SGML > HTML > HTML5 (eigenständig) (Hypertext Markup Language)

<!DOCTYPE html>	<html lang="en">
<head>	<title>My Site</title>
<meta charset="utf-8" />	<meta name="" content="" />

Includes (CSS, JS, Favicons)

<link rel="icon" href="" type="image/png" />	<style>…inline…</style>

<link rel="stylesheet" href="" type="text/css" media="screen|print|all" />

<script src="main.js" *async*></script> (immer schliessen! Ansonsten wir alles folgende als Script interpretiert. Im Footer einbinden)

Layout (Alles Block Elemente)

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

Text

<h1> … <h6> <p> <small> <address> <time> <abbr> <blockquote> (Zitat) <cite> (Titel einer Arbeit) <pre> <code> (Important) (Emphasized) <i> <hr/>

Hyperlinks

Report

Lists

	<ol start="" reversed>	<dl><dt>(Term) <dd> (Desc.)

Tables (nie fürs Layout verwenden → Nicht responsive, Wartbarkeit, Speed)

<table><caption>Text</caption><colgroup><col span="2"/></colgroup>
<thead><tr><th colspan="2"> <tbody><tr><th><td rowspan="2"><tfoot>

Images (<picture> hat aktuell schlechten Support) / Width,Heigh → Of Source

	

<picture><source type="image/jpeg" srcset=""/> <source type="image/webp" srcset=""/>
 (Mehrere Sourcen, IMG Tag als Fallback)

<figure><figcaption>Caption</figcaption></figure>

Audio / Video / Object / SVG (<embed> → do not use, <object> → more adaptable)

<audio controls autoplay loop muted><source src="" type="audio/ogg"/></audio>

<video poster="y.jpg"><source type="video/mp4" src=""/>Fallback link

<object data="x.svg" type="image/svg+xml|application.pdf">

Frames → Einbinden von Inhalten aus Web (andere Webseite, YouTube,…)

<iframe src="logo.svg|./user.html" name="User Site"></iframe>

Forms (HTML5 Validation → required & pattern="")

<form action="./send" method="post" autocomplete="on|off" nonvalidate>

<fieldset><legend>Employee Form</legend></fieldset>

<lable for="user-name">User Name</label> (kann auch Inputs enthalten (~~for~~))

<input type="hidden|text|search|password|url|tel|email|checkbox|radio|datetime-local|color" name="user-name" id="user-name" placeholder="text" required autofocus />

<textarea name="" id="" required pattern="([A-Z)?[0-9]*"></textarea>

<select name="" multiple><option value="" selected>Text</option></select>

<datalist id="foo"><option value="A"/> … </datalist> <input type="text" list="foo"/>

<input type="range" min="1" max="5" step="1" value="3" />

<input type="number" name="age" id="age" min="0" max="120" />

<button type="submit|reset">Text</button> (besser als Input type="submit")

Roles (WAI-ARIA: Web Accessibility Initiaitive – Accessible Rich Internet Apps)

banner, complementary, contentinfo, form, main, navigation, search

CSS3 (Cascading Style Sheets)

Box Model = Content + padding + border + margin (box-sizing :border-box)

Selektoren

Universal	*	Attribut	input[type=submit]
ID	#my-id	Klassen	.my-class
Nachfahren	.p a (irgendwann)	Kind	.p > a (direkt)
Geschwister	.p ~ a (irgendwann)	Nachbarn	.p + a (direkt)

Attributselektor: a[href=v] oder input[type=v] → Case Sensitive

Gleich	[a=v]	Ended	[a$=v]
Startet mit	[a|=v] (v alleinstehend)	Startet mit	[a^=v]
Beinhaltet	[a~=v] (v alleinstehend)	Beinhaltet	[a*=v]

Pseudoelemente:

::before, ::after, ::first-line, ::first-letter (erster Buchstabe), ::selection (selektierte Text)

Pseudoklassen

:first-child, :ntd-child(2n|odd|even), :empty, :hover, :active, :focus, :visited, :not()

Spezifität eines Selektor (Universalselektor * wird ignoriert) → Höchse Spezifität wins!

A+: Inline Style: 1=ja, 0=nein **A:** Vorkommen ID **B:** Vorkommen Pseudoklasse, Attributselektor, Klassenselektor (not() wird ignoriert, der Inhalt der Klammer aber gezählt) **C:** Typselektor oder Preudoelement.

