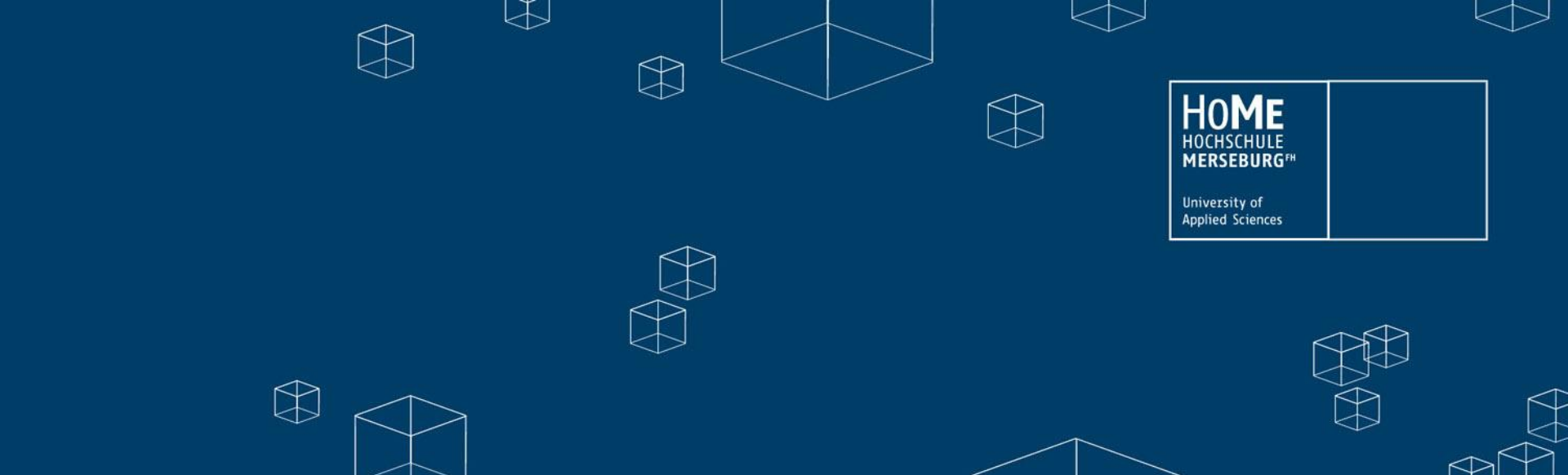


Interaktive XML-Verarbeitung im Browser mit Saxon-JS und XSLT 3.0

tekom-Jahrestagung 2017 – Stuttgart, 25. Oktober

Dr. Thomas Meinike

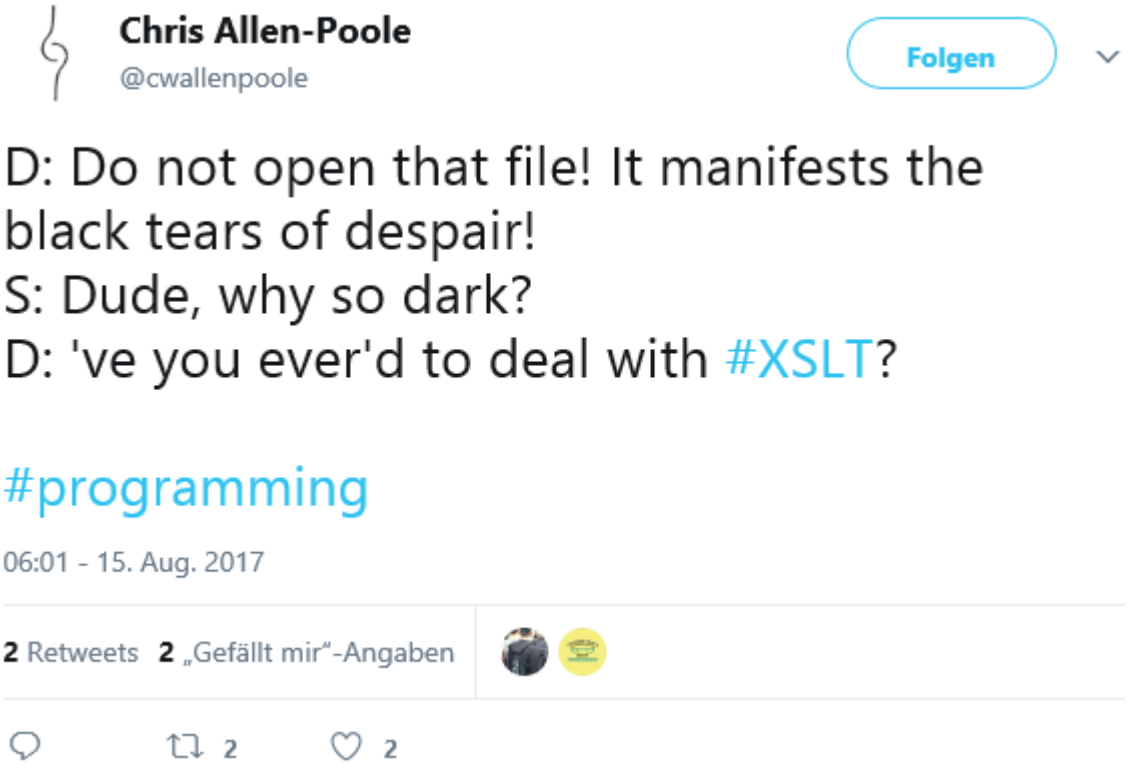
Hochschule Merseburg | FB Wirtschaftswissenschaften und Informationswissenschaften

web.hs-merseburg.de/~meiniket/

thomas.meinike@hs-merseburg.de

Vorwarnung

➔ Es geht um XSLT & Co. – solche Probleme sollte man nicht haben … ;)



HOME HOCHSCHULE MERSEBURG University of Applied Sciences

Motivation

- Direkte Nutzung von XSLT in Browsern ist für die Web-Entwicklung mangels Unterstützung der Hersteller meistens keine Option mehr.

