SVG-Aktionsprogrammierung

– Mit DOM-Methoden vom Ereignis zum Effekt

Dr. Thomas Meinike

thomas.meinike@et.fh-merseburg.de
http://www.et.fh-merseburg.de/person/meinike/

Fachhochschule Merseburg
Fachbereich Elektrotechnik, Informationstechnik und Medien
Studiengang „Kommunikation und Technische Dokumentation“

[Vortrag auf der tekom-Jahrestagung 2003 in Wiesbaden – gehalten am 20./21.11.2003]

Überblick

- Ziel des Vortrags
- SVG in 5 Minuten
- Einbindung von Skripten
- Schreiben eigener Funktionen
- Reaktion auf Ereignisse
- Wichtige SVG-DOM-Techniken
- Demonstrationen
- Zusammenfassung und Ausblick

Ziel des Vortrags

- Vermittlung von Grundlagen
  zur Entwicklung dynamischer, interaktiver SVG-Dokumente.

- Basis:
  SVG statisch, Kenntnisse zu Elementen, Attributen,
  CSS-Eigenschaften, XML-Dokumentstruktur.

- Programmierung:
  clientseitige Skriptsprachen (JavaScript, ECMAScript),
  Event-Handling,
  Objektmodell (W3C-SVG-DOM).

- Verwandte Konzepte:
  DHTML, ActionScript (Flash/SWF).

SVG in 5 Minuten – Was ist SVG?

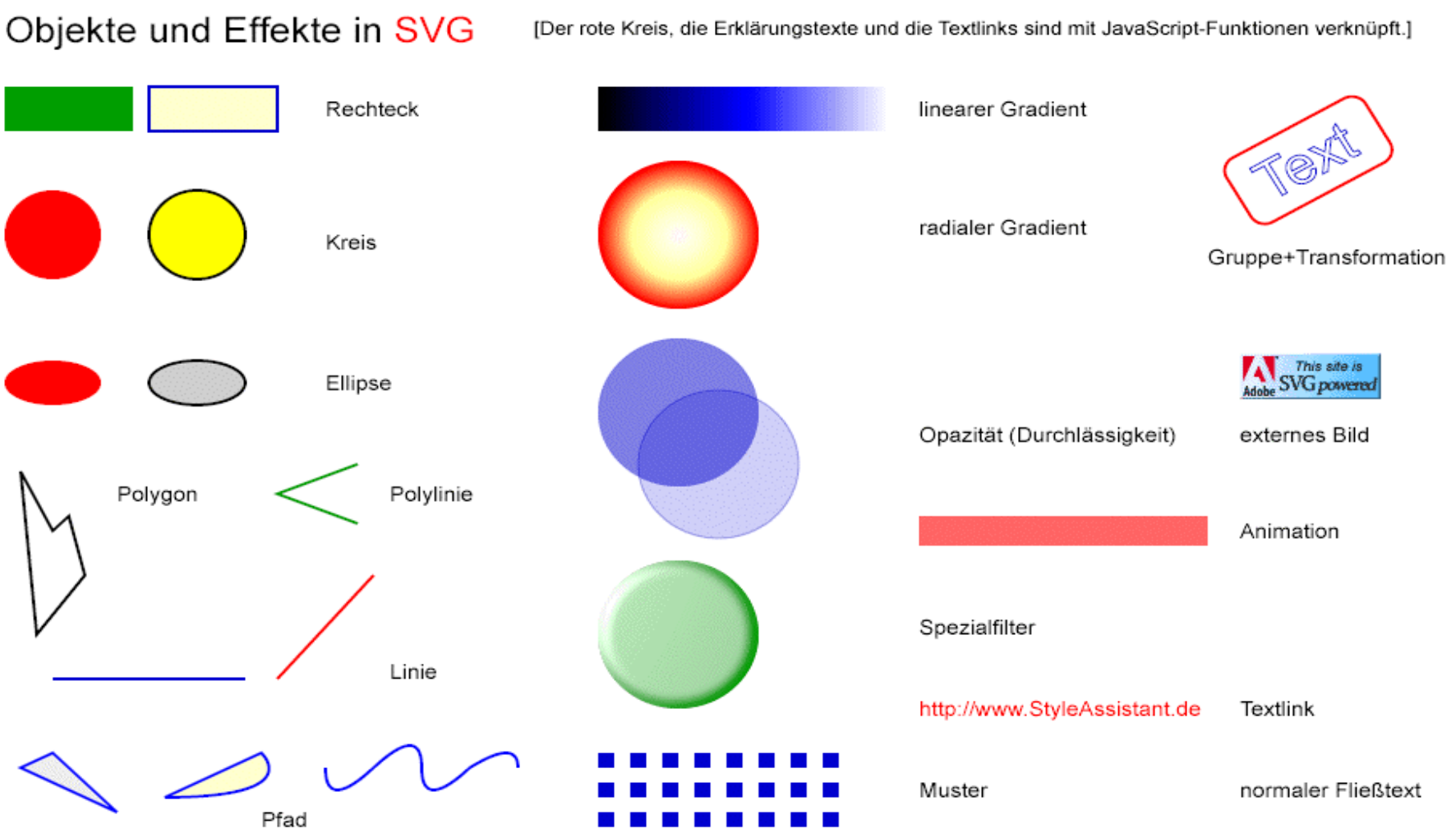
- SVG 1.0 ist eine im September 2001 verabschiedete W3C-Spezifikation zur Beschreibung von 2D-Vektorgrafiken in XML-Syntax [aktuell Version 1.1 (2003), 1.2 in Arbeit].

