Hochschule Merseburg (FH)
FB IKS / Studiengang TRW
Dr. T. Meinike
2005…2010

Wichtige DOM-Eigenschaften und -Methoden für die SVG-Aktionsprogrammierung

Hinweis: Die Eigenschaften und Methoden sind als weitgehend selbsterklärender Pseudocode formuliert.
Quellen: *http://www.w3.org/DOM/* und *http://www.w3.org/TR/SVG11/*

W3C-DOM

– Zugriff auf Element- und Dokumentknoten:

```
elementobjekt = document.getElementById("ID")
elementknotenliste = knoten.getElementsByTagName("elementname")

wurzelknoten = document.documentElement bzw. document.rootElement
dokumentknoten = elementknoten.ownerDocument
```

– Zugriff auf Elemente einer Knotenliste:

```
knotenliste.item(index)
```

– Element- und Textknoten erzeugen:

```
elementknoten = document.createElement("elementname")
textknoten = document.createTextNode("Text")
```

– Abfragen von Knoten auf Existenz:

```
elementknoten.hasAttribute("attributname") --> true|false
elementknoten.hasAttributes() --> true|false
knoten.hasChildNodes() --> true|false
```

– Knoten-Informationen:

```
elementname = elementknoten.tagName
knotenname = knoten.nodeName
knotenwert = knoten.nodeValue
knotentyp = knoten.nodeType --> [1...12]

  ELEMENT_NODE                =  1
  ATTRIBUTE_NODE              =  2
  TEXT_NODE                   =  3
  CDATA_SECTION_NODE          =  4
  ENTITY_REFERENCE_NODE       =  5
  ENTITY_NODE                 =  6
  PROCESSING_INSTRUCTION_NODE =  7
  COMMENT_NODE                =  8
  DOCUMENT_NODE               =  9
  DOCUMENT_TYPE_NODE          = 10
  DOCUMENT_FRAGMENT_NODE      = 11
  NOTATION_NODE               = 12
```

– DOM-Baum-Navigation:

```
knotenliste = knoten.childNodes
ersterkindknoten = knoten.firstChild
letzterkindknoten = knoten.lastChild
naechstergeschwisterknoten = knoten.nextSibling
vorherigergeschwisterknoten = knoten.previousSibling
elternknoten = knoten.parentNode
```

– Knoten-Manipulation:

```
knoten.appendChild(kindknoten)
knotenkopie = knoten.cloneNode(true|false)
knoten.insertBefore(kindknoten,knotennachfolger)
knoten.removeChild(kindknoten)
knoten.replaceChild(ersatzknoten,ersetzterknoten)
```

– Zugriff auf Textdaten lesend/schreibend:

```
textknoten.appendData("weiterer Text")
textknoten.deleteData(startzeichen_ab_0,zeichenanzahl)
textknoten.insertData(startzeichen_ab_0)
textknoten.replaceData(startzeichen_ab_0,zeichenanzahl,"ersatztext")
textinhalt = textknoten.data bzw. textknoten.nodeValue
textinhalt = textknoten.substringData(startzeichen_ab_0,zeichenanzahl)
```

– Zugriff auf Attribute lesend/schreibend:

```
elementknoten.attributes[index]
elementknoten.attributes["attributname"]

attributknoten = document.createAttribute("attributname")
attributknoten.value = "attributwert"
elementknoten.setAttributeNode(attributknoten)

elementknoten.getAttribute("attributname")
elementknoten.setAttribute("attributname","attributwert")

elementknoten.getAttributeNode("attributname")

elementknoten.removeAttribute("attributname")
elementknoten.removeAttributeNode(elementknoten.attributes["attributname"])
elementknoten.removeAttributeNode(elementknoten.attributes[index])
```

– Zugriff auf Stylesheet-Eigenschaften lesend/schreibend:

```
objekt.style.getPropertyValue("eigenschaft")
objekt.style.setProperty("eigenschaft","wert","priorität")
objekt.style.removeProperty("eigenschaft")
```

– Ereignisse global überwachen:

```
objekt.addEventListener("ereignis",Funktion,true|false)
objekt.removeEventListener("ereignis ",Funktion,true|false)
```

– Spezielle Techniken:

```
document.implementation.hasFeature(feature,version) --> true|false
(z. B. feature --> "XML" und version --> "1.0")

textknoten.splitText(zeichen_ab_0) --> textknoten.nextSibling

neuesdokument = document.implementation.createDocument("","",null)

dokumentfragment = document.createDocumentFragment()
dokumentfragment.[neue knoten erzeugen...]
document.appendChild(dokumentfragment)

kommentar = document.createComment("Text")
cdataabschnitt = document.createCDATASection("inhalt")
entityknoten = document.createEntityReference("entityname")
verarbeitungsanweisung = document.createProcessingInstruction("name","inhalt")
```

Hinweis:
Bei Bedarf lassen sich einige der vorgenannten Methoden analog auch mit einem definierten Namensraum aufrufen (im SVG-Kontext z. B. der XLink-Namensraum):

```
createElementNS("namespace", ...)      createAttributeNS("namespace", ...)
getAttributeNS("namespace", ...)       setAttributeNS("namespace", ...)

getAttributeNodeNS("namespace", ...)   removeAttributeNS("namespace", ...)
setAttributeNodeNS("namespace", ...)   hasAttributeNS("namespace", ...)

getElementsByTagNameNS("namespace", ...)
```

SVG-DOM

– Zeit abfragen/setzen:

```
laufzeit = svgrootelement.getCurrentTime()
svgrootelement.setCurrentTime(zeitwert_float)
```

