";
} else {
    statement = "Zu jung";
}
document.getElementById("demo").innerHTML =
statement;
</script>

</body>
</html>
```



Abb. 113: Lösung if-else-Übung

12.11 Abschluss

Lin W. Lan: „Wir haben Ihnen heute Bedingungen in JavaScript nähergebracht. Damit sind wir auch schon am Ende unserer gemeinsamen Zeit mit JavaScript angekommen. Wir hoffen, wir konnten Sie für JavaScript begeistern und Ihre Eigeninitiative zu dem ein oder anderen Thema wecken. Bleiben Sie auf jeden Fall am Ball und geben Sie nicht auf, wenn Ihr Code nicht beim ersten Mal funktioniert, das ist völlig normal! Bis bald!“

Anhang

Lösungen zur Übung in WBT 04 – Übung zu Datenlisten

Nr.	Frage	Richtig	Falsch
1	Eine Datenliste ist eine Liste von Einträgen, die dem Benutzer vorgeschlagen werden, wenn er in ein Texteingabefeld tippt. Eines der vorgeschlagenen Einträge muss gewählt werden.		
	Richtig		X
	Falsch	X	
2	Der Tag-Body des <option>-Tags wird bei Datenlisten ignoriert. Die angezeigten Einträge kommen aus dem value-Attribut.		
	Richtig	X	
	Falsch		X
3	Möglichkeiten eine Auswahlliste zu erstellen bieten:		
	Radio-Buttons	X	
	Checkboxen		X
	Select-Boxen	X	
	Datenlisten	X	
4	Die angezeigten Einträge der Auswahlliste werden durch den Tag-Body des <option>-Tags erzeugt.		
	Richtig		X
	Falsch	X	
5	Eine Datenliste wird in HTML durch den <option>-Tag eingeleitet.		
	Richtig		X
	Falsch	X	
6	Die <option>-Tags werden in den <datalist>-Tag verschachtelt.		
	Richtig	X	
	Falsch		X
7	Im list-Attribut wird auf die id der Datalist verwiesen.		
	Richtig	X	
	Falsch		X

Tab. 9: Lösung zu Datenlisten

Lösungen zum Abschlusstest in WBT 05

Nr.	Frage	Richtig	Falsch
1	Gegenüber älteren Methoden bietet das responsive Webdesign den Vorteil, dass man eine angepasste Variante zusätzlich zu der originalen HTML-Datei hat.		
	Richtig		X
	Falsch	X	
2	Die Web-Site-Anzeige auf der Smartwatch passt sich automatisch an.		
	Richtig		X
	Falsch	X	
3	Beispiele für Medientypen sind:		
	screen	X	
	handheld	X	
	all	X	
4	Betrachtet man eine Web Site auf verschiedenen Ausgabegeräten, ergibt sich das Problem, dass die _____ und die _____ variieren.		
	Bildschirmgröße		
	Bildschirmauflösung		
5	Das Responsive Webdesign sorgt dafür, dass die Web Site auf Eigenschaften des jeweils benutzten Endgeräts, vor allem Smartphones und Tablet-Computer, reagieren kann.		
	Richtig	X	
	Falsch		X
6	Medienabfragen in HTML sehen wie folgt aus: @media screen		
	Richtig		X
	Falsch	X	
7	Für die Technik des Responsive Webdesign benötigt man ausschließlich HTML.		
	Richtig		X
	Falsch	X	
8	Media Queries sind der Kern des_____.		
	Responsive Webdesign		
9	Beispiele für Medienmerkmale sind:		
	height	X	

	high		X
	width	X	
	monochrome	X	
10	Beim Responsive Webdesign passen sich die Inhalts- und Navigationselemente sowie auch der strukturelle Aufbau der Web Site der Bildschirmauflösung des mobilen Endgeräts an.		
	Richtig	X	
	Falsch		X
11	Erfüllt das verwendete Ausgabemedium alle Kriterien einer Medienabfrage, so wird die damit verbundene CSS-Ressource eingebunden.		