- Dieses Vektorformat ergänzt die gegenwärtig im Internet verwendeten Rasterformate (GIF, JPEG, PNG).

- SVG ermöglicht das Erstellen strukturierter Dokumente und Verarbeitung im Umfeld anderer XML-Technologien.

- SVG-Dokumente lassen sich statisch und dynamisch generieren und relativ einfach in Server-seitige Anwendungen integrieren.

SVG in 5 Minuten – Objekte und Effekte



SVG in 5 Minuten – Aufbau (1/8)

⇨ SVG-XML-Grundgerüst:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/
   DTD/svg10.dtd">

<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  <title>optionaler Titel</title>
  <desc>optionale Beschreibung</desc>
  <defs>Stylesheets, Skriptcode, Referenzen</defs>

  <!-- weitere SVG-Inhalte -->

</svg>
```

SVG in 5 Minuten – Aufbau (2/8)

- Grafische Grundformen wie Rechteck, Kreis, Ellipse, Linie, Polylinie, Polygon:

```
<rect x="..." y="..." width="..." height="..."/>

<circle cx="..." cy="..." r="..."/>

<ellipse cx="..." cy="..." rx="..." ry="..."/>

<line x1="..." y1="..." x2="..." y2="..."/>

<polyline points="x1,y1,...,xn,yn"/>

<polygon points="x1,y1,...,xn,yn"/>
```

- Textinhalte:

```
<text x="..." y="...">Textinhalt</text>
```

SVG in 5 Minuten – Aufbau (3/8)

⇨ Formatierung über Stylesheets (CSS) [oder Präsentationsattribute]

- roter Kreis mit blauem Rand der Stärke 2 Pixel:

```
<circle cx="100" cy="200" r="50" style="fill:
  #F00; stroke: #00C; stroke-width: 2px"/>
```

- grüner Text, 14 Pixel hoch in der Schriftart Arial (bzw. Alternativen):

```
<text x="20" y="50" style="fill: #090;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 14px">Text in SVG</text>
```

- interne bzw. externe Stylesheet-Definitionen (z. B. Klassen):

```
<defs><style type="text/css">
  <![CDATA[ /* CSS-Definitionen */ ]]>
</style></defs>

<?xml-stylesheet href="styles.css" type="text/css"?>
```

SVG in 5 Minuten – Aufbau (4/8)

⇨ Pfade (Kurvenzüge, Bögen)

- elliptischer Bogen:

```
<path d="M 200,200 L 200,100 A 100,100 0 0,1
 289,154 Z" style="fill: #FF0"/>
```

[M: moveto, L: lineto, A: elliptical arc, Z: closepath (hier abs. Koordinaten)]

⇨ Animationen

- Rechteckbreite erreicht innerhalb von 10 Sekunden 180 Pixel:

```
<rect x="600" y="360" width="0" height="20"
  style="fill: #F00;"><animate attributeType="XML"
  attributeName="width" begin="0s" dur="10s"
  fill="freeze" from="0" to="180"/></rect>
```

SVG in 5 Minuten – Aufbau (5/8)

- Gradienten (lineare und radiale Farbverläufe)

