Eksempel på transformation: XML -> RTF dokument:

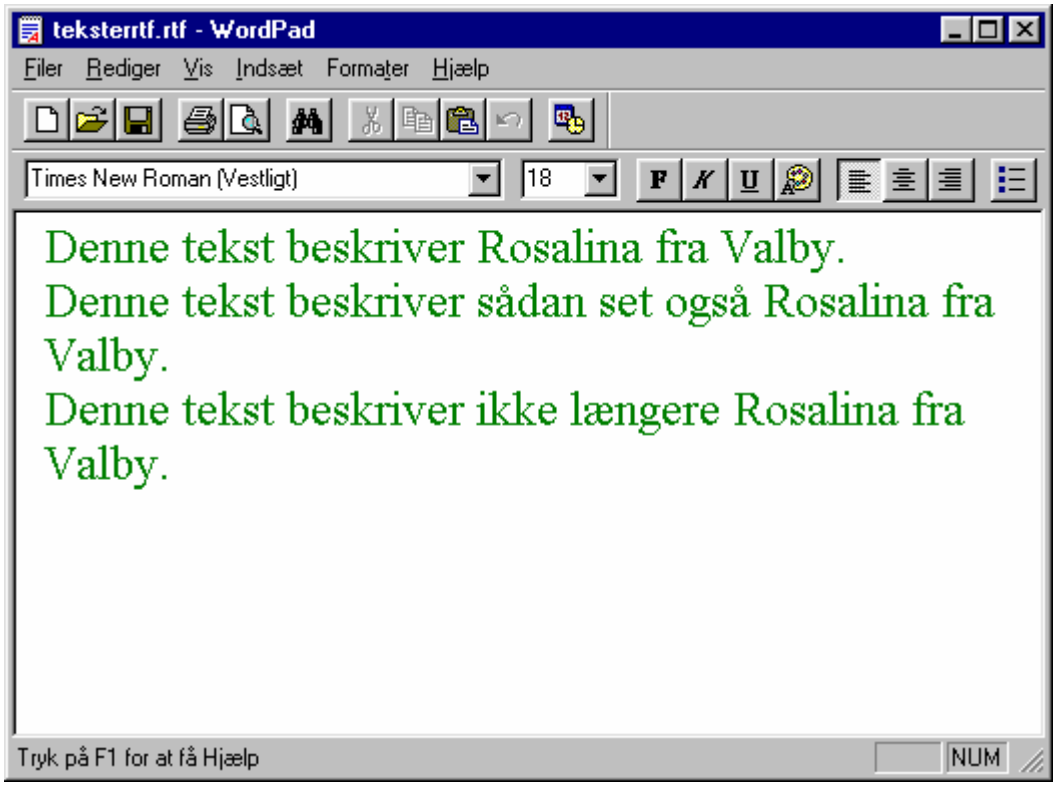
RTF dokumenter er **Rich Text** dokumenter der kan formateres med farver og forskellige skrifttyper. I nogle tilfælde kan det være interessant at transformere et XML dokument til et RTF dokument. Dette sker igen grundlæggende ved at vi sørger for at indsætte **literale** tekst data som normalt bruges i et RTF dokument! Ellers kan vi blot hente informationerne fra et XML dokument!

Vi kan starte med dette simple XML dokument:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<?xml-stylesheet type="text/xsl" href="til_rtf.xsl"?>

<tekster>
   <tekst>
   Denne tekst beskriver Rosalina fra Valby.
   </tekst>
   <tekst>
   Denne tekst beskriver sådan set også Rosalina fra Valby.
   </tekst>
   <tekst>
   Denne tekst beskriver ikke længere Rosalina fra Valby.
   </tekst>
</tekster>
```

Slut resultatet skulle gerne ser nogenlunde sådan ud:



Vi kan **transformere** XML dokumentet til et **RTF** dokument således:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output
method="text"
encoding="iso-8859-1"
omit-xml-declaration="yes"
indent="yes"
/>

<xsl:template match="tekster">

{\rtf1\ansi\ansicpg1252\deff0\deflang1030{\fonttbl{\f0\fnil\fcharset0 Times New Roman;}}
{\colortbl ;\red0\green128\blue0;}
<xsl:apply-templates select="tekst" />
}
</xsl:template>

<xsl:template match="tekst">

\viewkind4\uc1\pard\cf1\f0\fs36 <xsl:value-of select="." />\cf0\fs20\par
</xsl:template>


</xsl:stylesheet>
```