Quelle (autor, user, browser) > !important Tag > Spezifität > Reihenfolge im CSS

Vererbte CSS Eigenschaften

color, font, text-indext, text-align, cursor, list-style

Einheiten / Grössen (Default Basis = 16px = 1rem = 1 em)

px = Fix	em = Relativ Parent	rem = Relativ Root (html)	vw,vh = % Viewport

Display

Inline: nur left/right margin/padding, ignoriert width/height, erlaubt andere elem auf Zeile
block: erlaubt margin/padding, füllt ganze Zeile (p, div, ul, article, table, figure, h1..6, u.w)
inline-block: erlaubt margin/padding, erlaubt andere elem auf Zeile, erlaubt width/height
none: entfernt, elem rutschen nach != visibility: hidden → entfernt, space bleibt

Titel-Nummerierung / Counter: body{counter-reset: my-counter;}
h3:before {counter-increment: my-counter; content: counter(my-counter);}

JavaScript / ECMAScript (wird interpretiert)

Primitives	String, number, boolean, null, undefined, symbol
Objekte	Plain Objects, Arrays, Regex, Functions, undefined, null

Semikolons werden theoretisch vom Interpreter automatisch eingefügt

Rechenregeln: Punkt vor Strich und von Links nach Rechts.

"4" + "2" = "42"	"4" - "2" = 2 (-,*,/)	"foo" + + "abc" = "fooNaN"	"a" - -1 = NaN

Vergleichen: Immer mit === (verhindert Typumwandlung von primitiven DT)

"123" == 123 → true	"123" === 123 → false	"A" < "B" → true

Boolean: false = 0, "" (empty String), null, undefined, NaN Rest: true = "0","false",[],{}

Numbers: 0.333333333333 * 3 == 1 (Check mit Number.isSafeInteger();)

parseInt("1.2ab") = 1	parseFloat("1.2ab") = 1.2	"1ab" = NaN	[1,2]=="1,2"→true

NaN = parseInt("abc"), 0 / 0, []	typeOf(NaN) = 'number'	NaN == NaN → false

Infinity: 3 / 0, Math.pow(2,10000)	**Negativ Infinity:** -Math.pow(2,10000)

Undefined : typeof (myVar) != 'undefined' && myVar.name != undefined;

NULL: null == undefined → true	null == false → false	[] == ![] → true

Object: var person = {name: 'Tim', hallo: function(){}};	var name = person["name"];

Functions: var fnHello = function(fnPrint) { fnPrint();};

Offene Parameterliste (arguments): Array.prototype.slice.call(arguments);

Rest Parameters (ECMA6): foo(name, …restParams) { params.join(':')};

Lambda: arr.filter(elem => elem > 5); / arr.map(x => x*x);

String: .length, slice(), trim(), indexOf(), .replace("pattern", "val")

String Templates: `my name is ${myVariable}` → Backtick verwenden!

Arrays: arr[0] = 'test' / arr.push('test') / arr.lenght / arr[key] / arr.forEach(function(element, index){…}); / arr.map(function(element){…}); / arr.filter(function(element){return element > 5;}); = arr.every(function(element){return element > 5;});

Iterieren: default = Zugriff via Index / in = Property Namen / of = Werte

for(var i=0;i<arr.length;i++){ console.log(arr[i]);	for(var p **in** arr){ `${p} : ${arr[p]}`	for(var v **of** arr) { `value = ${v}`

Regex: /muster/flags **Flags:** g(lobal),i(gnore case), m(ultiline), u(nicode)
'John Smith'.replace(/(\w+)\s(\w+)/, '$2, $1'); // Smith, John
str.split(regx); // array of strings / str.match(regx); // null/string / regx.test(str); // bool

Scope: Ohne var = global, ansonsten im aktuelle Bereich sichtbar
ECMA6: let verhält sich wie var, mit dem Unterschied dass sie innerhalb einer Funktion nur in einem Block (z.B for-Loop) sichtbar ist.

Hoisting: Deklarationen werden vom Interpreter an den Anfang des Scopes verschoben (let Variable sind nicht betroffen) → Die Zuweisung wird nicht verschoben!

