# Første seminar

Opgave 1:

Kig på koden bag websites som [dr.dk](http://www.dr.dk/) og [folketinget.dk](http://www.ft.dk/)

* Hvilke tags kan du genkende?

*<html>,  
<head>: Container for head elementer  
<title>: Page title,*<body>: Body of the document  
*<h1>, <h2>, <h3>: header levels  
<ul>: Unordered list  
<li>: Item in a list  
<p>: new paragraph*

* Valider siderne med [W3C validatoren](http://validator.w3.org/). Hvilke typer fejl ser du?

Opgave 2:

Gem [denne side](http://www.brics.dk/ixwt/exercises/physical.html) og lav den om til at bruge CSS og logisk markup

*se output på google code*

* Validerer din løsning med W3C validatoren?
  + *Ikke umiddelbart*

Opgave 3:

Afprøv [HTML Tidy værktøjet](http://www.w3.org/People/Raggett/tidy/) på nogle af de websites, der ikke var gyldige i opgave 1. Kan værktøjet reparere fejlene?

Bonus:

Kig på den foreløbige udgave af [HTML 5 specifikationen](http://dev.w3.org/html5/spec/Overview.html).

* Hvad er de væsentligste nyheder i forhold til HTML 4?
* Hvad gør det nye section element? Hvad er forskellen fra div?
* Hvilke nye værdier kan type attributten for input have? Hvorfor er de nyttige?
* Kan de nye tilføjelser betegnes som fysiske eller logiske?
* Hvilke elementer er helt fjernet i forhold til HTML 4?
* Er disse elementer fysiske eller logiske?

Opgave 4:

Forklar med egne ord kort forskellene mellem SGML og XML, og mellem HTML og XHTML

*XML delmængde af SGML*

*XHTML stiller skrappere krav til wellformedness end HTML*

Opgave 5:

Kig på og diskutér strukturen af [opskriftsamlingen](http://www.brics.dk/ixwt/examples/recipes.xml) i en browser.

*Namespace deklaration gælder fra og med den knude hvor den er defineret*

*Derfor vil Mathias mundtlige eksempel med at redefinere default namespace inde i dokumentet være en dårlig idé.*

*Namespace deklaration i recipes.xml:*

<rcp:collection xmlns:rcp="http://www.brics.dk/ixwt/recipes">

I dette tilfælde ville det have været lidt pladsbesparende at lade det være default namespace

Opgave 6:

Skriv [dette XML træ](http://www.brics.dk/ixwt/exercises/tree2.gif) i tekstformat. Check at resultatet er velformet, ved at kigge på det i Internet Explorer. Check det derefter igen med nedenstående Java-værktøj.

XML well-formedness checker: java -classpath [jdom.jar](http://www.daimi.au.dk/dWebTek/tools/jdom.jar):. [CheckWellformed](http://www.daimi.au.dk/dWebTek/tools/CheckWellformed.class) *foo.xml*  
  
Opgave 7:

Kig på [dette XML dokument](http://www.brics.dk/ixwt/exercises/namespaces.xml) (brug *View Source*). Hvilke namespaces tilhører elementerne foo, bar, og baz?

<a:foo qux="A" xmlns:a="B" xmlns:b="C">  
<b:bar a:quux="D" xmlns="E"/>  
<xmlns xmlns:a="F" xmlns="G">  
<baz a:corge="H" xmlns:baz="I" xmlns=""/>  
</xmlns>  
</a:foo>

*foo tilhører B*

*bar tilhører C*

*baz tilhører default ns (den tomme streng)*

Hvilke namespaces tilhører attributterne qux, quux, og corge? (Læs om de [mærkelige navne](http://en.wikipedia.org/wiki/Metasyntactic_variable).)

*qux tilhører B fordi den er attribut i a:foo og dermed arver ns fra a*

*quux er henvist som a:quux og tilhører derfor ns B*  
*corge tilhører F fordi a er redefineret*  
  
Bonus:

Opgave 2.2 ([fil](http://www.brics.dk/ixwt/exercises/driving.txt)). Prøv at strukturere dataene i så høj grad som muligt. Tænk på at lave logisk markup, ikke på at beskrive den fysiske opstilling i filen ved for eksempel at lave en line knude der direkte indeholder al rå tekst data for hver linje.

*Se løsning på google code*  
  
Opgave 8:

Forklar forskellen på *well-formed* og *valid* med egne ord

*wel-formed så overholder den XML standarden*

*valid er i forhold til dokumentets skema*

Opgave 9:

Kig på [DTD for XHTML](http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd) og svar på disse spørgsmål

* Hvad er content modellen for head elementet?