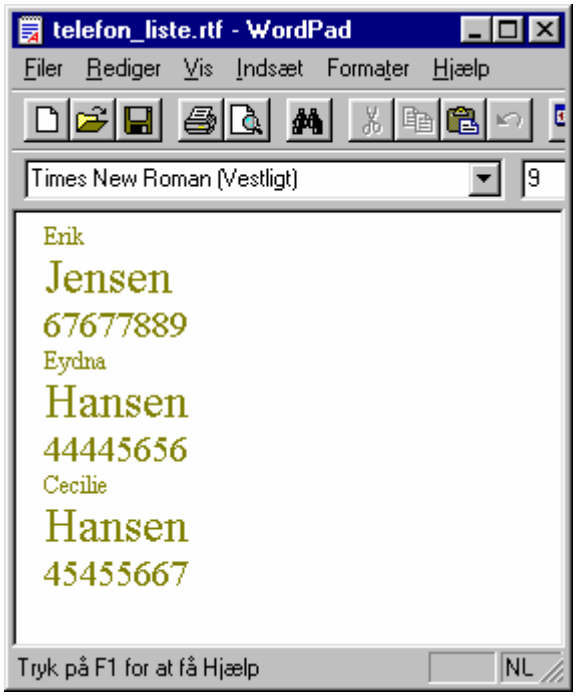
De mange koder er – stort set - alle nødvendige for at formatere RTF dokumentet! Koderne definerer **farver, skriftstørrelsen** og skrift **typen**.

Her er defineret Times New Roman, farven grøn og en skrift størrelse på 18 punkt (nemlig 36 delt med to!).

RTF filen skal altså dels have et **hovede** som kun skal stå en gang og dels en række **paragraffer** eller afsnit som indledes med linjeskift (som i en tekst behandling)!

Det er vigtigt for at denne proces skal lykkes at man transformerer til en tekst fil som f. eks. kaldes fil.**rtf**. Derefter skal denne RTF fil åbnes f. eks. i **Wordpad** og **gemmes** som **tekst** type og **samtidigt** med **filtypen** .rtf altså som **dokument**.**rtf** f. eksempel!

Det er en lidt kompliceret proces, men det kan lade sig gøre og når først stylesheet'et **er** skrevet een gang er det **let** at bygge videre!



Eksempel på transformation: XML -> SVG:

SVG er et XML sprog som kan bruges til at definere grafiske elementer. SVG har sin egen DOCTYPE (dtd) som definerer hvad der er et legalt SVG dokument. Vi kan transformere et XML dokument til et SVG dokument f. eks. for at vise data grafisk!

Et simpelt eksempel på et XML dokument kunne være:

```
<?xml version="1.0"?>
<?xml-stylesheet href="resultater_svg.xsl" type="text/xsl" ?>
<data>
<rubrik>Forskellige overskud i firmaet.</rubrik>
<liste>
	<resultat>230</resultat>
	<resultat>130</resultat>
	<resultat>120</resultat>
	<resultat>210</resultat>
	<resultat>530</resultat>
	<resultat>430</resultat>
	<resultat>630</resultat>
	<resultat>130</resultat>
	<resultat>370</resultat>
</liste>
</data>
```

Det vi så ønsker at få vist disse overskud i et **søjlediagram**. Vi kan så skrive dette stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:output
method="xml"
encoding="iso-8859-1"
omit-xml-declaration="no"
indent="yes"
doctype-public="-//W3C//DTD SVG 1.0//EN"
doctype-system="http://localhost/svg10.dtd"
/>

<xsl:template match="/">

<svg xmlns="http://www.w3.org/2000/svg" width="400" height="300">

<xsl:for-each select="//resultat">
<rect x="{(position()*50)-50}" y="50" width="48" height="{. div 2}" >
<xsl:attribute name="style">
<xsl:choose>
<xsl:when test="position() mod 2 = 0">
fill:#669966
</xsl:when>
<xsl:otherwise>
fill:#996666
</xsl:otherwise>
</xsl:choose>

