Lab instructions: HTML Tables and JavaScript

In this course we will work with HTML, CSS and JavaScript. In the first lab we looked into HTML and CSS from the beginning. Now we will take this one step further by introducing JavaScript to make the pages even more Dynamic. But before this we will make use of another HTML construct, namely the `<TABLE>`. We mentioned tables briefly in the previous lab, in connection with Layout (which is not the intended usage). However, tables are still very useful for presentation of information from, for example, databases. Using JavaScript it is easy to make the tables dynamic and also make use of different views on the data.

Sizing and positioning

One important part of the layout is the sizing and positioning of the various components on the page. Both can be absolute, or relative. Absolute sizing and positioning can be useful in some contexts but is in many cases not recommendable. We want the user to be able to resize the pages according to his or her desire (at least as much as is practically possible) without having to scroll the page in order to see the whole page. Relative sizing of elements is a much more desirable way to layout the page, although this can also result in some strange results, if there is some erroneous scope declarations. A relative size is either relative to the window size or to the size of the parent element, and is expressed in "%" (of the parent containers size).

LayoutIdeas.html

In the file LayOutIdeas.html there are three divs that display the use of relative and fixed sizes. Try to resize the window and make it as small as possible. What happens to the layout? Can you explain the behaviour? Most important of all, why does the text in the last div in the HTML file behave as it does when you shrink the window? Can you use this feature in some way?

LayOutIdeas1.html

In the file LayoutIdeas1.html we have instead used 5 different classes to illustrate how positioning works. In the HTML file there are 7 divs, that have been given different classes, which results in some strange looking results. Go into the file and look at the styles. Try to understand how the positioning works (you might get some help if you look here: http://www.w3schools.com/css/css_positioning.asp) in the file as it is given.

Now try to change the classes of the divs. What happens if you give all divs the same class (try each of the classes and see what happens for each)? Can you predict the result before you reload the page? Can you explain why the divs behave as they do?

With the use of positioning and sizing, it is possible to create very nice layouts, as soon as you understand how the different attributes work together.

Folder: Layout and positioning

In the folder Layout and positioning you will find a third example of how you can use layout and positioning plus layers to create some interesting displays. The heading stays at the top, even when you scroll the text in the window. The text flows under it, as well as under the blue sidefloat. Try to hover the mouse over the blue tilted square, and see what happens.

The big square on the middle of the page is a stretch of canvas. The canvas is a new construct which came with HTML5. It is mostly used with JavaScript in order to paint on it. You don't have to look at the JavaScript for the Canvas yet, since we have not yet looked at JavaScript so far.

Tables

Tables in HTML are syntactically simple, but can also provide you with some unexpected quirks, if you slip a little on the keyboard. The standard table structure provides you with a number of columns and rows, plus a table heading. The simplest table in HTML this looks as below:

Tables.html

```
<TABLE>
    <TR>
        <TH> N1 </TH>
        <TH> N2 </TH>
    </TR>
    <TR>
        <TD> 1 </TD>
        <TD> 2 </TD>
    </TR>
</TABLE>
```

As you can see if you load this into the browser, it will not look like much of a table. There are no borders between the elements, although at least they are separated by whitespace.

Borders on the table are added using… CSS (surprise?). You can add it in the table directly, but we will keep the HTML as clear as possible, and use proper CSS-style sheets. The attributes are fairly straight forward, and there are a large number of styles that will be appropriate. To add a first look to the table we put the following in a style sheet:

```
table {
    border: 2px solid darkred;
    border-collapse: collapse;
}
th {
    border: 1px dashed green;
    border-collapse: collapse;
}
td {
    border: 1px dashed blue;
    border-collapse: collapse;
}
```

The "`border-collapse:collapse`" attribute causes borders from the different tags to be collapsed into a single border. Try to remove one or more of them from the style-sheet and see what happens.

Cell padding and spacing

There are two interesting attributes that will be useful in making the tables look better, namely padding and spacing. The "`padding`" attribute describes how much space will be added between the cell content borders. This is used to make the table more

spacious and easy to read. Try to add the padding line to the attributes of the cell data <TD> tags. Did you see any difference?

```
td {
    border: 1px dashed blue;
    border-collapse: collapse;
    padding: 4px;
}
```

Spacing, on the other hand describes how much space there should be between the cells themselves. The attribute is written `border-spacing` and is applied only to the <TABLE> element. Try to add it to your table, and see how the table changes. Note that this attribute collides with the `border-collapse` attribute.

