



HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

[Thema der Bachelorarbeit]

[Vor- und Zuname des/der Bachelorkandidaten/in]

Konstanz, [Datum]

BACHELORARBEIT

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

an der

Hochschule Konstanz

Technik, Wirtschaft und Gestaltung

Fakultät Informatik

Studiengang [Software-Engineering/Technische
Informatik/Wirtschaftsinformatik]

Thema: **[Thema der Bachelorarbeit]**

Bachelorkandidat: [Vor- und Zuname des/der Bachelorkandidaten/in], [Stra"se],
 [PLZ][Ort]

1. Pr"ufer: [Titel, Vor- und Zuname des 1. Pr"ufers]
2. Pr"ufer: [Titel, Vor- und Zuname des 2. Pr"ufers]

Ausgabedatum: [Datum]
Abgabedatum: [Datum]

Zusammenfassung (Abstract)

Thema: [Thema der Bachelorarbeit]

Bachelorkandidat: [Vor- und Zuname des/der Bachelorkandidaten/in]

Firma: [HTWG oder Firmenname]

Betreuer: [Titel, Vor- und Zuname des 1. Pr"ufers]

[Titel, Vor- und Zuname des 2. Pr"ufers]

Abgabedatum: [Datum]

Schlagworte: [Platz, für, spezifische, Schlagworte, zur, Ausarbeitung]

[Text der Zusammenfassung etwa 150 Worte. Es soll der L"osungsweg beschrieben sein.]

Ehrenw"ortliche Erkl"arung

Hiermit erkl"are ich *[Vor- und Zuname des/der Bachelorkandidaten/in]*,
geboren am [Datum]in [Geburtsort], dass ich

- (1) meine Bachelorarbeit mit dem Titel

[Thema der Bachelorarbeit]

bei der [HTWG oder Firmenname] unter Anleitung von [Titel, Vor- und Zuname des 1. Pr"ufers] selbst"andig und ohne fremde Hilfe angefertigt und keine anderen als die angef"uhrten Hilfen benutzt habe;

- (2) die "Ubernahme w"ortlicher Zitate, von Tabellen, Zeichnungen, Bildern und Programmen aus der Literatur oder anderen Quellen (Internet) sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erkl"arung rechtliche Folgen haben wird.

Konstanz, [Datum]

(Unterschrift)