```
<defs>
  <linearGradient id="lingra1">
    <stop offset="0%" style="stop-color: #000"/>
    <stop offset="50%" style="stop-color: #00F"/>
    <stop offset="100%" style="stop-color: #FFF"/>
  </linearGradient>
</defs>

<rect x="400" y="70" width="180" height="30"
  style="fill: url(#lingra1)"/>
```

- Filter (u. a. bekannt aus Adobe Photoshop/Illustrator)
  - Filterelemente **feFilterName** mit Parametern als Attributwerte.

SVG in 5 Minuten – Aufbau (6/8)

- Textlinks (ähnlich zu Links in HTML)

```
<a xlink:href="http://svglbc.datenverdrahten.de/">
  <text x="100" y="200" style="fill: #F00">
    SVG - Learning By Coding
  </text>
</a>
```

- externe Bilder (ähnlich zu Bildern in HTML)

```
<image x="300" y="400" xlink:href="einbild.jpg"
  width="150" height="80"/>
```

Wichtig für Referenzen: XLink-Namespace deklarieren!

`xmlns:xlink="http://www.w3.org/1999/xlink"`

SVG in 5 Minuten – Aufbau (7/8)

- ⇨ Gruppierung (Zusammenfassung von Objekten zu Gruppen)
- ⇨ Transformationen (Rotieren, Verschieben, Neigen, Skalieren)

[rotate(w,x,y), translate(dx,dy), scale(fx,fy), skewX(w), skewY(w) bzw. matrix(a,b,c,d,e,f)]

- Gruppe (g) aus Rechteck und Text wird um -30° gedreht:

```
<g transform="rotate(-30,620,500)">
  <rect x="620" y="500"
    rx="10" ry="10" width="100" height="50"
    style="fill: none; stroke: #F00; stroke-width:
    2px"/>
  <text x="640" y="520" style="font-size: 36px;
    fill: none; stroke: #00C; stroke-width:
    1px">Textinhalt</text>
</g>
```

SVG in 5 Minuten – Aufbau (8/8)

⇨ SVG-Einbindung in HTML

- W3C-konform über die Element object oder iframe:

```
<object data="datei.svg" width="..." height="..."
  type="image/svg+xml">
  <!- Alternativinhalt -->
</object>

<iframe src="datei.svg" width="..." height="..."
  frameborder="0">
  <!- Alternativinhalt -->
</iframe>
```

- ggf. für Netscape 4.x mittels embed oder object/embed-Kombination:

```
<embed src="datei.svg" width="..." height="..."
  type="image/svg+xml">
  <!- Alternativinhalt -->
</embed>
```

Einbindung von Skripten

⇨ JavaScript oder ECMAScript (seltener VBScript):

Programmcode wird im Bereich von <defs>...</defs> abgelegt

- intern (als CDATA-Abschnitt, da reservierte Zeichen wie `<` `&` vorkommen können):

```
<script type="text/javascript">
<![CDATA[

   /* JS-Code */

]]>
</script>
```

- extern (als einfache Textdatei oder auch gzip-komprimiert):

```
<script xlink:href="datei.js" type="text/javascript"></script>
<script xlink:href="datei.js.gz" type="text/javascript"></script>
```

Schreiben eigener Funktionen

- Funktionen kapseln Codeblöcke für den späteren Aufruf:

```
var ...; // globale Variablen

function Funktionsname1(arg)
{
  var ...; // lokale Variablen
  /* weiterer Code der Funktion */
}

function Funktionsname2(arg1,arg2,...)
{
  var ...; // lokale Variablen
  /* weiterer Code der Funktion */
}
```

- Funktionen werden ereignisgesteuert aufgerufen
- Ablage erstellter Funktionen in Bibliotheken sinnvoll (*.js)

Reaktion auf Ereignisse (1/2)

- Die SVG verarbeitende Instanz (Browser, Plug-in) registriert bei der Interaktion ständig so genannte Ereignisse (Events):

  Beispiel: click-Event beim Anklicken von Objekten

- Zur Reaktion auf diese Events dienen Event-Handler:

  Beispiel: **on**click-Event-Handler (Prefix **on**)

- Event-Handler werden als Element-Attribute verwendet:

  Beispiel (Aufruf einer JS-Funktion):

