Introduction to web development using XHTML and CSS

Lars Larsson

Lecture #1

Material

During this course, we will cover server side web development with JavaServer Pages. JSP is an exciting technology that lets developers dynamically generate XHTML, XML and other types of documents in response to Web client requests. It also creates an easy way to provide users with interfaces to web services.

Course literature

This course uses the following freely available resources as course literature:

- Lecture slides,
- Core Servlets and JavaServer Pages, 1st Ed. by Marty Hall,
- More Servlets and JavaServer Pages, 1st Ed. by Marty Hall,
- Thinking in Java, 3rd Ed. by Bruce Eckel and
- any extra links posted on the web site.

You are encouraged to support the authors by buying the printed versions of the books. All the resources listed above are used with permission.

Assignments

3 compulsory assignments, 1 voluntary (for those who want the possibility of passing with distinction). The assignments will cover the following topics:

1. Basic JSP,
2. data validation and storage – sessions,
3. tag libraries and
4. as-of-yet undecided, but involving web services.

Specification for the first assignment is already available on the web page, the others will be uploaded in time for you to complete them.

Teachers

This course has three teachers, P-O Östberg, Dennis Olsson and me, Lars Larsson. We will be available in our rooms, as well as via e-mail and for chat sessions. See the web page for office hours and when we will be in the chat room. Send all e-mail questions to the following addresses (always send to all of them):

- `p-o_5dv076@cs.umu.se`,
- `larsson+webserver@cs.umu.se` and
- `denniso+webserver@cs.umu.se`

The chat room is `#5DV076` at the IRC server `irc.acc.umu.se`. P-O is also available on MSN (using the address above).

Important email information

Questions that may be of interest to all students attending the course will be sent to the course mailing list. The mailing list will send the mail to your CS mail address (`username@cs.umu.se`). Make sure that you either get into the habit of checking this address daily or forward it to your main account!

Support (`http://support.cs.umu.se/`) has information on how to handle your mail.

Course registration

To register for the course, send an email to the following address stating your name and Swedish "personnummer" – those with "Villkor 10" also need to provide proof that they may attend the course:

`p-o_5dv076@cs.umu.se`

Registering is **very** important, since we cannot grade your assignments unless you have registered. Do it as soon as possible!

Development environment

Your solutions to the assignments will be written in JSP. A easy-to-install package containing Tomcat, Axis and Derby will be available on the web page. In addition, you will need to install Ant and Java.

All your XHTML and CSS must pass W3C validation. The web browser that will be used for evaluating your solutions will be the latest version of Firefox.

Distance education

As stated on the web page, it is possible to complete this course from other locations than Umeå. It is of course recommended to attend the lectures if possible, but it is **not** a formal requirement.

The lecture slides will be (or have been) written in such a manner that it should be possible to understand them without attending the lectures.

XHTML and CSS

All students attending this course are required to have taken a course in client side web development (such as 5DV077), containing basic programming in JavaScript or equivalent. This lecture will attempt to bring all students up to speed on XHTML (eXtensible HTML) and CSS (Cascading Style Sheets) – the next will cover basic object-oriented development in Java.

We assume that most students have some basic previous knowledge of HTML, and we will focus on differences between HTML and the combination of XHTML and CSS.

XHTML 1.0 versions

There are three versions of XHTML 1.0:

- XHTML 1.0 Strict – the most strict standard, allows no presentational information
- XHTML 1.0 Transitional – compatibility version, that allows some presentational information to ease the transition from HTML to XHTML
- XHTML 1.0 Frameset – used to divide an XHTML page in two or more frames

During this course, we will use XHTML 1.0 **Strict** combined with CSS.

All XHTML documents must, at the very least, contain the following:

- A **document type declaration** that identifies that the document is, in fact, XHTML and should be interpreted as such and
- the **html** tag as the one containing all other tags. These are placed in one of the following sections:
  - the **head** section, containing information such as the title of the page and other information that is not displayed in the main window of the web browser and
  - the **body**, containing the elements (tags and their content) that shall be displayed on screen.

DOCTYPEs

We have three versions of XHTML (strict, transitional and frameset) and we need to tell the web browser which one should be used for the current page. The following line, placed on the first line in the file, identifies the document as XHTML 1.0 Strict:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The others are written in the same way, exchanging “strict” in both places for the version that should be used instead.

XHTML tags

XHTML 1.0 contains the same tags as HTML 4.01, but requires valid XML syntax. Chiefly, this means that XHTML requires all tags to be **well-formed** (opened and closed and properly nested). A single erroneous tag invalidates the entire page. The following is valid XHTML, and specifies that we want to make some text in a paragraph (<p>) bold (<b>).

```
<p>Normal text. <b>This part is bold</b>.</p>
```

XHTML tags continued

In XHTML, even empty elements (also known as singular tags) such as `img` must be closed. Singular tags are closed immediately, like so:

```
<img src="circle.gif" alt="A red circle" />
```

Note the slash at the end, immediately closing the tag.

XHTML tags continued

HTML attribute minimisation is not permitted in XHTML, so while this was valid HTML:

```
<option selected>...
```

it must be written this way in XHTML:

```
<option selected="selected">...
```

XHTML 1.0 Strict

Tags that were deemed entirely presentational, such as `font` and `center` are not permitted for use in XHTML 1.0 Strict. Instead, we must use XHTML in combination with CSS to specify the look of our pages.