- Der Einsatz von XSLT zur clientseitigen Verarbeitung von XML-Daten und alternativen Formaten bietet sich in unterschiedlichen Kontexten dennoch an.

- Saxon-JS ist das jüngste Produkt aus dem Hause Saxonica unter Federführung von Michael Kay, u. a. als Herausgeber der XSLT-Spezifikation bekannt (aktuelle Version 3.0 vom Juni 2017).

- Neben XSLT 3.0 unterstützt Saxon-JS den Standard XPath 3.1 (vom März 2017) mit einer Vielzahl an Funktionen (fn:… / xs:… usw.).

Einstieg in Saxon-JS 1/5

- JavaScript-basierte Laufzeitumgebung zur Transformation von XML- und JSON-Daten im Web-Browser, also in HTML-Dokumente eingebunden.

- Version 1.0.0 ist im Februar 2017 auf saxonica.com erschienen. Danach folgten zwei Wartungsupdates: 1.0.1 (Juli) / 1.0.2 (Oktober).
  Beispielanwendungen wurden mit 1.0.0 entwickelt und bis zur aktuellen Version getestet.

- Eingebunden wird als Hauptmodul die ~230 KB umfassende Bibliothek SaxonJS.min.js.

Einstieg in Saxon-JS 2/5

- Der grundsätzliche technologische Unterbau stammt von Saxon-CE (siehe tekom-Vortrag des Autors von 2013).

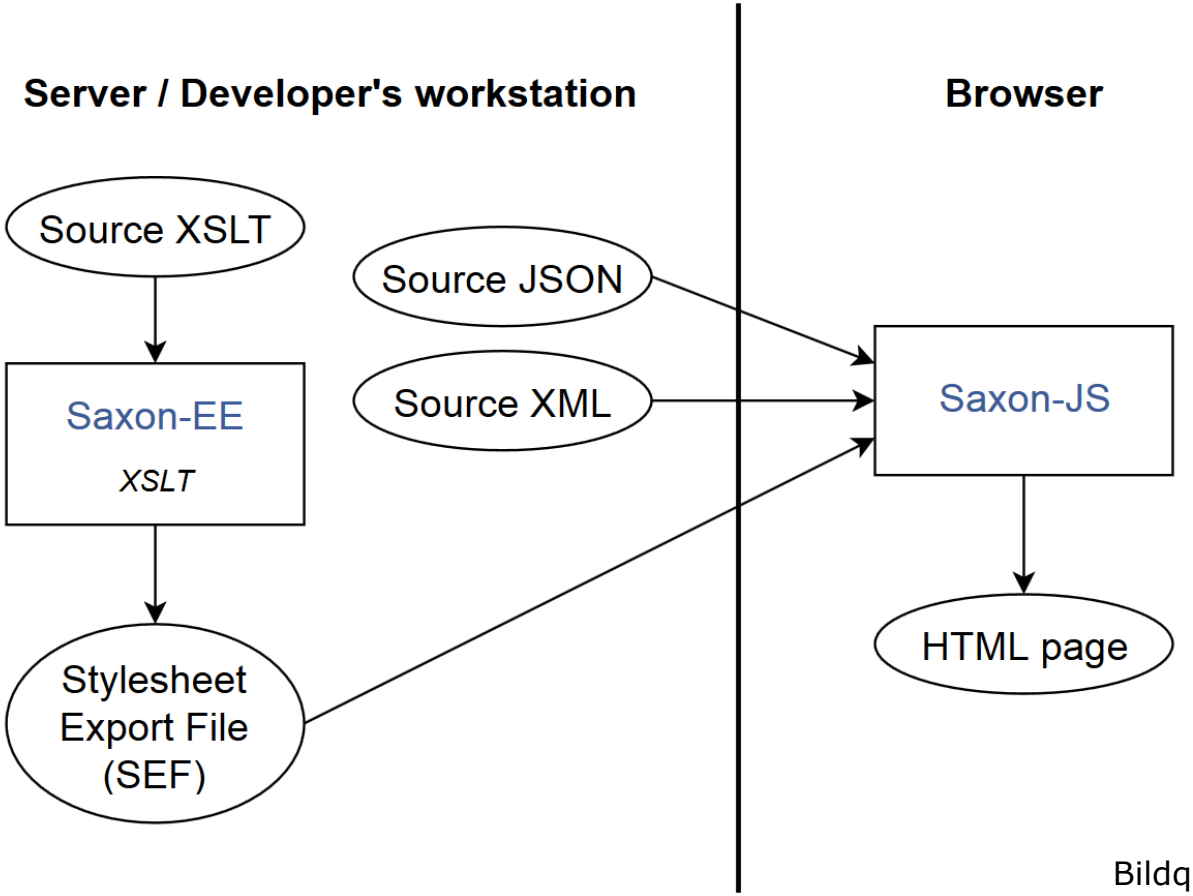
- Im Unterschied zum CE-Konzept werden die Eingabedaten nicht direkt mit XSLT-Stylesheets prozessiert, sondern über vorkompilierte SEF-Dateien.

- SEF = Stylesheet Export File, ein optimiertes XML-Format für den Transformationsbaum. Zur Erstellung wird der kommerzielle Prozessor Saxon-EE ab Version 9.7 (standalone oder in <oXygen/>) benötigt.