```
<circle cx="50" cy="50" r="20" style="fill: #F00"
  onclick="EineFunktion(evt)"/>
```

Reaktion auf Ereignisse (2/2)

⇨ Event-Handler von SVG 1.x:

- Dokument:

  onabort, onerror, onresize, onscroll, onunload, onzoom

- Grafikobjekte:

  onactivate, onclick, onfocusin, onfocusout, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup

- Animationen:

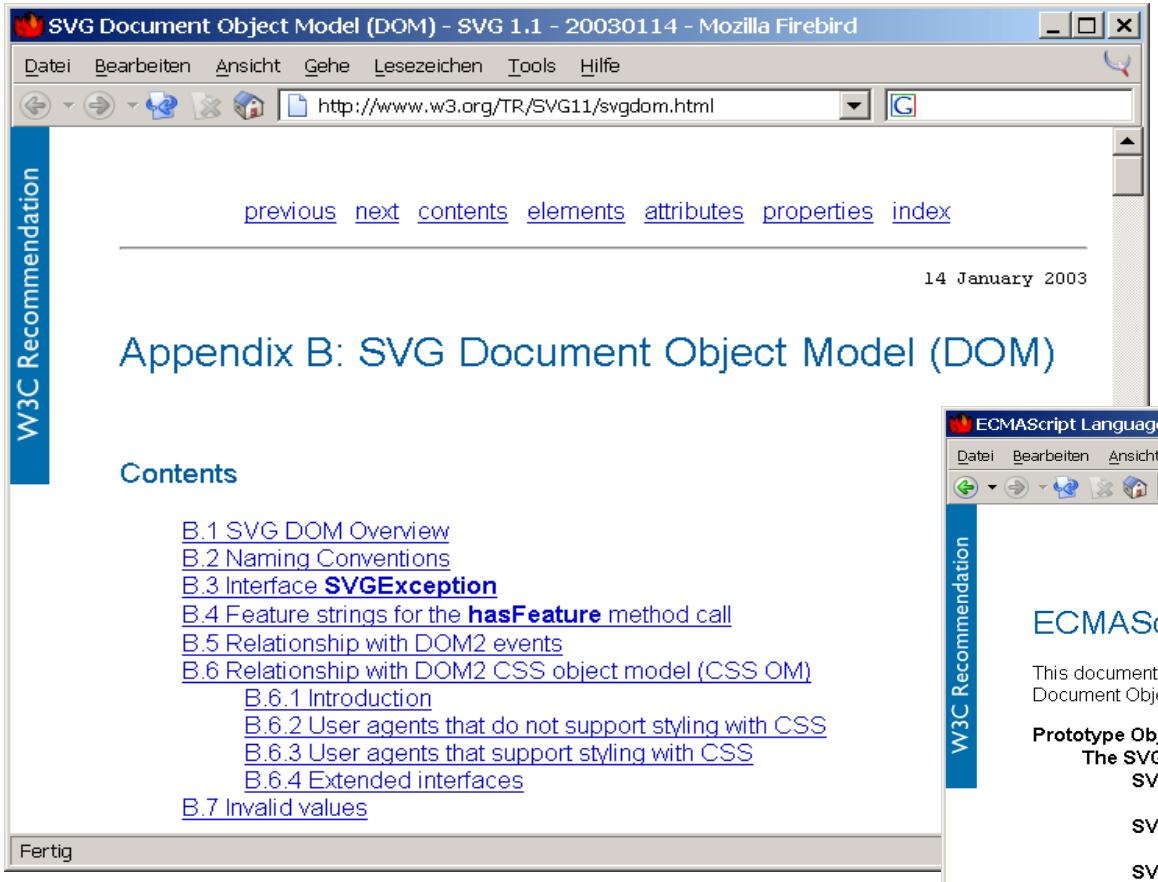
  onbegin, onend, onload, onrepeat

⇨ Im Adobe SVG Viewer zusätzlich verfügbar:

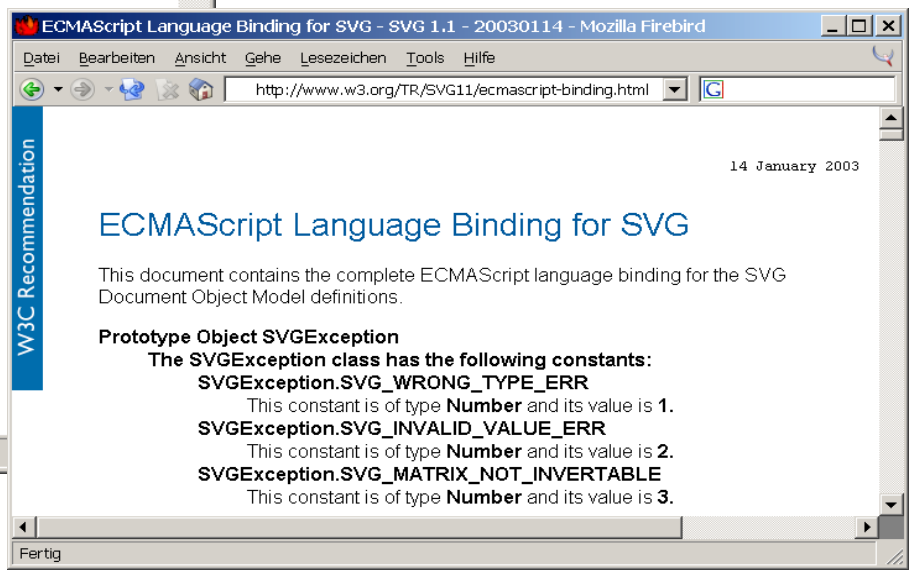
onkeydown, onkeypress, onkeyup

Wichtige SVG-DOM-Techniken (1/7)

⇨ SVG 1.x unterstützt das W3C-Document Object Model (DOM) Level 2 sowie zusätzliche Eigenschaften und Methoden.



DOM = plattform- und sprach-unabhängiges Interface (API) für den dynamischen Zugriff auf Inhalt, Struktur und Stil von Dokumenten.



Wichtige SVG-DOM-Techniken (2/7)

⇨ Verfügbare ...

DOM-Eigenschaften:	DOM-Methoden:
attributes	addEventListener(), appendChild()
childNode, childNodes	appendData(), cloneNode()
data, doctype	createAttribute(), createElement()
documentElement	createTextNode(), deleteData()
entities, firstChild	getAttribute(), getElementById()
lastChild, length	getElementsByTagName(), getPropertyValue()