XHTML tags

We will not present all XHTML tags here, since that takes too long and is beyond the scope of this course. You are recommended to visit the following site for a complete reference:

`http://www.w3schools.com/tags/`

CSS

CSS (Cascading Style Sheets) may be new to students who have taken an HTML-based course in web design. Thus, we devote some time to the **very basics** of CSS. You should study the topic on your own to accomplish what you need, this is just a primer.

CSS basics

CSS lets us apply rules that affect the way elements are presented on a web page. The basic format of CSS is:

```
selector {
  propertyA: valueA;
  propertyB: valueB;
  ...
}
```

The next slide contains a simple example.

CSS basic example

```
h1 {
  color: green;
}
```

The CSS above will change the color of all level one headers to green.

Basic CSS selectors

There are many selectors in CSS that allows the developer to match elements based on their relationship to other nodes, but we will for time reasons look only at the most basic ones – the others are left for independent study.

- Type selector,
- Ancestor selector,
- Class and ID selector and
- combined selectors.

Type selector

The example we just saw used the type selector: we specify a tag name, and the rules are applied to all tags of the specified type. It's as simple as that!

Type selectors are very common, since we usually want all headers of a given level to look the same, and so on.

Ancestor selector

The basic format of the ancestor selector is `S1 S2`, where both `S1` and `S2` are selectors. This expression will match only such elements `S2` that have `S1` as an ancestor. For example, this will match only such `<a>` tags that are somewhere in a `<li>`:

```
li a { ...rules here... }
```

The list of ancestors may be of any length.

Class selector

We can for almost all elements in XHTML specify a class attribute, such as:

```
<p class="important">Important text</p>
```

To match only such tags that are of the class `important`, we can write:

```
.important { ...rules here... }
```

ID selector

Tags in XHTML can be given an ID, that is, a unique identifier. Many elements can belong to the same class, but only one element can have a given ID:

```
<h1 id="foo">Header with ID</h1>
```

To match the only object that has an ID of `foo`, we write the following CSS selector:

```
#foo { ...rules here... }
```

Combining selectors

Selectors can be combined – for instance, the following will match **only paragraphs** of the class `foo`. Headers of the same class will not be matched.

```
p.foo { ...rules here... }
```

Properties and values

We cannot possibly list all CSS properties and their corresponding values here, as there are too many of them! Suffice to say that we can change **everything** about how a page and its elements is displayed.

See `http://www.w3schools.com/css/css_reference.asp` for a complete reference.

Container layout

XHTML has support for two main types of containers: `<div>` and `<span>`. The basic principle of CSS-powered web design is to put related content in a container, and use CSS to place them on the page, giving them the desired look (background colour, fonts and so forth). Use the following resource to learn the basics of CSS web design:

`http://www.w3schools.com/css/css_positioning.asp`

XHTML Forms

In this course, we will get user input using different controls: text input, selecting radio boxes or items from drop down menus and so forth.

All related controls must be placed in an enclosing `<form>` tag.

<form>

The <form> tag requires an attribute called action, which specifies the name of the resource (server side program) to which the data in the form shall be sent. For instance, if we have a page called sendEmail.jsp our form should be declared as such:

```
<form action="sendEmail.jsp">
...
</form>
```

<form> continued

Form data is sent to the resource specified by `action` using either a method called GET or POST. During a coming lecture, the difference between these will be discussed in detail. The default is GET – if we want to specify which method should be used, we can do so in the <form> declaration:

```
<form action="sendEmail.jsp" method="post">
...
</form>
```

<input>

User data can be entered via the <input> tag. As the attribute type, we specify what kind of input control we want to display: button, checkbox, file, hidden, image, password, radio, reset, submit or text. The default is text, meaning a box where a single line of text can be entered.

The submit type is special, since when clicked, it will make the web browser submit the data in the form to the form's action target.

<input> continued

One should always use the `name` attribute for controls, as it will be sent along with the data in the form. It will thus be easy to parse. An example:

```
<form action="sendEmail.jsp" id="email">
  <input type="text" name="email_address" />
  <input type="submit" value="Send e-mail" />
</form>
```

The code above will create a form with a single line input text field and a button with the text “Send e-mail”. When clicked, it will send the what the user entered in the text field to `sendEmail.jsp` and it will be easy to see that it was in the “address” field, since we gave it a name.

<fieldset>

To group related controls in a graphical box, we use the <fieldset> tag. It does not effect the data in the form, it is merely a graphical hint to your users that the controls are related.

Use the <legend> tag in the fieldset to show a title.

<select>

The <select> tag creates a drop down menu, where the user can select a single choice. The choices are listed as <option> tags within the <select> tag:

```
<select name="gender">
  <option value="female">Female</option>
  <option value="male">Male</option>
</select>
```

A large example form

See `example-form.html` for an example of many of the things that a form can do.

Best practises

An entire lecture will be devoted to best practises, but here are a few basic ones:

- include the name of the form in the names of the controls – it makes it easier to keep track of multiple forms on a single page,
- pick a naming convention with variables, and stick to it – some platforms ignore case, others (such as JSP) are case sensitive,
- avoid Swedish and other non-English characters in variable names.

Summary

We have briefly looked at some differences between HTML and XHTML combined with CSS. CSS-based web design is a huge topic, and students unfamiliar with it are recommended to take a course on it.

In particular, we have studied XHTML Forms and seen an example of how we can build a form easily.

Next lecture will cover basic Java programming.