- Die SEF-Dateien selbst laufen innerhalb von Anwendungen ohne weitere Lizenz auf einem Web-Server (als application/xml) oder lokal über das Dateisystem.

Einstieg in Saxon-JS 3/5

➔ Anwendungsarchitektur:



Bildquelle: Saxon-JS-Dokumentation

Einstieg in Saxon-JS 4/5

➔ Kompilation eines XSLT-Stylesheets → SEF über die Kommandozeile, praktikabel mit Batch-Datei oder Shellscript gesteuert:

```
java -jar X:\Pfad_zu\saxon9ee.jar -t -xsl:name.xsl -export:name.sef -target:JS -nogo -relocate:on
```

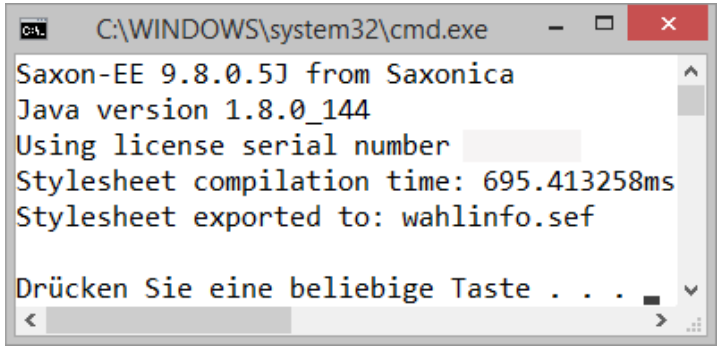
(alternativ: java net.sf.saxon.Transform –t …)

➔ Bedeutung der Transformationsparameter:

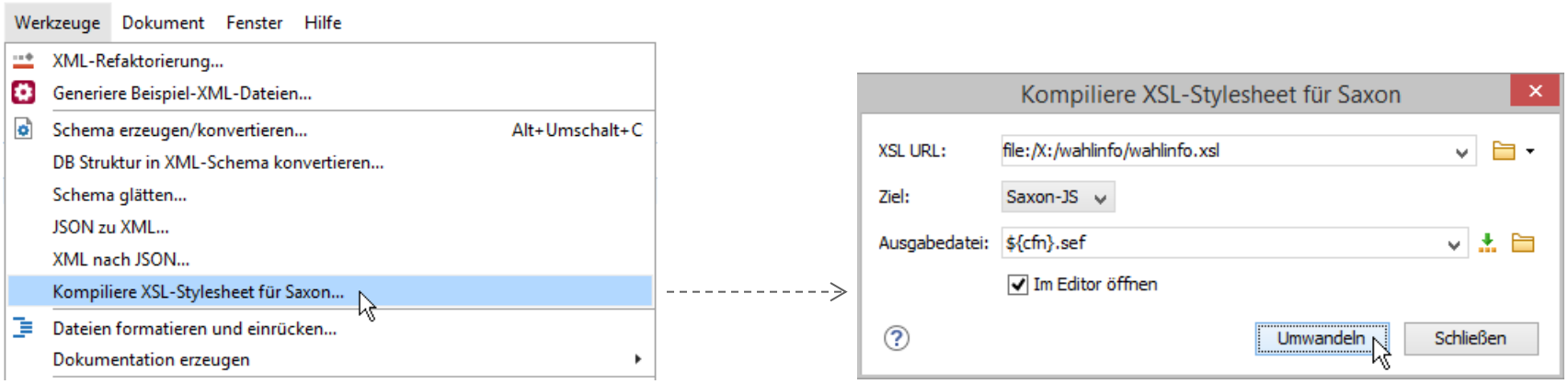
-t	Anzeige von Fortschrittsmeldungen
-xsl	Ausgangs-Stylesheet (Quelle)
-export	Exportiertes SEF-Dokument (Ziel)
-target	Bindet IXSLT-Unterstützung für Zielsprache JS ein
-nogo	Unterdrückt Ausführungsversuche des Stylesheets
-relocate	Ermöglicht mit Wert **on** die SEF-Nutzung unabhängig von den Pfaden bei Entwicklung bzw. Ausführung (erst ab Saxon-EE 9.8)

Einstieg in Saxon-JS 5/5

➔ Kompilation mit Saxon-EE an der Kommandozeile:



➔ Kompilation über den <oXygen/> XML Editor ab 19.x:



HOME HOCHSCHULE MERSEBURG University of Applied Sciences

Praktische Entwicklung 1/16

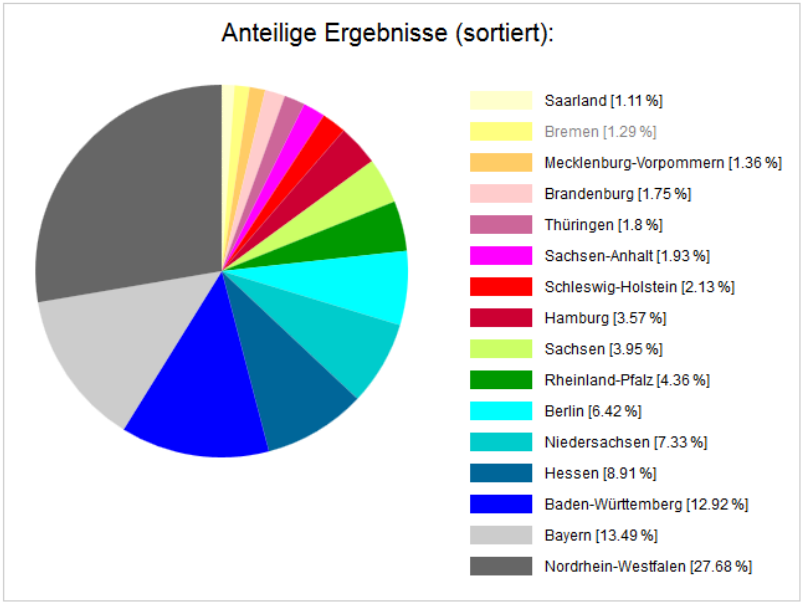
➔ Projekt »Studierende« aus dem CE-Vortrag mit Saxon-JS adaptiert:

Studierende

zum Wintersemester …

○ 2010/11 ○ 2011/12 ○ 2012/13 ○ 2013/14 ○ 2014/15 ○ 2015/16 ◉ 2016/17 ☑ Grafik anzeigen

Bundesland	Semester 2016/17
Baden-Württemberg	362194
Bayern	378203
Berlin	180096
Brandenburg	49017
Bremen	36228
Hamburg	100133
Hessen	249810
Mecklenburg-Vorpommern	38008
Niedersachsen	205497
Nordrhein-Westfalen	776114
Rheinland-Pfalz	122119
Saarland	31182
Sachsen	110849
Sachsen-Anhalt	54192
Schleswig-Holstein	59758
Thüringen	50516
Gesamt	**2803916**



[Projekt by T. Meinike 2013…17 | Umgesetzt mit Saxon-JS (sowie Saxon-CE) | Datenquelle: Statistisches Bundesamt]

Praktische Entwicklung 2/16

➔ Projekt »Studierende« – HTML-Struktur:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="UTF-8" /> <title>Studierende in Deutschland</title>
    <link rel="stylesheet" href="studierende.css" type="text/css" />
    <script type="text/javascript" src="../SaxonJS/SaxonJS.min.js"></script>
    <script type="text/javascript" src="action.js"></script>
  </head>
  <body>
    <header id="header">
      <h1>Studierende</h1>
    </header>
    <main id="ausgabe">
      <section id="tabelle"></section>
      <section id="grafik"></section>
    </main>
    <footer id="footer"></footer>
    <noscript><p>Infotext …</p></noscript>
  </body>
</html>
```

Ausgabe-Blöcke

Saxon-JS-Bibliothek

Saxon-JS-Initialisierung
+ eigene Funktionen

Praktische Entwicklung 3/16

➔ Projekt »Studierende« – JavaScript-Code in action.js:

```
// Initialisierung von Saxon-JS
window.onload = function()
{
  SaxonJS.transform(
  {
    sourceLocation:     "studierende.xml",
    stylesheetLocation: "studierende.sef"
  });
}
```

```
<bundesland>
  <name kurz="ST">Sachsen-Anhalt</name>
  <daten>
    <wert>54078</wert>
    <wert>55761</wert>
    <wert>55876</wert>
    <wert>55954</wert>
    <wert>54989</wert>
    <wert>54954</wert>
    <wert>54192</wert>
  </daten>
</bundesland>
```

XML-Quelle und SEF-Dokument

```
// Eigene Funktion(en)
function FixSVG4Edge()
{
  // diese wird später erläutert …
}
```

Weitere Optionen sind möglich, z. B. Übergabe-Parameter an das Stylesheet (dort mit xsl:param zu empfangen):

```
stylesheetParams: {"name1" : "wert1",
"name2" : "wert2"}
```

Praktische Entwicklung 4/16

➔ Projekt »Studierende« – XSLT-Codeteil:

Hier Ausgabe in den **#header**-Block des HTML-Dokuments.

```
<xsl:template match="studierende">
  <!-- Header (Formular) -->
  <xsl:result-document href="#header" method="ixsl:append-content">
    <h2>zum {zeitraum/@info} …</h2>
    <form name="auswahl">
      <xsl:for-each select="zeitraum/semester">
        <xsl:variable name="pos" select="fn:position()" as="xs:integer"/>
        <input id="radio{$pos}" name="semester" type="radio" value="{.}"
          onclick="setTimeout(FixSVG4Edge, 0)"/>
        <label for="radio{$pos}">{' ' || .}</label>
      </xsl:for-each>
      <input id="gronoff" type="checkbox" checked="checked"/>
      <label for="gronoff"> Grafik anzeigen</label>
    </form>
  </xsl:result-document>

 <!-- Tabelle | Grafik | Footer via xsl:result-document erzeugen … (+140 Zeilen) -->
</xsl:template>
```

xsl:result-document + **href** wird für die Ausgaben in vorgefertigte Blöcke genutzt.

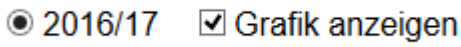
ixsl kennzeichnet die Erweiterungen für Interactive XSLT.

Praktische Entwicklung 5/16

➔ Projekt »Studierende« – Interaktionen:

– Grafik mit Checkbox aus/-einblenden, hier über ixsl:onclick, analog zu:

document.getElementById('grafik').style.display = 'none' | 'block';




```
<xsl:template match="input[@type eq 'checkbox'][@id eq 'gronoff']"
  mode="ixsl:onclick">
  <xsl:variable name="aktobj" select="." as="element()"/>
  <ixsl:set-style object="id('grafik')" name="display"
    select="if (ixsl:get($aktobj, 'checked') eq xs:boolean('false'))
    then 'none, else 'block'"/>
</xsl:template>
```

Hinweis: **ixsl:set-style** ist neu unter Saxon-JS und kürzt die ebenfalls noch verfügbare Syntax mit ixsl:property und name="style.display" etwas ab.

Praktische Entwicklung 6/16

➔ Projekt »Studierende« – Anpassungen für Saxon-JS:

– Nutzung der nativen XPath-Funktionen math:sin(…) und math:cos(…) sowie math:pi() zur Berechnung der Kreisdiagramm-Segmente (Pfade):