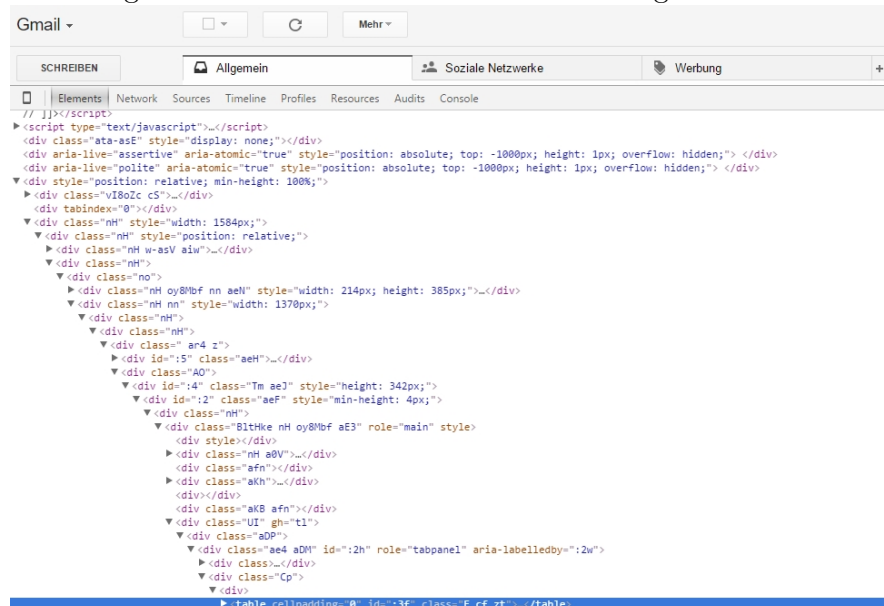
Chapter 1

Custom Elements

- TODO:
- Ausformulieren:
- Complete

Einführung

- Momentan: Suppe von Divs, die nicht aussagekräftig sind, siehe folgender Ausschnitt der Inbox der Google-Mail Webseite



```
// ]]></script>
<script type="text/javascript">...</script>
<div class="ata-asE" style="display: none;"></div>
<div aria-live="assertive" aria-atomic="true" style="position: absolute; top: -1000px; height: 1px; overflow: hidden;"></div>
<div aria-live="polite" aria-atomic="true" style="position: absolute; top: -1000px; height: 1px; overflow: hidden;"></div>
<div style="position: relative; min-height: 100%;">
  <div class="vIBo2c c5">...</div>
  <div tabindex="0"></div>
  <div class="NH" style="width: 1584px;">
    <div class="NH" style="position: relative;">
      <div class="NH w-asV aiw">...</div>
      <div class="NH">
        <div class="no">
          <div class="NH oy8Mbf nn aeH" style="width: 214px; height: 385px;">...</div>
          <div class="NH nn" style="width: 1370px;">
            <div class="NH">
              <div class="NH">
                <div class="NH">
                  <div class="ar4 z">
                    <div id="15" class="aeH">...</div>
                    <div class="A0">
                      <div id="14" class="Tm ae3" style="height: 342px;">
                        <div id="12" class="aeF" style="min-height: 4px;">
                          <div class="NH">
                            <div class="BltHke NH oy8Mbf ae3" role="main" style="position: relative;">
                              <div style="position: relative;">
                                <div class="NH a0V">...</div>
                                <div class="afn">...</div>
                                <div class="akh">...</div>
                                <div></div>
                                <div class="akB afn">...</div>
                              <div class="UI" gh="tl">
                                <div class="a0P">
                                  <div class="ae4 a0M" id="12h" role="tabpanel" aria-labelledby="12w">
                                    <div class="Cp">...</div>
                                  <div>...</div>
                                <div>...</div>
                              <div>...</div>
                            <div>...</div>
                          <div>...</div>
                        <div>...</div>
                      <div>...</div>
                    <div>...</div>
                  <div>...</div>
                <div>...</div>
              <div>...</div>
            <div>...</div>
          <div>...</div>
        <div>...</div>
      <div>...</div>
    <div>...</div>
  <div>...</div>
</div>
<table cellpadding="0" id="13f" class="F cf zt">...</table>
```

- Besser: Elemente, die semantisch aussagekräftig sind. So könnte die Google-Mail Webseite folgender Maßen aussehen

```

<hangout-module>
  <hangout-chat from="Paul, Addy">
    <hangout-discussion>
      <hangout-message from="Paul" profile="profile.png"
        profile="118075919496626375791" datetime="2013-07-17T12:02">
        <p>Feelin' this Web Components thing.</p>
        <p>Heard of it?</p>
      </hangout-message>
    </hangout-discussion>
  </hangout-chat>
<hangout-chat>...</hangout-chat>
</hangout-module>

```

- Custom Elements ermöglichen es
- neue DOM Elemente zu definieren
- Elemente zu definieren, die vorhandene Elemente erweitern
- eigene Funktionalitäten in einem Element zu bündeln
- die APIs vorhandener DOM Elemente zu erweitern

[Eric Bidelman 2013]

Neue Elemente registrieren

- Laut W3C Spezifikation muss ein Custom Element ein Bindestrich im Namen haben, z.B. `my-element` (<http://w3c.github.io/webcomponents/spec/custom/#concepts>)
- Ein neues Element wird mit der Funktion `var MyElement = document.registerElement('my-element');` registriert
- Als zweiter Parameter kann der `prototype` mit angegeben werden

```

var MyElement = document.registerElement('my-element', {
  prototype: Object.create(HTMLElement.prototype)
});

```

- Dadurch steht es in der Registry des Browsers, welche dazu verwendet wird um die Definitionen der Elemente aufzulösen
- Nachdem das Element registriert wurde, kann es per JavaScript oder HTML Deklaration verwendet werden

JavaScript

```
var myelement = document.createElement('my-element');
document.body.appendChild(myelement);
```

HTML

```
<div class="wrapper">
  <my-element><my-element>
</div>
```

[Developing Web Components 2015]

Vorteile von Custom Elements

- Unangemeldete, unregistrierte Custom Tags wie z.B. `<myelement>` benutzen das Interface `HTMLUnknownElement`
- Angemeldete, registrierte Custom Elements wie z.B. `<my-element>` benutzen das Interface `HTMLElement`
- Somit können für neue HTML Elemente eigene APIs erzeugt werden, indem eigene Eigenschaften und Methoden hinzugefügt werden

[Eric Bidelman 2015]

Nachteil

- Eventueller FOUC (Flash of unstyled content), da das Element schon im DOM steht, aber erst noch registriert werden muss
- Kann verhindert werden, in dem man den `:unresolved`-Selector benutzt und die Elemente ausblendet

```
my-element:unresolved{      display: none;  }
```

[Peter Gasston 2014]

Vorhandene Elemente erweitern (Type extensions)

- Statt neue Elemente zu erzeugen, können vorhandene auch erweitert werden
- So können native HTML Elemente erweitert werden
- Um einen erweitertes `button` zu erzeugen muss also folgendes gemacht werden:


```
var ButtonExtendedProto = document.registerElement('button-extended', {
  prototype: Object.create(HTMLButtonElement.prototype),
  extends: 'button'
});
```

- Ein erweitertes Element kann nun wie folgt via JavaScript oder HTML Deklaration verwendet werden:

JavaScript:

```
var buttonExtended = document.createElement('button', 'button-extended');

// Oder

var buttonExtended = new ButtonExtendedProto();
```

HTML:

```
<div class="wrapper">
  <button is="button-extended"></button>
</div>
```

[Eiji Kitamura 2014]