name, nextSibling	hasAttributes(), hasChildNodes()
nodeName, nodeType	insertBefore(), insertData(), item()
nodeValue	removeAttribute(), removeChild()
ownerDocument	replaceData(), removeEventListener()
previousSibling	removeProperty(), replaceChild()
style, target	setAttribute(), setProperty()
value, ...	substringData(), ...

Wichtige SVG-DOM-Techniken (3/7)

⇨ Zusätzliche …

SVG-DOM-Techniken (Eigenschaften bzw. Methoden):

Animationen: getStartTime(), pauseAnimations(), unpauseAnimations()

Datenzugriff: getURL(), postURL(), parseXML(), printNode()

Datum/Zeit: getCurrentTime(), setCurrentTime()

Länge von Textinhalten: getComputedTextLength(), getSubStringLength()

Markierungen aufheben: deselectAll()

Objektrelationen: checkEnclosure(), checkIntersection()

Pfadinformationen: getTotalLength(), getPointAtLength()

Transformationen: getCTM(), flipX(), flipY(), inverse(), multiply()

Verschiebung: currentTranslate, previousTranslate

Zeichnungsparameter: viewBox, getBBox()

Zoomfaktor: currentScale, previousScale

...

Aufrufdetails sind in den Spezifikationen enthalten.

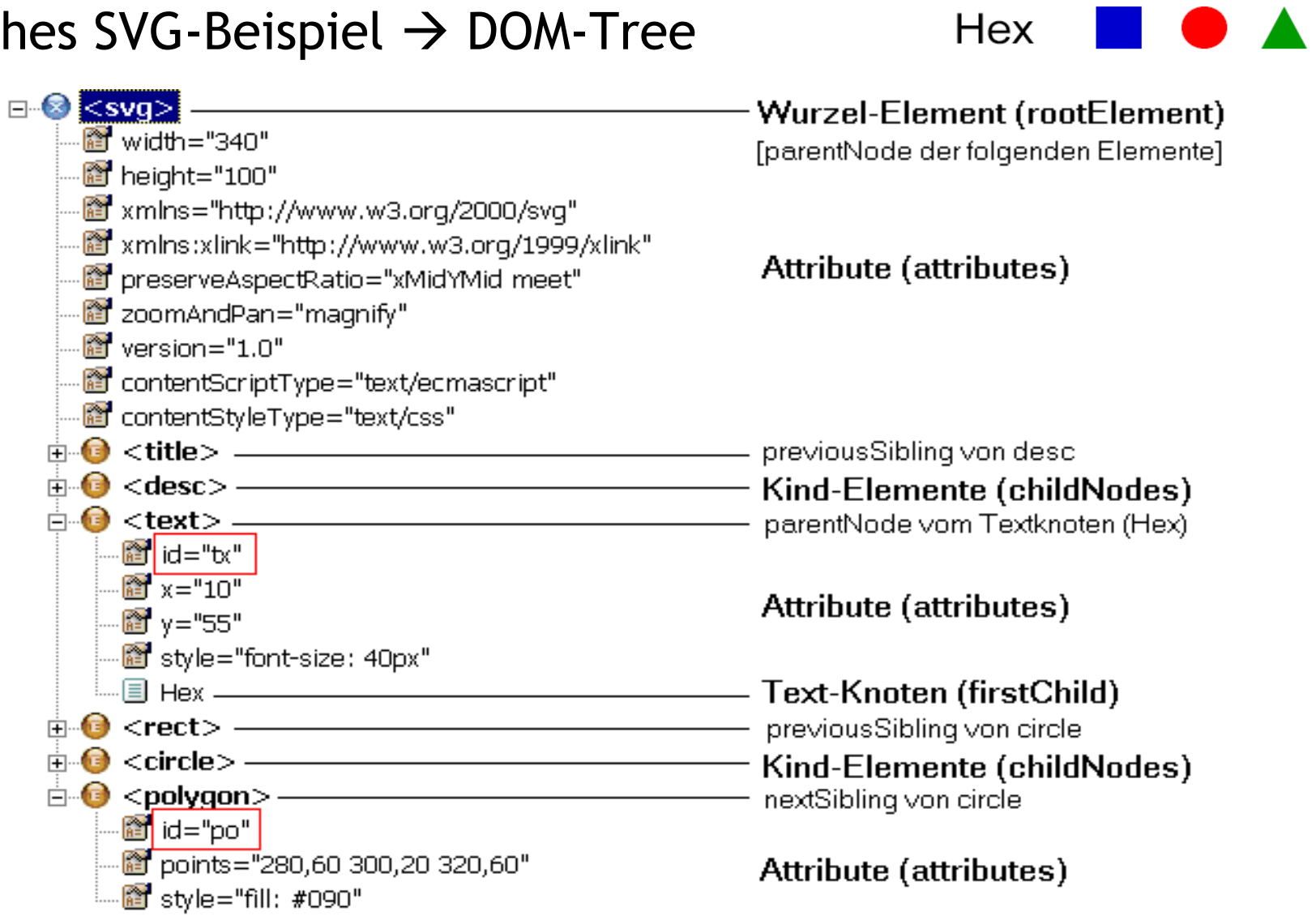
Wichtige SVG-DOM-Techniken (4/7)

⇨ Einfaches SVG-Beispiel → Code