```
<xsl:variable name="punkt_xs" select="fn:round-half-to-even(math:cos($startwinkel
              * math:pi() div 180) * $kreis_r + $kreis_x, 2)" as="xs:double"/>
<xsl:variable name="punkt_ys" select="fn:round-half-to-even(math:sin($startwinkel
              * math:pi() div 180) * $kreis_r + $kreis_y, 2)" as="xs:double"/>
```

– Die 16 Farben für die Bundesländer wurden mit der neuen Array-Syntax […] eingebunden, zuvor als Sequenz (…):

```
<xsl:variable name="farben" select="['#FFC','#FF0', ...]" as="array(*)"/>
```

und im Kontext über die jeweilige Position $i zugewiesen:

```
<xsl:variable name="farbe" select="$farben($i)" as="xs:string"/>
```

Alternativ ist auch array:get($farben, $i) möglich.

Praktische Entwicklung 7/16

➔ Projekt »Studierende« – Anpassungen für Saxon-JS:

– Konsequenter Einsatz der neuen Text-Value-Templates <x>{Inhalt}</x> als Alternative zu <x><xsl:value-of select="Inhalt"/></x>, z. B.:

```
<tr>
  <td>{name}</td><td>{daten/wert[$sem_index]}</td>
</tr>
```

Dazu ist entweder global bei xsl:stylesheet oder beim konkreten Element, etwa xsl:text, zu setzen: expand-text="yes".

– Ebenfalls sehr nützlich ist die neue Kurzform || für String-Verknüpfungen:

```
<g id="{'g_' || name/@kurz}">…</g>
```

statt:

```
<g id="{fn:concat('g_', name/@kurz)}">…</g>
```

Praktische Entwicklung 8/16

➔ Projekt »Studierende« – Anpassungen für Saxon-JS:

– Ein Problem ergab sich bei der SVG-Ausgabe im MS-Browser Edge unter Windows 10. Alle Füllungen, Konturen usw. erschienen schwarz. Liegt nicht an Saxon-JS, sondern an der Edge-Eigenart dynamisch große Attributnamen wie FILL, STROKE, … ins DOM zu schreiben. FixSVG4Edge() löst es:

```
function FixSVG4Edge() {
  if(navigator.userAgent.indexOf("Edge") != -1) {
    var svgobj = document.getElementById("piechart");
    var svgcode = svgobj.innerHTML;
    svgcode = svgcode.replace(
      /FILL=|STROKE=|STROKE-WIDTH=|OPACITY=|FILL-OPACITY=|STROKE-OPACITY=|STROKE-
      DASHARRAY=|FONT-SIZE=|FONT-FAMILY=|FONT-WEIGHT=|FONT-STYLE=|TEXT-DECORATION=/g,
      function(attr) { return attr.toLowerCase(); }
    );
    svgobj.innerHTML = svgcode;
  }
}
```

Praktische Entwicklung 9/16

➔ Projekt »Bierkonsum« – Zugriff auf JSON-Daten über neuen Map-Typ:

– Statistische Daten wurden in JSON-Form aufbereitet (daten.json):

```
{"daten": {
   "info":    "Bierkonsum in Deutschland",
   "titel":   "Statista: Deutscher Brauer-Bund – …",
   "quelle":  "https://de.statista.com/statistik/daten/studie/29630/umfrage/…",
   "einheit": "1000 hl",
   "eintrag": [
     {
       "jahr":  1960,
       "menge": 52633
     },
     { … },
     {
       "jahr":  2016,
       "menge": 85532
     }
   ]
}}
```

```
// Initialisierung von Saxon-JS ohne XML-Quelle
window.onload = function()
{
  SaxonJS.transform(
  {
    stylesheetLocation: "bierkonsum_json.sef",
    initialTemplate:    "main"
  });
}
```

– … und mit fn:json-doc(…) im main-Template als Map geladen:

```
<xsl:variable name="json" select="fn:json-doc('daten.json')" as="map(*)"/>
```

Praktische Entwicklung 10/16

➔ Projekt »Bierkonsum« – Zugriff auf JSON-Daten über neuen Map-Typ:

– Abfrage der Daten, z. B. des info-Wertes, wahlweise:

```
<xsl:variable name="info" select="$json?daten?info" as="xs:string"/>
<xsl:variable name="info" select="$json('daten')('info')" as="xs:string"/>
```

– Sortierung der Einträge Jahr/Menge nach Jahreszahlen → $eintraege:

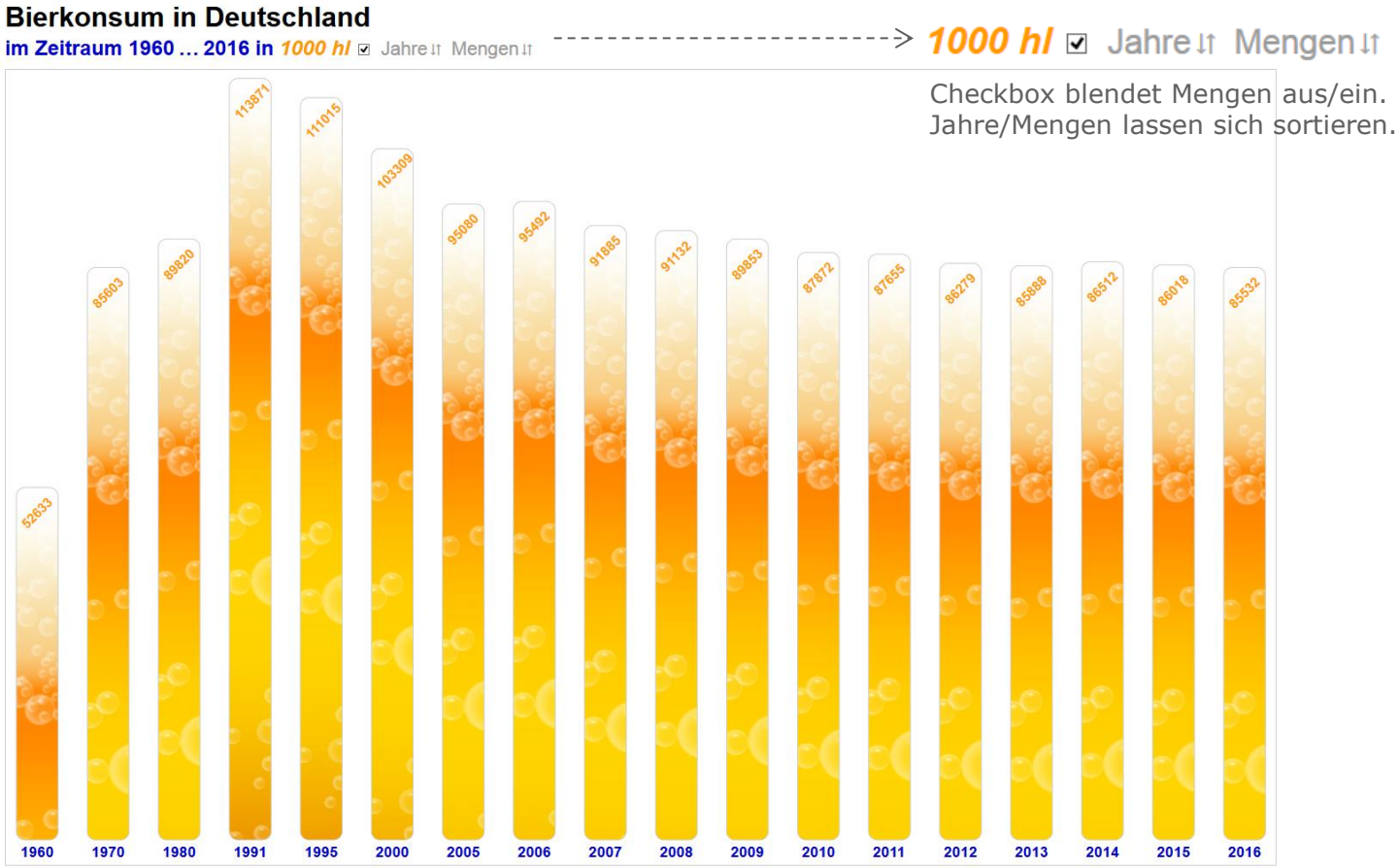
```
<xsl:variable name="eintraege" as="map(xs:string, xs:double)*">
  <xsl:for-each select="$json?daten?eintrag?*">
    <xsl:sort select="if($modus eq 'jahr') then ?jahr else ?menge"
              data-type="number" order="{$order}"/>
    <xsl:for-each select=".">
      <xsl:map>
        <xsl:map-entry key="'jahr'" select="?jahr"/>
        <xsl:map-entry key="'menge'" select="?menge"/>
      </xsl:map>
    </xsl:for-each>
  </xsl:for-each>
</xsl:variable>
```

Neu: xsl:map + xsl:map-entry

– Weitere Ausgabeoperationen erzeugen als Ergebnis eine SVG-Balkengrafik …

Praktische Entwicklung 11/16

➔ Projekt »Bierkonsum« – Zugriff auf JSON-Daten über neuen Map-Typ:



Praktische Entwicklung 12/16

➔ Projekt »Bierkonsum« – Zugriff auf JSON-Daten über neuen Map-Typ:

– Einsatz des neuen Elements xsl:iterate zum Durchlaufen der Datensätze:

```
<!-- Mengen als Gruppe zum Aus-/Einblenden via Checkbox -->
<g id="mengen">
  <xsl:iterate select="1 to $anzahl">
    <xsl:variable name="pos" select="fn:current()" as="xs:integer"/>
    <xsl:variable name="h_rect" select="fn:round($eintraege?menge[$pos] div $max_menge
                                        * $max_hoehe)" as="xs:double"/>
    <xsl:variable name="x_info" select="23 + ($pos - 1) * 85" as="xs:integer"/>
    <xsl:variable name="y_info" select="$max_hoehe - $h_rect + 60" as="xs:double"/>
    <text class="info" x="{$x_info}" y="{$y_info}"
          transform="rotate(-45,{$x_info},{$y_info})">{$eintraege?menge[$pos]}</text>
  </xsl:iterate>
</g>
```

Praktische Entwicklung 13/16

➔ Projekt »Wahlinfo« – Zugriff auf XML-Daten zur Bundestagswahl:

– Der Verein wahlinfo+ e. V. stellt unter deutschlandwaehlt.de zusätzlich Open Data zur Verfügung → Idee für eine Saxon-JS-Anwendung:

– **XML-Struktur (2-stufig)**:

(1) Übersicht zu allen Parteien