</xsl:attribute>
</rect>
</xsl:for-each>
</svg>
</xsl:template>

<xsl:template match="person">
</xsl:template>


</xsl:stylesheet>
```

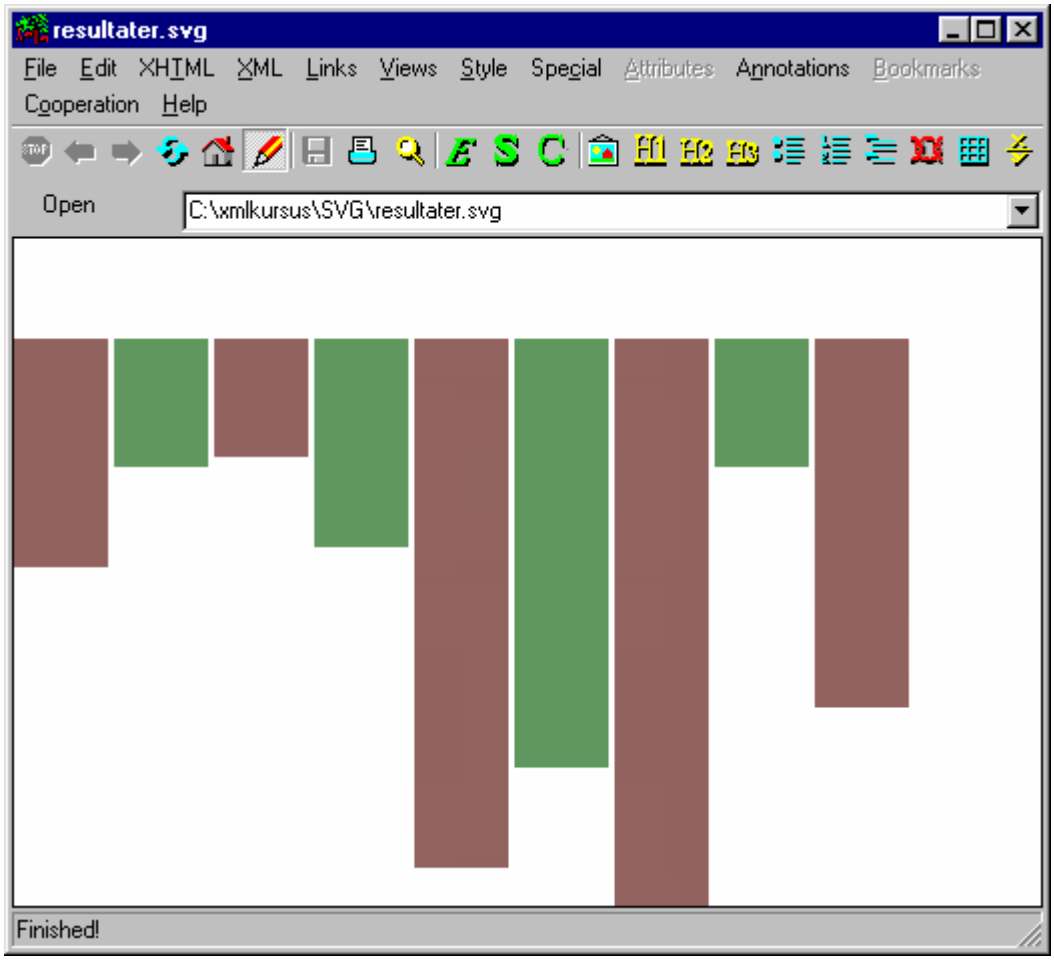
Vi kan som vist indsætte en DOCTYPE i xsl:output – på den måde kan vi også validere at det producerede dokument er gyldigt! Her går vi ud fra at DTD'en gemt lokalt!

```
doctype-public="-//W3C//DTD SVG 1.0//EN"
doctype-system="http://localhost/svg10.dtd"
```

SVG dokumenter skal have en bestemt rod og et bestemt namespace for at være gyldige. dem indsætter vi som direkte tekst i output træet:

```
<svg xmlns="http://www.w3.org/2000/svg" width="400" height="300">
```

Vi viser blot her søjlerne på den **letteste** måde – dette stylesheet kunne let **forbedres** ved at gøre højde og bredde af søjlerne variabel!:



Filen resultater.svg kan vises i f. eks. browseren Amaya fra http://www.w3.org og selve filen kan f. eks. produceres med msxsl:

msxsl resultater.xml –pi –o resultater.svg

Eksempel på transformation: SVG -> VML:

Vi har andetsteds omtalt **SVG** og **VML** som er to **XML** sprog der bruges til at definere vektor grafik. VML er et Markup Language som kun bruges af **Microsoft** produkter – bl. a. Office pakken. Grundlæggende er VML en simplere – og tidligere – udgave af SVG.

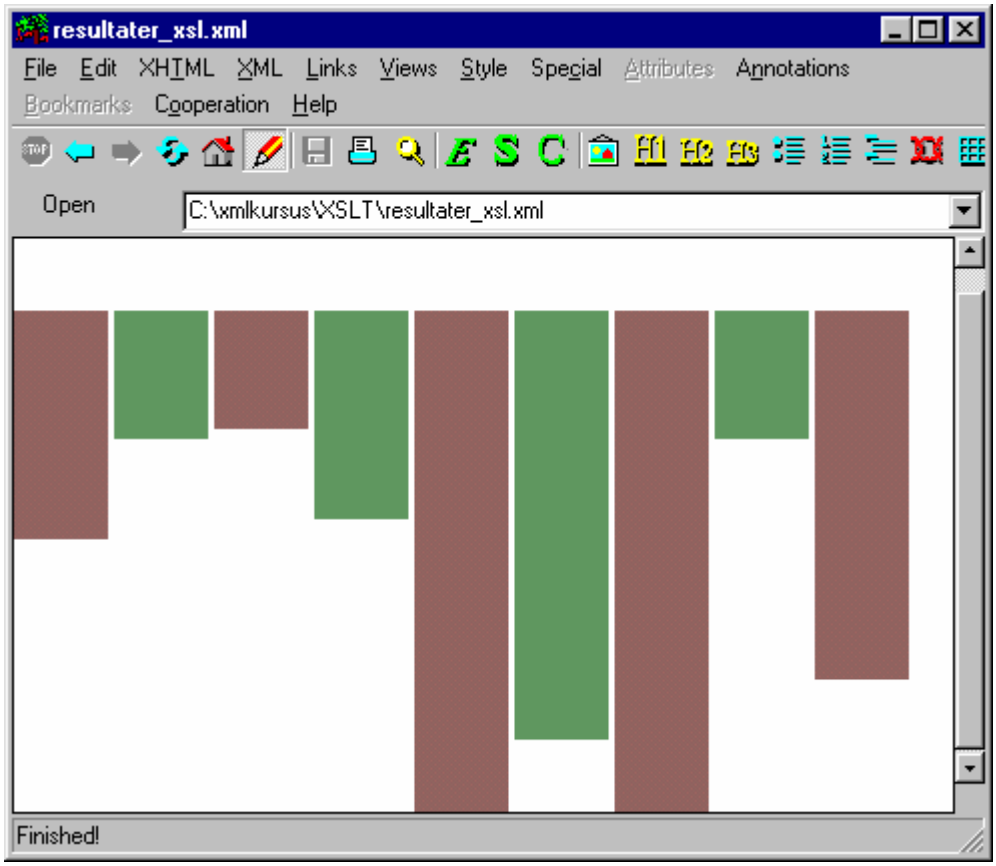
Vi vil i dette eksempel se på hvordan man kan **transformere** et SVG dokument til et VML format.

Vores **SVG** dokument ser således ud:

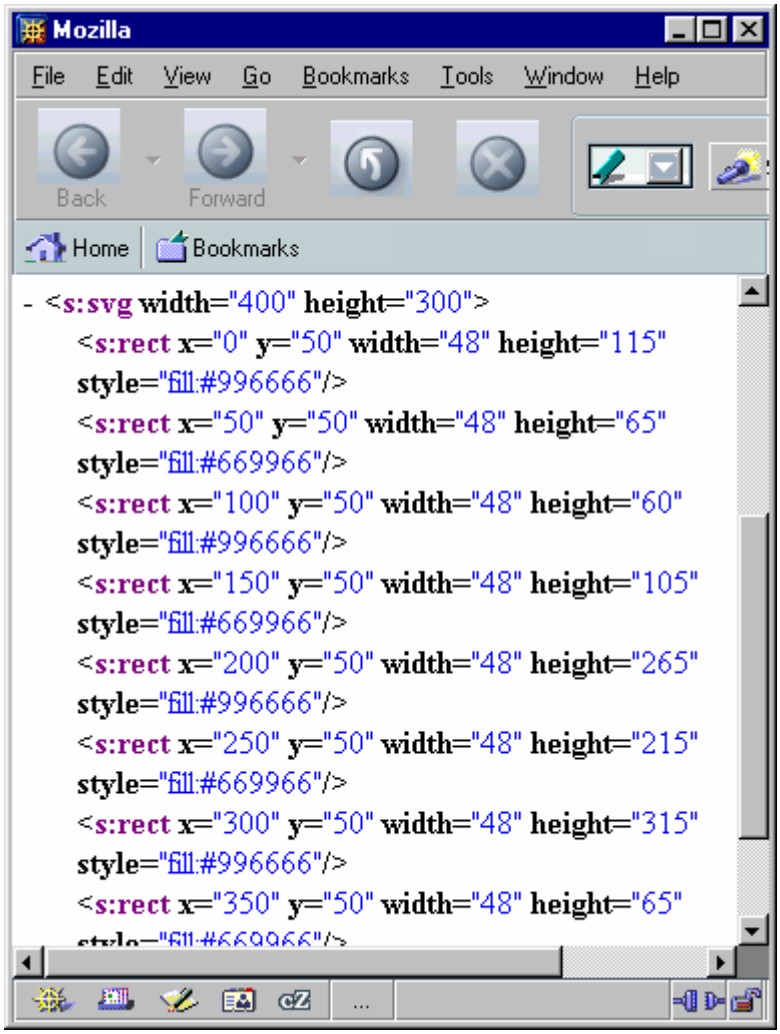
```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://localhost/svg10.dtd">
<s:svg width="400" height="300" xmlns:s="http://www.w3.org/2000/svg">
<s:rect x="0" y="50" width="48" height="115" style="fill:#996666" />
```