*Sådan er head beskrevet i dtd'en:  
<!ELEMENT head (%head.misc;,  
     ((title, %head.misc;, (base, %head.misc;)?) |  
      (base, %head.misc;, (title, %head.misc;))))>  
  
Content model er denne del: (%head.misc;,  
     ((title, %head.misc;, (base, %head.misc;)?) |  
      (base, %head.misc;, (title, %head.misc;))))  
  
%head.misc foldes ud til: (script|style|meta|link|object)\**

* Hvilke attributter kan et h3 element have?

<!ENTITY % attrs "%coreattrs; %i18n; %events;">  
  
*makroerne (entities) er defineret som følger  
<!ENTITY % coreattrs  
 "id          ID             #IMPLIED  
  class       CDATA          #IMPLIED  
  style       %StyleSheet;   #IMPLIED  
  title       %Text;         #IMPLIED"  
  >  
  
<!ENTITY % i18n  
 "lang        %LanguageCode; #IMPLIED  
  xml:lang    %LanguageCode; #IMPLIED  
  dir         (ltr|rtl)      #IMPLIED"  
  >  
  
<!ENTITY % events  
 "onclick     %Script;       #IMPLIED  
  ondblclick  %Script;       #IMPLIED  
  onmousedown %Script;       #IMPLIED  
  onmouseup   %Script;       #IMPLIED  
  onmouseover %Script;       #IMPLIED  
  onmousemove %Script;       #IMPLIED  
  onmouseout  %Script;       #IMPLIED  
  onkeypress  %Script;       #IMPLIED  
  onkeydown   %Script;       #IMPLIED  
  onkeyup     %Script;       #IMPLIED"  
  >  
  
Svaret er: id, class, style, title, lang, xml:lang, dir og masser af events fra events makroen.*

* Skal input elementer stå inden i form elementer?

Form er defineret således  
  
<!ELEMENT form %form.content;>   <!-- forms shouldn't be nested -->  
  
som er defineret således  
  
<!ENTITY % form.content "(%block; | %misc;)\*">  
osv.:  
<!ENTITY % block  
     "p | %heading; | div | %lists; | %blocktext; | fieldset | table">  
  
  
Vi starter lige nedefra i stedet  
  
<!ENTITY % inline.forms "input | select | textarea | label | button">  
  
som anvendes her  
  
<!ENTITY % inline "a | %special; | %fontstyle; | %phrase; | %inline.forms;">  
  
<!ENTITY % Flow "(#PCDATA | %block; | form | %inline; | %misc;)\*">  
  
*Det ser ud til at blive meget langhåret af dechiffrere spørgsmålet ud fra DTD'en. I stedet er der testet ved at skrive et html dokument og validere det, se projekt subversion.*  
*Svaret er NEJ.  
Motivationen er, at man ønsker at input tags skal kunne være inde i tabeller*.

Opgave 10:

Skriv et XML dokument, der er gyldigt for [dette DTD skema](http://www.brics.dk/ixwt/exercises/store.dtd). Valider dokumentet med nedenstående Java-værktøj.

DTD validator: java -classpath [jdom.jar](http://www.daimi.au.dk/dWebTek/tools/jdom.jar):. [ValidateDTD](http://www.daimi.au.dk/dWebTek/tools/ValidateDTD.class) *foo.xml*  
  
*Se på google code*

Opgave 11:

Oversæt [dette DTD skema](http://www.brics.dk/ixwt/exercises/moviestudio.dtd) til XML Schema.

*Se på google code*

*Namespace "http://www.w3.org/2001/XMLSchema" er anvendt som default namespace.*

Opgave 12:

Skriv et XML dokument som er gyldigt i forhold til dette [XML Schema skema](http://www.brics.dk/ixwt/exercises/movies.xsd). Kontroller din løsning med nedenstående værktøj

XML Schema validator: java -classpath [jdom.jar](http://www.daimi.au.dk/dWebTek/tools/jdom.jar):. [ValidateXMLSchema](http://www.daimi.au.dk/dWebTek/tools/ValidateXMLSchema.class) *foo.xml*  
XML Schema Quality Checker: java -classpath [sqc.jar](http://www.daimi.au.dk/dWebTek/tools/sqc.jar) com.ibm.sketch.util.SchemaQualityChecker *foo.xsd*