	Richtig	X	
	Falsch		X
12	Durch eine Media Query ändert sich die Priorität der CSS-Regeln nicht.		
	Richtig	X	
	Falsch		X

Tab. 10: Lösung zu Abschlusstest WBT 05

Lösungen zum Abschlusstest in WBT 06

Nr.	Frage	Richtig	Falsch
1	Pseudoklassen sprechen immer ein Element an.		
	Richtig		X
	Falsch	X	
2	Die Hintergrundfarbe des Eingabefeldes einer korrekten Eingabe kann durch die Pseudoklasse :valid geändert werden.		
	Richtig	X	
	Falsch		X
3	Mit der Pseudoklasse :hover hebt man das Element hervor, das vom Benutzer mit dem Mauszeiger angeklickt wird.		
	Richtig		X
	Falsch	X	
4	Inkorrekte Eingabefelder können durch die Pseudoklasse :focus hervorgehoben werden.		
	Richtig		X
	Falsch	X	
5	:optional spricht die <input>-tags an, die ein required-Attribut gesetzt haben.		
	Richtig		X
	Falsch	X	
6	Die Pseudoklassen :required und :optional beziehen sich auf das _____-Attribut bei den Eingabefeldern eines Formulars.		
	required		
7	Pseudoklassen beginnen im Stylesheet immer mit einem _____.		
	Doppelpunkt		
8	Bei Formularen auch verwendbar:		
	Schrifteigenschaft	X	
	Farbe	X	
	Rahmen	X	
9	Die Pseudoklasse :focus bekommt das Element, das gerade vom Benutzer den Fokus hat. Das kann ein Eingabefeld sein, in das er tippt, aber auch ein Link, auf den gerade geklickt wird.		
	Richtig	X	
	0Falsch		X

10	:focus und :hover werden selten für Formulare verwendet.		
	Richtig		X
	Falsch	X	

Tab. 11: Lösung zu Abschlusstest WBT 06

Lösungen zum Abschlusstest in WBT 08

Nr.	Frage	Richtig	Falsch
1	Der Aktionsbefehl <code>__doc.write__</code> führt die JavaScript-Befehle direkt auf der Web-Seite aus.		
2	JavaScript als Programmiersprache ist zuständig für die farbliche Gestaltung einer Web Site.		
	Richtig		X
	Falsch	X	
3	JavaScript kennt die vier Grundrechenarten.		
	Richtig	X	
	Falsch		X
4	Durch die "alert"-Funktion wird ein Dialogfenster geöffnet, auf welches der Nutzer reagieren muss. Solange dies nicht geschieht, lädt die Seite keine weiteren Inhalte.		
	Richtig	X	
	Falsch		X
5	Wählen Sie die richtige Antwort aus.		
	Die Anführungszeichen in der Klammer eines JavaScript Befehls zeigen den Verlauf einer Zeichenkette.	X	
	Ein Befehl wird immer mit einem Semikolon beendet.	X	
	Der noscript-Tag kommt bei JavaScript nur selten zur Anwendung.		X
6	Bei dem Befehl "document.write" wird das Ergebnis einer Funktion direkt im Text der Seite angezeigt.		
	Richtig	X	
	Falsch		X
7	JavaScript...		
	..ist eine vollwertige Programmiersprache.	X	
	...wird zur Programmierung komplexer Programme angewendet.	X	
	...muss direkt im HTML-Dokument ausgeführt werden.		X

Tab. 12: Lösung zu Abschlusstest WBT 08

Lösungen zum Abschlusstest in WBT 09

Nr.	Frage	Richtig	Falsch
1	Wiederholen Sie die Übung der vorigen Seite, aber rechnen Sie nun 7.9 cm in Inch um. Welches Ergebnis, zeigt Ihnen Ihr Web Browser gerundet auf zwei Nachkommastellen an?		
	4.9	X	
	3.11	X	
	3.42	X	
2	Die Variable <code>___var___</code> ist von jedem Browser einwandfrei interpretierbar.		
3	JavaScript kann nur mit ganzen Zahlen rechnen.		
	Richtig		X
	Falsch	X	
4	Der Einsatz von Variablen in einer Programmiersprache dient dazu, dem Programmierer unnötige Arbeit zu ersparen, da man auf Variablen immer wieder zugreifen kann.		
	Richtig	X	
	Falsch		X
5	Variablen können nur mit "strings" befüllt werden, nicht aber mit "numbers".		
	Richtig		X
	Falsch	X	
6	Wählen Sie die richtige Antwort aus.		
	"const" kennzeichnet eine konstante Variable.		X
	Für Casarella eignet sich die Variable "var" am Besten.	X	
	Die Nachkommastellen sind in JavaScript begrenzt.	X	
7	Eine Schreibweise, mit der man Variablen kennzeichnen kann, heißt "camelCase".		
	Richtig	X	
	Falsch		X

Tab. 13: Lösung zu Abschlusstest WBT 09

Lösungen zum Abschlusstest in WBT 10

Nr.	Frage	Richtig	Falsch
1	Die Fehleranalyse eines JavaScript-Codes erfolgt in der ____Konsole____.		
2	Das Schlüsselwort ____function____ leitet eine Funktion in JavaScript ein.		
3	Ein rundes Klammernpaar rahmt in JavaScript einen Code-Block ein.		
	Richtig	X	
	Falsch		
4	Das Verfahren zur Darstellung einer JavaScript-Funktion im Browser ist eine Kombination aus HTML- und Ja- vaScript-Elementen.		