```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="340" height="100" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <title>SVG-Beispiel</title>
  <desc>Es werden je ein Text, Rechteck, Kreis und Polygon definiert.</desc>

  <text id="tx" x="10" y="55" style="font-size: 40px">Hex</text>

  <rect id="re" x="135" y="20" width="40" height="40" style="fill: #00C"/>

  <circle id="kr" cx="230" cy="40" r="20" style="fill: #F00"/>

  <polygon id="po" points="280,60 300,20 320,60" style="fill: #090"/>

</svg>
```

Wichtige SVG-DOM-Techniken (5/7)

⇨ Einfaches SVG-Beispiel → DOM-Tree



Wichtige SVG-DOM-Techniken (6/7)

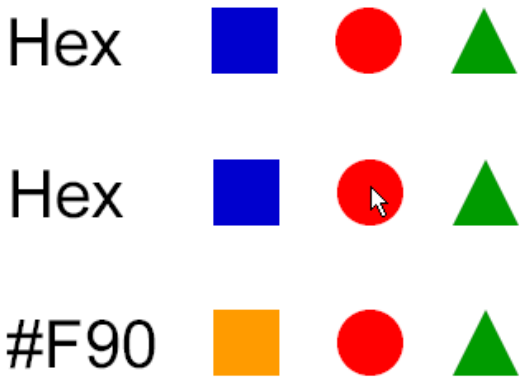
⇨ Beispiel erweitert mit SVG-DOM-Zugriff

- SVG-Objekte