– Animationen steuern:

```
svgrootelement.pauseAnimations()
svgrootelement.unpauseAnimations()
svgrootelement.animationsPaused() --> true|false

startzeit = animationsobjekt.getStartTime()

animationsobjekt.beginElement()
animationsobjekt.endElement()
animationsobjekt.beginElementAt(zeitwert_float)
animationsobjekt.endElementAt(zeitwert_float)

Basis- und animierte Attributwerte:
objekt.attribut.baseVal.value bzw. objekt.attribut.animVal.value
```

– Pfadinformationen:

```
pfadlaenge = pfadobjekt.getTotalLength()
punktobjekt = pfadobjekt.getPointAtLength(laenge)
xwert = punktobjekt.x
ywert = punktobjekt.y
```

– Skalierung und Verschiebung der Grafik:

```
neueskalierung = svgrootelement.currentScale
alteskalierung = svgrootelement.previousScale

neueverschiebung = svgrootelement.currentTranslate
alteverschiebung = svgrootelement.previousTranslate

xverschiebung = [neue|alte]verschiebung.x
yverschiebung = [neue|alte]verschiebung.y
```

– Text-Informationen:

```
zeichenzahl = textobjekt.getNumberOfChars()
textlaenge_in_px = textobjekt.getComputedTextLength()
textteillaenge_in_px = textobjekt.getSubStringLength(zeichen_ab_0,zeichenanzahl)
zeichen_x_start_position = textobjekt.getStartPositionOfChar(zeichen_ab_0).x
zeichen_y_start_position = textobjekt.getStartPositionOfChar(zeichen_ab_0).y
zeichen_x_end_position = textobjekt.getEndPositionOfChar(zeichen_ab_0).x
zeichen_y_end_position = textobjekt.getEndPositionOfChar(zeichen_ab_0).y
zeichen_breite = textobjekt.getExtentOfChar(zeichen_ab_0).width
zeichen_hoehe = textobjekt.getExtentOfChar(zeichen_ab_0).height
zeichen_rotation = textobjekt.getRotationOfChar(zeichen_ab_0)

punkt = svgrootelement.createSVGPoint()
punkt.x = xwert
punkt.y = ywert
zeichenindex = textobj.getCharNumAtPosition(punkt)

Textauswahl: textobjekt.selectSubString(zeichen_ab_0,zeichenanzahl)
Auswahl aufheben: svgrootelement.deselectAll()
```

– Zeichnungsparameter:

```
viewBox als Attribut des äußeren SVG-Elements bzw. Parameter beliebiger Objekte
abfragen [svgelement.getAttribute("viewBox")]

Umgebungs-Rechteck eines SVG-Objektes:
rahmenbox = svgelementobjekt.getBBox()
x = rahmenbox.x
y = rahmenbox.y
breite = rahmenbox.width
hoehe = rahmenbox.height
```

– Beeinflussung von Transformationen mit Matrixmethoden:

```
matrix = svgrootelement.createSVGMatrix()
Matrixparameter a bis f (lesen/schreiben):
neuematrix.a bis neuematrix.f

matrix.translate(x,y)
matrix.rotate(winkel)
matrix.rotateFromVector(x,y)
matrix.scale(faktor)
matrix.scaleNonUniform(xfaktor,yfaktor)
matrix.skewX(winkel)
matrix.skewY(winkel)

produktmatrix = matrix1.multiply(matrix2)   inversematrix = matrix.inverse()
xspiegelmatrix = matrix.flipX()             yspiegelmatrix = matrix.flipY()
svgobjektmatrix = svgobjekt.getCTM()
```

– Eigenschaften von (Maus-)Ereignissen (evt = event-Objekt):

```
Ursprung eines Ereignisses: evt.target    Ereignis-Typ: evt.type
Bildschirm-Koordinaten:     evt.screenX   evt.screenY
Fenster-Koordinaten:        evt.clientX   evt.clientY
Ctrl-, Shift-, Alt-Taste:   evt.ctrlKey   evt.shiftKey   evt.altKey
```

– Objektrelationen bezogen auf einen rechteckigen Rahmen:

```
Einschluss von Objekten:
svgrootelement.checkEnclosure(grafikobjekt,rahmenrechteck)

Überlagerung von Objekten:
svgrootelement.checkIntersection(grafikobjekt,rahmenrechteck)

rahmenrechteck = svgrootelement.createSVGRect()
rahmenrechteck.x = xwert
rahmenrechteck.y = ywert
rahmenrechteck.width = breite
rahmenrechteck.height = hoehe
```

– Datenzugriff (ASV 3/6):

```
getURL(url,callback)
url = externe Datenquelle (muss sich auf demselben Server befinden)
callback = Client-seitiges Skript zu Auswertung der Daten


postURL(url,text,callback,type,enc)
url = externe Datenquelle (muss sich auf demselben Server befinden)
text = zu postende Daten
callback = clientseitiges Skript zu Auswertung der Daten
type = MIME-Type (optional)
enc = encoding-Parameter: gzip oder deflate (optional)


function callback(urlRequestStatus)
{
  /*
    Rückgabewerte des Objektes urlRequestStatus:
    - urlRequestStatus.success (true|false)
    - urlRequestStatus.contentType (MIME-Typ)
    - urlRequestStatus.content (Objekt)

    weitere Verarbeitung von urlRequestStatus.content,
    z. B. mittels split("Trennzeichen")
  */
}


xmldokumentfragment = parseXML(urlRequestStatus.content,context)
nach dem Laden eines XML-Dokuments mittels
getURL("datei.xml",callback)


rueckgabe = printNode(knoten)
z. B.: printNode(document.getElementById("ID"))
```

Hinweis: Browser mit nativer SVG-Unterstützung ermöglichen AJAX-Nutzung auf der Basis des XMLHttpRequest-Objektes.