```
<s:rect x="50" y="50" width="48" height="65" style="fill:#669966" />
<s:rect x="100" y="50" width="48" height="60" style="fill:#996666" />
<s:rect x="150" y="50" width="48" height="105" style="fill:#669966" />
<s:rect x="200" y="50" width="48" height="265" style="fill:#996666" />
<s:rect x="250" y="50" width="48" height="215" style="fill:#669966" />
<s:rect x="300" y="50" width="48" height="315" style="fill:#996666" />
<s:rect x="350" y="50" width="48" height="65" style="fill:#669966" />
<s:rect x="400" y="50" width="48" height="185" style="fill:#996666" />
</s:svg>
```

Vi kunne have skrevet dokumentet lidt lettere – med tanke på at det skal transformeres til VML! – men vi har valgt denne form! Dokumentet er gemt som resultater.xml. Det kan vises i **Amaya** således:



Hvis vi ønsker det vist f. eks. i browseren Mozilla eller i Internet Explorer fås:



Vi kan nu transformere SVG dokumentet til **VML** med dette stylesheet:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:s="http://www.w3.org/2000/svg"
xmlns:vml="urn:schemas-microsoft-com:vml"
>
<xsl:output method="xml" omit-xml-declaration="yes" indent="yes" />

<xsl:template match="/">

<html xmlns:vml="urn:schemas-microsoft-com:vml" >
   <head>
        <style>
        vml\:*{behavior:url(#DEFAULT#VML);}
        </style>
        </head>
   <body>
        <xsl:apply-templates />
   </body>
</html>
</xsl:template>
```

```
<xsl:template match="//s:rect">
<vml:rect fillcolor="{substring-after(@style,'fill:')}" style="width:{@width};height:{@height};top:{@y};left:{@x}"/>
</xsl:template>

</xsl:stylesheet>
```

Princippet er altså meget simpelt: **Hver** gang XSL **processoren** finder et element s:**rect** skal den oprette et nyt element ved navn **vml**:rect!

Det mest besværlige her i virkeligheden at vi skal have fat i **farven** som er en del af en style i SVG dokumentet. Derfor bruger vi XSLT (og XPATH) funktionen **substring**-after for at hente det som kommer efter 'fill:' inden i strengen!