```
<text id="tx" x="10" y="55" style="font-size: 40px">Hex</text>
<rect id="re" x="135" y="20" width="40" height="40" style="fill: #00C"/>
<circle id="kr" cx="230" cy="40" r="20" style="fill: #F00"
  onclick="ChangeColor(evt,'re','#F90')"/>
<polygon id="po" points="280,60 300,20 320,60" style="fill: #090"/>
```

- JavaScript-Funktion

```
function ChangeColor(click_evt,objid,col)
{
  var svgdoc,obj,txt;

  svgdoc=click_evt.target.ownerDocument;
  obj=svgdoc.getElementById(objid);
  obj.style.setProperty("fill",col);

  txt=svgdoc.getElementById("tx");
  txt.firstChild.nodeValue=col;
}
```



Wichtige SVG-DOM-Techniken (7/7)

⇨ Hinweise zu einigen Techniken (siehe Beispiele)

- Neue Elemente erzeugen

```
neues_element=svgDocument.createElement("elementname");
neues_element.setAttribute("attributname","attributwert");
object.appendChild(neues_element);
```

- Neue Textknoten erzeugen

```
neuer_textknoten=svgDocument.createTextNode("Textinhalt");
object.appendChild(neuer_textknoten);
```

- Stylesheet-Eigenschaften lesen/schreiben/entfernen

```
object.style.getPropertyValue("eigenschaft");
object.style.setProperty("eigenschaft","wert");
object.style.removeProperty("eigenschaft");
```

- Ereignisse global überwachen

```
object.addEventListener("event_type",Funktion,true|false);
object.removeEventListener("event_type",Funktion,true|false);
```

- Externe Ressourcen ansprechen [ASV, offiziell ab SVG 1.2]

```
getURL("http://www.example.com/abc.php?xyz=123",callback);
postURL("http://www.example.com/abc.php","daten",callback);
```

Demonstrationen

- ⇨ SVG-DOM-Scripting kompakt
- ⇨ Mini-Zeichenprogramm
- ⇨ Periodensystem mit Online-Datenabfrage
- ⇨ SVG-Zugriff über ein HTML-Dokument
- ⇨ SVG — Learning By Coding (Beispielsammlung)

Zusammenfassung und Ausblick

- DOM-basierte Aktionsprogrammierung erweitert die Möglichkeiten von SVG um Dynamik und Interaktion.
- Standardisierte und offene Technologien erlauben prinzipiell plattformübergreifende Entwicklung.
- Erfahrungen aus dem DHTML-Bereich lassen sich relativ einfach auf den SVG-Kontext übertragen (steile Lernkurve).
- Die aktuellen SVG-Viewer bieten (leider) noch keinen einheitlichen Funktionsumfang. Außerhalb von Intranets kann es somit zu Problemen bei der Umsetzung von Lösungen kommen.
- Vor dem Schreiben eigener Routinen sollten die vorhandenen Basistechniken ausgereizt sein. Vor allem zur Erzielung von Effekten reichen die Animationselemente oft aus.

Nützliche Online-Ressourcen

- http://www.w3.org/Graphics/SVG/
- http://www.w3.org/TR/DOM-Level-2-Core/
- http://www.w3.org/TR/SVG11/svgdom.html
- http://www.w3.org/TR/SVG11/ecmascript-binding.html
- http://www.w3.org/TR/DOM-Level-2-Core/ecma-script-binding.html
- http://www.ecma-international.org/publications/standards/Ecma-262.htm
- http://www.adobe.com/svg/
- http://xml.apache.org/batik/
- http://www.svgfoundation.org/
- http://www.scale-a-vector.de/
- http://svg.tutorial.aptico.de/
- http://phrogz.net/ObjJob/
- http://svglbc.datenverdrahten.de/ → [Beispielmaterial zum Vortrag]
- http://www.corel.com/svg/
- http://mozilla.org/projects/svg/
- http://www.svgx.org/
- http://www.svg-cafe.com/
- http://www.carto.net/papers/svg/
- http://www.protocol7.com/svg-wiki/