	Richtig	X	
	Falsch		
5	Funktionen in JavaScript sind nichts anderes als Funktio- nen aus der Mathematik.		
	Richtig		
	Falsch		X
6	Die Darstellung einer JavaScript-Funktion im Browser muss aktiv programmiert werden. Durch ein reines "aus- führen" der Funktion ist für den Anwender im Browser nichts zu sehen.		
	Richtig	X	
	Falsch		

Tab. 14: Lösung zu Abschlusstest WBT 10

Lösungen zum Abschlusstest in WBT 11

Nr.	Frage	Richtig	Falsch
1	Die Methode <code>____charAt()____</code> sucht die Position eines Buchstabens in einer JavaScript-Zeichenkette..		
2	Die Verwendung der Methode <code>charAt(5)</code> bei dem String "Casarella" würde das Zeichen <code>____"e"____</code> ausgeben.		
3	Strings werden immer ohne Anführungszeichen aufgeführt.		
	Richtig		
	Falsch		X
4	Das Verfahren zur Darstellung einer JavaScript-Funktion im Browser ist eine Kombination aus HTML- und JavaScript-Elementen.		
	Richtig	X	
	Falsch		
5	Strings haben nur eine Eigenschaft: <code>width</code> .		
	Richtig		
	Falsch		X
6	Die Methoden <code>lastIndexOf()</code> und <code>indexOf()</code> unterscheiden sich nicht.		
	Richtig		
	Falsch		X

Tab. 15: Lösung zu Abschlusstest WBT 11

Impressum



- Reihe:** **Arbeitspapiere Wirtschaftsinformatik** (ISSN 1613-6667)
- Bezug:** <https://wi.uni-giessen.de>
- Herausgeber:** Prof. Dr. Axel Schwickert
Prof. Dr. Bernhard Ostheimer
- c/o Professur BWL – Wirtschaftsinformatik
Justus-Liebig-Universität Gießen
Fachbereich Wirtschaftswissenschaften
Licher Straße 70
D – 35394 Gießen
Telefon (0 64 1) 99-22611
Telefax (0 64 1) 99-22619
eMail: Axel.Schwickert@wirtschaft.uni-giessen.de
<https://wi.uni-giessen.de>
- Ziele:** Die Arbeitspapiere dieser Reihe sollen konsistente Überblicke zu den Grundlagen der Wirtschaftsinformatik geben und sich mit speziellen Themenbereichen tiefergehend befassen. Ziel ist die verständliche Vermittlung theoretischer Grundlagen und deren Transfer in praxisorientiertes Wissen.
- Zielgruppen:** Als Zielgruppen sehen wir Forschende, Lehrende und Lernende in der Disziplin Wirtschaftsinformatik sowie das IT-Management und Praktiker in Unternehmen.
- Quellen:** Die Arbeitspapiere entstehen aus Forschungs-, Abschluss-, Studien- und Projektarbeiten sowie Begleitmaterialien zu Lehr-, Vortrags- und Kolloquiumsveranstaltungen der Professur BWL – Wirtschaftsinformatik, Prof. Dr. Axel Schwickert, Justus-Liebig-Universität Gießen sowie der Professur für Wirtschaftsinformatik, insbes. medienorientierte Wirtschaftsinformatik, Prof. Dr. Bernhard Ostheimer, Fachbereich Wirtschaft, Hochschule Mainz.
- Hinweise:** Wir nehmen Ihre Anregungen zu den Arbeitspapieren aufmerksam zur Kenntnis und werden uns auf Wunsch mit Ihnen in Verbindung setzen.
- Falls Sie selbst ein Arbeitspapier in der Reihe veröffentlichen möchten, nehmen Sie bitte mit einem der Herausgeber unter obiger Adresse Kontakt auf.
- Informationen über die bisher erschienenen Arbeitspapiere dieser Reihe erhalten Sie unter der Web-Adresse <https://wi.uni-giessen.de/>