```
<partylist>
  <meta>…</meta>
  <data> <!-- … -->
    <party id="179" modified="1500469834">
      <name>Partei für Arbeit, Rechtsstaat,
        Tierschutz, Elitenförderung und
        basisdemokratische Initiative</name>
      <shortname>Die PARTEI</shortname>
      <data>http://deutschlandwaehlt.de/partei/
        die-partei/export_xml</data>
    </party> <!-- … -->
  </data>
</partylist>
```

(2) Details einer bestimmten Partei

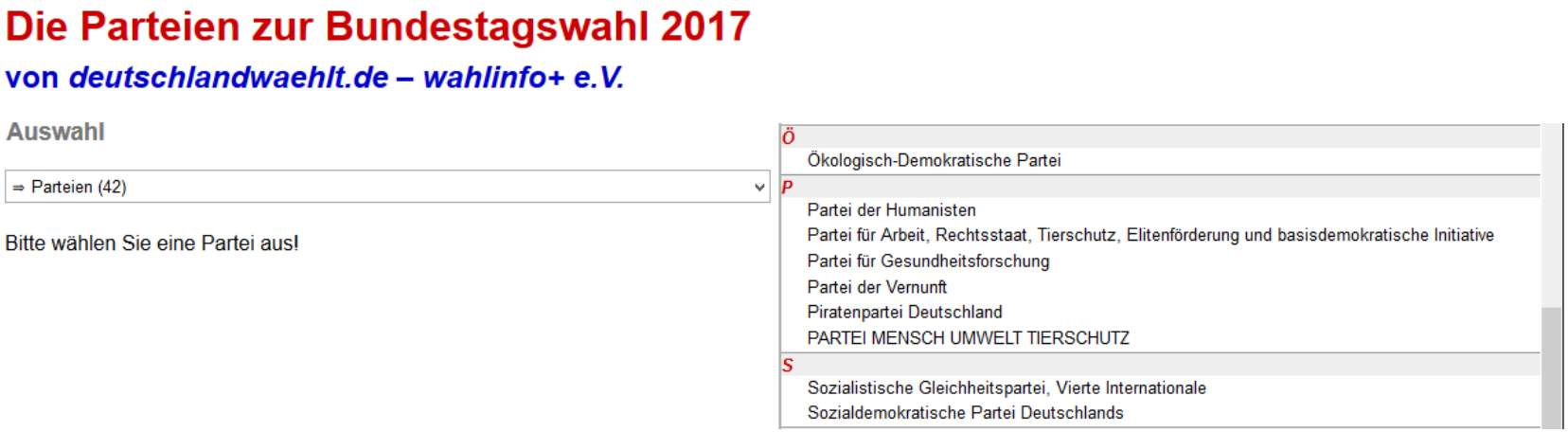
```
<party id="179">
  <meta>...</meta>
  <data>
    <name>Partei für Arbeit, …</name>
    <shortname>Die PARTEI</shortname>
    <program>http://…/…btw17.pdf</program>
    <homepage>https://…</homepage>
    <email>mail@die-partei.de</email>
    <twitter>…/DiePARTEI</twitter>
    <facebook>…/DiePARTEI</facebook>
    <instagram…/diepartei/</instagram>
    <address>… 10965 Berlin</address>
    <candidates>…</candidates>
  </data>
</party>
```

Praktische Entwicklung 14/16

➔ Projekt »Wahlinfo« – Zugriff auf XML-Daten zur Bundestagswahl:

– **Schritt 1**: Ausgeben der (42) Parteinamen in eine alphabetisch sortierte und nach Anfangsbuchstaben gruppierte Auswahlliste.

– Zugriff auf die XML-Ressource mit der Funktion fn:doc(…) und zusätzliches PHP-Skript [liest als Proxy die externen Daten über file_get_contents(…) ein]. Dabei wurde zur Fehlerbehandlung auch xsl:try / xsl:catch verwendet.



Praktische Entwicklung 15/16

➔ Projekt »Wahlinfo« – Zugriff auf XML-Daten zur Bundestagswahl:

– **Schritt 2**: Beim Auswählen einer konkreten Partei wird die Nutzeraktion über ixsl:onchange getriggert und die jeweils gelieferten Daten wie Adressen, Links zu PDF-Programmen sowie die verfügbaren E-Mail-, Web- und Social-Media-Kontakte werden mit Änderungsdatum ausgeliefert.

```
<xsl:template match="select[@id eq 'parteien']" mode="ixsl:onchange">
  <xsl:variable name="aktpar" select="ixsl:source()//party[@id=$aktval]/data"
    as="xs:string"/>
  <xsl:variable name="xmldoc" select="'proxy.php?xmldoc=' || fn:substring-after($aktpar,
   'http://deutschlandwaehlt.de/')" as="xs:string"/>
  <xsl:variable name="partei" as="document-node()" select="fn:doc($xmldoc)"/>
  <!-- Ergebnis-Templateaufruf zur jeweiligen Partei-Auswahl -->
  <xsl:apply-templates select="$partei/party"/>
</xsl:template>
```

Praktische Entwicklung 16/16

➔ Projekt »Wahlinfo« – Zugriff auf XML-Daten zur Bundestagswahl:

– **Ergebnis** für einen Datensatz:

Auswahl

Partei für Arbeit, Rechtsstaat, Tierschutz, Elitenförderung und basisdemokratische Initiative

Informationen

Partei:	Partei für Arbeit, Rechtsstaat, Tierschutz, Elitenförderung und basisdemokratische Initiative
Kurzform:	Die PARTEI
Adresse:	Kopischstraße 10 10965 Berlin
Programm:	PDF-Version
Kontakte:	E-Mail | Facebook | Instagram | Twitter | Web

Kandidaten (1)

Name:	Serdar Somuncu
Kontakte:	Facebook | Twitter | Web

Änderungsdatum: 19.07.2017

Hinweise 1/2

➔ Saxon-JS nutzt XSLT 3.0 Basic Level – aktuell nicht unterstützt:

- Streaming (für sehr große XML-Dokumente, neu im Standard)
- Serialisierung von Ergebnissen
- Schema-Validierung von Ausgaben
- Higher-Order Functions, inkl. anonyme Inline-Funktionen function(){…}
- xsl:assert zur Prüfung von Annahmen
- XPath-Funktionen fn:… zum Teil in ihren Argumenten eingeschränkt
- Weitere Detailinformationen unter:
  http://www.saxonica.com/saxon-js/documentation/index.html#!conformance

Hinweise 2/2

➔ Wesentliche Namensräume und <oXygen/>-Unterstützung (ixsl):