Verwendung bei Github

- Die “Latest commit” Angaben eines Repositories auf Github sind ein erweitertes time-Element (Type Extension Custom Element mit time-Element)
- Statt des Commit-Datums und der Zeit, wird - wenn JavaScript aktiviert ist - die berechnete Zeit seit dem letzten Commit angezeigt

```
▼ <span class="css-truncate css-truncate-target">
  <time datetime="2015-03-05T05:03:00Z" is="time-ago" title="5. März 2015, 06:03 MEZ">8 months ago</time>
</span>
```

Figure 1.1: Bild: Github Einsatz eines Custom Element

- Dabei dient das `time` Element als Basis
- Das `datetime` Attribut gibt die absolute Zeit des Commits an
- `is="time-ago"` ist die Erweiterung des `time` Elements
- Der Inhalt des `time` Elements zeigt die relative Zeit an

- Falls der Browser nun keine Custom Elements (mit Polyfill) unterstützt oder JavaScript deaktiviert ist, wird dennoch das “normale” `time` Element mit der absoluten Zeit angezeigt

[Eiji Kitamura 2014]

Eigenschaften und Methoden definieren

- Custom Elements machen erst so richtig Sinn, wenn man für diese auch eigene Eigenschaften und Methoden definieren kann
- Wie bei nativen HTML Elementen ist dies bei Custom Elements möglich, dies geschieht auf die gleiche Weise

```
// Methode definieren
ButtonExtendedProto.prototype.alert = function () {
    alert('foo');
};
```

```
// Eigenschaft definieren
ButtonExtendedProto.prototype.answer = 42;
```

[Developing Web Components 2015]

Custom Element Life Cycle Callbacks - TODO?

- Custom Elements bieten eine standardisierte API um verschiedene Methoden zu unterschiedlichen Zeitpunkten im “Leben” eines Custom Elements auszuführen. Diese ermöglicht es zu bestimmen, wie und wann ein bestimmter Code des Custom Elements ausgeführt wird.

createdCallback

- Wird ausgeführt, wenn eine Instanz des Custom Elements erzeugt wird.
Beispiel: `document.createElement('custom-element');`

attachedCallback

- Wird ausgeführt, wenn ein Custom Element dem DOM angehängt wird.
Beispiel: `document.body.appendChild();`

detachedCallback

- Wird ausgeführt, wenn ein Custom Element aus dem DOM entfernt wird. Beispiel: `document.body.removeChild()`;

attributeChangedCallback

- Wird ausgeführt, wenn ein Attribut eines Custom Elements geändert wird. Beispiel: `MyElement.setAttribute()`;

Beispiel mit button-extended

Anhand des `button-extended` Beispiels würde dies folgender Maßen funktionieren:

```
var ButtonExtendedProto = Object.create(HTMLElement.prototype);

ButtonExtendedProto.createdCallback = function() {...};
ButtonExtendedProto.attachedCallback = function() {...};

var ButtonExtended = document.registerElement('button-extended', {prototype: ButtonExtendedProto});
```

[Raoul Schaffranek 2014]

Styling von Custom Elements

- Custom Elements können wie native HTML Elemente gestyled werden

```
my-element {
  foo: bar;
}
```

- Element-Erweiterungen können per Attribut-Selektor angesprochen werden

```
[is="button-extended"] {
  foo: bar;
}
```

[Developing Web Components 2015]

Browserunterstützung

- Noch nicht standardisiert, sind noch ein Working Draft (<http://w3c.github.io/webcomponents/spec/>)
- Deshalb bisher auch nur in Chrome und Opera unterstützt

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8		38	43					4.3	
9		39	44	7.1		7.1		4.4	
10		40	45	8	31	8.4		4.4.4	
11	12	41	46	9	32	9	8	4.4	46
	13	42	47		33				
		43	48		34				
		44	49						

Figure 1.2: Bild: Browserunterstützung von Custom Elements

[Can I Use 2015]

Quellen

- [Developing Web Components 2015] Jarrod Overson & Jason Strimpel, Developing Web Components, O'Reilly 2015, S.127-138
- [Eiji Kitamura 2014] Eiji Kitamura, Introduction to Custom Elements, <http://webcomponents.org/articles/introduction-to-custom-elements/>
- <http://w3c.github.io/webcomponents/spec/custom/>
- [Eric Bidelman 2013] Eric Bidelman, Custom Elements, <http://www.html5rocks.com/en/tutorials/>
- [Can I Use 2015] Can I Use, <http://caniuse.com/#feat=custom-elements>
- [Peter Gasstton 2015] Peter Gasstton, A Detailed Introduction To Custom Elements, <http://www.smashingmagazine.com/2014/03/introduction-to-custom-elements/>
- [Raoul Schaffranek 2014] Raoul Schaffranek, Web Components – eine Einführung, <https://blog.selfhtml.org/2014/12/09/web-components-eine-einfuehrung/>

Chapter 2

Was anderes

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Inventore id minus dolorem architecto, neque voluptatibus et, culpa fugiat ad, voluptas perspiciatis magni earum. Ex ipsum eum mollitia, eius sint, est.