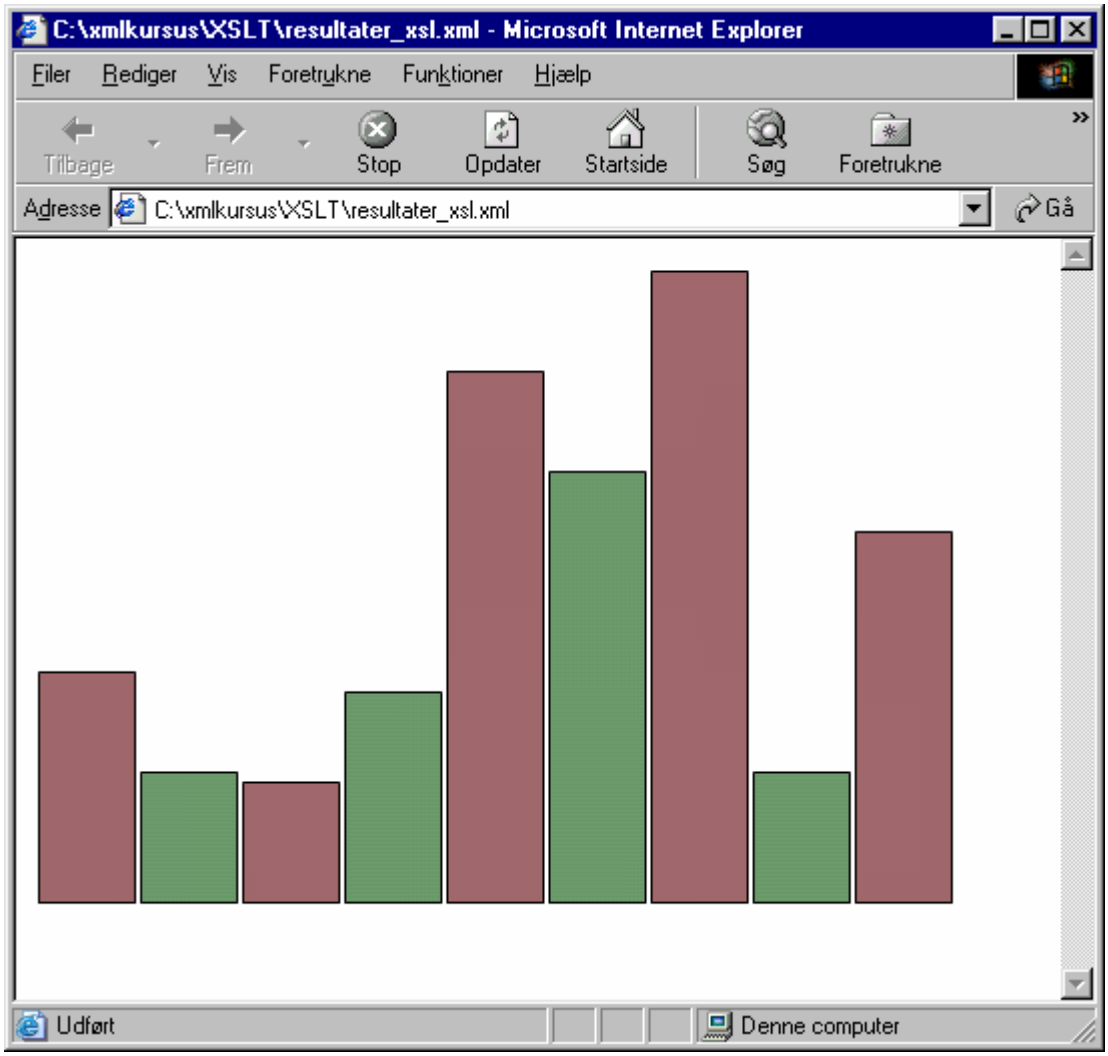
Vi kan se at transformationen i meget høj grad består i at vi indsætter literal tekst i output træet. Dette gælder også <head> som skal indeholde disse VML erklæringer. Vi importerer de to namespaces for VML og SVG og kan på den måde bruge dem i stylesheetet! Vi oversætter derefter værdierne fra SVG til VML ved hjælp af såkaldte attribut value templater – jvf brugen af { og }.

Dette stylesheet kunne gøres noget mere fleksibelt hvis vi inddrog flere elementer – men her har vi blot illustreret processen med et element nemlig rect!

Hvis vi sætter en stylesheet erklæring på XML filen således:

```
<?xml version="1.0"?>
<?xml-stylesheet href="svg_vml.xsl" type="text/xsl"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://localhost/svg10.dtd">
<s:svg width="400" height="300" xmlns:s="http://www.w3.org/2000/svg">
.....
```

kan vi få grafen vist i Internet Explorer:



Vi kan stadig vise XML dokumentet (SVG dokumentet) i Amaya – selv om det nu har fået et stylesheet. Men hvis vi transformerer XML dokumentet til et nyt HTML dokument – kan Amaya ikke vise denne fil!

Mozilla kan heller ikke vise XML (SVG) dokumentet efter at det har fået et stylesheet. Mozilla forstår ikke koderne til VML!