```
<xsl:stylesheet version="3.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:ixsl="http://saxonica.com/ns/interactiveXSLT"
  xmlns:math="http://www.w3.org/2005/xpath-functions/math"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:array="http://www.w3.org/2005/xpath-functions/array"
  xmlns:map="http://www.w3.org/2005/xpath-functions/map"
  xmlns:err="http://www.w3.org/2005/xqt-errors"
  xmlns:fn="http://www.w3.org/2005/xpath-functions"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:js="http://saxonica.com/ns/globalJS"
  xmlns="http://www.w3.org/1999/xhtml"
  exclude-result-prefixes="#all"
  extension-element-prefixes="ixsl"
  expand-text="yes">
```

```
<xsl:result-document href="#grafik" method="">
  <xsl:variable name="eintraege" as="map(xs:st
    <xsl:for-each select="$json?daten?eintrag?
      <xsl:sort select="if($modus eq 'jahr') t
      <xsl:for-each select=".">
        <xsl:map>
          <xsl:map-entry key="'jahr'" select="
          <xsl:map-entry key="'menge'" select=
        </xsl:map>
```

◇ ixsl:append-content
◇ ixsl:replace-content

Fazit und Ausblick

- Saxon-JS bereichert die Web-Entwicklung mit Fokus auf XML- oder JSON-Daten.

- Bei entsprechender Akzeptanz können interessante Anwendungen entstehen.

- Der breite Unterbau in Form der aktuellen Spezifikationen von XSLT und XPath konnte hier nur angerissen werden. Weitere Ansätze warten auf ihre Erforschung und Nutzung: HTTP-Requests (GET, POST, …), dynamische XPath-Zugriffe über xsl:evaluate uvm.

- Entwickler finden auf der Saxonica-Plattform aktuelle Informationen und Problemlösungen und können sich mit dem Team und anderen Interessenten austauschen.

Literatur und Ressourcen 1/2

Cagle, K.: Why You Should Be Using XSLT 3.0.
https://www.xml.com/articles/2017/02/14/why-you-should-be-using-xslt-30/

Kay, M.: Transforming JSON using XSLT 3.0. In: XML Prague 2016 – Conference Proceedings, S. 167-183. http://archive.xmlprague.cz/2016/files/xmlprague-2016-proceedings.pdf

Lockett, D. und Kay, M.: Saxon-JS: XSLT 3.0 in the Browser. In: Proceedings of Balisage: The Markup Conference 2016, DOI: 10.4242/BalisageVol17.Lockett01.
https://www.balisage.net/Proceedings/vol17/html/Lockett01/BalisageVol17-Lockett01.html

Lumley, J., Lockett, D. und Kay, M.: XPath 3.1 in the Browser. In: XML Prague 2017 – Conference Proceedings, S. 1-18. http://archive.xmlprague.cz/2017/files/xmlprague-2017-proceedings.pdf

Saxonica: Saxon-JS. http://saxonica.com/html/saxon-js/ + http://www.saxonica.com/saxon-js/documentation/

Saxonica: Developer Community. https://saxonica.plan.io/projects/saxon-js/

W3C: XSL Transformations (XSLT) Version 3.0. http://www.w3.org/TR/xslt-30/

W3C: XML Path Language (XPath) Version 3.1. http://www.w3.org/TR/xpath-31/

Literatur und Ressourcen 2/2

Meinike, T.: 3.0-Updates von XSLT und XPath auf einen Blick. In: tekom, Gesellschaft für technische Kommunikation e. V., Tagungsband zur Jahrestagung 2012, S. 341-343.

Meinike, T.: XSLT 2.0 im Browser mit Saxon-CE. In: tekom, Gesellschaft für technische Kommunikation e. V., Tagungsband zur Jahrestagung 2013, S. 338-341.

Meinike, T.: Studierende in Deutschland. http://datenverdrahten.de/xslt3/saxon-js/studis/
+ **Statistisches Bundesamt (Destatis)**: Studierende – Insgesamt nach Bundesländern und tiefer gegliederten Angaben. https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/BildungForschungKultur/Hochschulen/Tabellen/StudierendeInsgesamtBundeslaender.html

Meinike, T.: Bierkonsum in Deutschland. http://datenverdrahten.de/xslt3/saxon-js/bierkonsum/
+ **Statista**: Deutscher Brauer-Bund – Konsum von Bier in Deutschland in den Jahren 1960 bis 2016 (in 1.000 Hektoliter). https://de.statista.com/statistik/daten/studie/29630/umfrage/bierverbrauch-bierkonsum-in-deutschland/

Meinike, T.: Die Parteien zur Bundestagswahl 2017. http://datenverdrahten.de/xslt3/saxon-js/wahlinfo/
+ **wahlinfo+ e. V.**: Deutschland wählt – Bundestagswahl 2017. http://deutschlandwaehlt.de/

Feedback erwünscht …

- URL direkt aufrufen (auch nach der Tagung möglich):
  http://in29.honestly.de

- Oder QR-Code scannen:

Danke für Ihr Interesse!