Bubbling: (event.target = effektives Event Objekt) Events werden im DOM Tree nach oben propagiert (disable → event.stopPropagation())

Use Strict: Striktere Interpretation welche komische Seiteneffekte unterbindet→"use strict";

DOM (HTML → DOM Tree → Render Tree)

Objekte: console, window, document.body, document.height, document.title

Funct:getElementById(), getElementsByClassName(), getElementsByTagName()

Events: function myFunc(event){…}; document.getElementById('id').onclick = myFunc / button.addEventListener("click", myFunc); / butten.removeEventListener('click', myFunc);

click, dblclick, mousedown, mouseup, mouseover, mouseenter, mouseleave, mousemove, keyup, keydown, keypress, copy, cut, paste, focus, select, load, resize, scroll

Load=Alles geladen / **Ready**: DOM Manipulation möglich, Bilder nicht geladen

JQUERY (V1: alte Browser, V2: kleinere Dateigrösse)

jQuery.noConflict(); (function($) { …. })(jQuery);	$(document).ready(function(){ … }); $(function() { … }); (Kurzform)

Funktionen: val(), text(), html(), attr(), hide(), remove(), append(), toggle(), addClass(), removeClass(), $("p").each(function(index){..}), $("p").on("eventName", function(event){..})

JSON → JSON.stringify(obj) , form.serialize(); / JSON.parse()

Beispiel: {"alter":"18", "name":"Regula"} (Double Quotes verwenden!)

Nan, Infinity werden zu NULL konvertiert

AJAX (Asynchronous JavaScript and XML)

```
$.ajax({ type: "POST", dataType: "json", url: "some/url/", data: JSON.stringify(postDataObj),
contentType: "application/json" }) /* dataType = Answer parsing (xml, html, json, text) */
.done(function(data, textstatus, jqXHR) { … })
.fail(function(jqXHR, textStatus, errorThrown){ … })
.always(function(data|jqXHR, textSta, jqHR|error){ … });
```

$.get("url", function(data) {…}	$.post("url", postData, function(data){…}

```
var form = $("#contactForm");
form.on("submit", function(event) {
  $.post(form.attr("action"), form.serialize(), function(data) {console.log(data) ;}); });
```

XHR: XMLHttpRequest → XMLHttpRequest.DONE = 4 / request.status = 200|201|304

```
var request = new XMLHttpRequest();
request.onreadystatechange = function(){
if(request.readyState === XMLHttpRequest.DONE && request.status === 200) {
var responseText = request.responseText; } };
request.open('POST', 'http://url/file', true);      var postData = JSON.stringify(postDataObj);
request.setRequestHeader("Content-Type", "application/json"); request.send(postData);
```

Handlebars Templating

Template:

```
<script id="template" type="text/x-handlebars-template">
  {{#each products}}
    <li>{{name}}<li>
    <span>{{#if specialOffer}}…{{else if}}…{else}…{{/if}}</span>
    <span>by {{#with seller}} {{this.name}} {{/with}} </span>
  {{/each}}
</script>
```

Data:

```
var productList = [{name:"Stabilo", specialOffer: true, seller:{name: "eBay"}},
{name:"Tea", specialOffer: false, seller:{name: "Amazon"}}, {…}, {…}];
```

Binding:

```
var templateSrc = $("#template").html();
var template = Handlebars.compile(templateSrc) ; //compile
var view = template({products: productList }) ; //bind
$("#container").html(view) ; // set rendered template
```

Static

```
<ul id="container"></ul>
```

HTTP

Cookies: document.cookie = "username=Foo; expires=Thu, 18 Dec 2013 12";

Same Orgin Policy = JS darf nur auf Server Zugreifen von welchem es geladen wurde. (Protocol, Domain, Port)

UCD: User Centered Design (Analyse → Modell → Spez. → Realisierung)

Farbenblinde, Nicht mehr als 4 Farben, Augenführung, Schriften, Kulturen

Usability ISO 9241-11: Effektiv, Effizient, Zufriedenheit

Prozess planen → Benutzungsumfeld analysieren & spezifizieren → Anforderungen an Produkt ableiten → Lösung produzieren → Lösung evaluieren gegen Anforderungen

Usability von Stone

Visibility: Der erste Schritt zum Ziel ist sichtbar

Affordance / Begreifbarkeit: Aktionsresultat ist vorhersehbar

Feedback: Es ist klar was passiert ist oder gerade passiert

Simplicity: Nicht mehr als nötig für die Aufgaben

Structure: Logische und konsistente Organisation der Inhalte

Consistency: Vorhersagbarkeit durch Konsistenz

Toleranz: Fehler vermeiden, Wiederherstellung vereinfachen

Accessability: Design für alle Personengruppen & Situationen

5 Ebenen Modell von Garrett

Oberfläche	Attraktiv, Vertrauenserweckend (Farben, Schriften, Icons)
Raster	Erwartungskonform, Effizient, Fehlertolerant (Augenführung, Ausrichtung)
Struktur	Aufgabengerecht, Fehlertolerant, Effizient (Navigation, Card Sorting)
Umfang	Zielgruppengerecht, effektiv (Anforderungsanalyse, Feature Reduktion)
Strategie	Marktgerecht (Personas entwickeln)