Row- and columnspan

Finally there is also a possibility to have a cell span over two or several rows or columns, using the `rowspan` and `colspan` attributes. Compared to the previous attributes this is best added directly in the HTML code (although you could of course use classes or IDs to select the cells that should have the span attribute). So, the following table will have a header with two columns and a data row with three, and the first header element will cover two rows:

```
<TABLE>
    <TR>
        <TH colspan="2">N1</TH>
        <TH>N2</TH>
    </TR>
    <TR>
        <TD>1</TD>
        <TD>2</TD>
        <TD>3</TD>
    </TR>
</TABLE>
```

This example will be the last on HTML formatting and in the next part of the lab we will go through the basics of using JavaScript for web programming.

JavaScript

JavaScript is one fundamental part of Web programming. It is commonly used, and once you master the language you will be able to perform amazing feats with a simple (well, not quite simple) web page. Just let us start by checking a few things out before we start.

1. JavaScript is **not** java. Although it might be tempting to think of them as very similar, they are not. The syntax is somewhat resembling, but there are a number of pitfalls you might end up with. There is something called object-orientation in JavaScript, but this is completely different from Java OOP.
2. JavaScript is a scripting language. This means that it will be interpreted when it is used. This can take some time in certain cases. Unfortunately, this also has the consequence that JavaScript can be very difficult to debug, since the errors will appear in the silent bowels of the web browser.
3. JavaScript will have some constraints in what you can do, not least due to that it is used for web programming. This means that sometimes you can't use the

most straightforward approach, but will have to make a workaround. And sometimes it becomes very difficult.

Debugging

Although it might sound depressing, the first thing you will have to do is to understand the debugging of the scripts (home pages). Since JavaScript is a very silent language, it is seldom it will tell you anything about what went wrong. However, most browsers do give you some support, and this is how it works in Chrome and Firefox:

Right-click on the web page in the browser. In the menu that appears, select "Inspect Element" (the lowest menu item). When you do this the window splits in two parts and in the lower part you have a work space with some tabs and other information. The two most important and useful tabs are Elements (Chrome) or Inspector (Firefox), and the console (both browsers). In the first tab the structure of the HTML content is shown, and you can fold and unfold elements to get a better structure view of larger pages. In the second you will see any possible error message from JavaScript (not always very legible, unfortunately). Make sure that you know how to find this debugging support, because you will definitely need it.

Introduction to JavaScript in web programming

JavaScript is used to add interactive control over the web pages. There are already some interactive elements in the shape of `<form>` elements, where you can use buttons and input text fields to enter data from the user. However, if we want to make the page a bit more interactive we need to use some programming as well. Enter JavaScript.

JavaScript, just as styles can be added inline, in the tags, or as an internal script or as an external file (or several). We will most of the time be using the external file version in order to keep the HTML code as clean and easy to read as possible.

There is a complete reference guide to JavaScript at the w3Schoolswebsite:

http://www.w3schools.com/jsref/default.asp

It might be a good idea to bookmark that web page from now on, so that you have the reference available at all times. You will definitely use it. Also, there are a large number of dedicated JavaScript tutorials on the net, which provide you with many details on the programming language. Here we will just give you the basic introduction to the language, together with a number of examples (example-based learning).

So, how do we start? Actually we will start by looking at a small example from the W3Schools:

```
<!DOCTYPE html>
<html>

<head>
</head>
<body>

   <h1>My First JavaScript</h1>
   <button type="button
         onclick="document.getElementById('demo').innerHTML = Date()">
         Click me to display Date and Time.</button>
    <p id="demo"></p>
</body>
</html>
```

If you run this in a browser, you will get a button that, when pressed, displays the date and time below. No big deal, you might think, but there is one important thing to

note here. The web page was **not** reloaded. Instead the date was inserted directly in its place on the web page. This is the most important feature of using JavaScript. There is no need to link between pages in order to change the information. So what happened in the code?

The standard stuff for defining the web page is on top, and the interesting stuff is in the body. First there is an element that defines a standard button shape. It is also of type "button", which makes it clickable and initialises some other things. Then there is an attribute "onclick" which contains a small piece of JavaScript code.

The code snippet:

1. checks in the current document for the element with ID 'demo'
2. calls a JavaScript function that returns the date.
3. and replaces the content of the element with ID 'demo' with the date.

The "internalHTML" points to the content of an HTML element, in this case the `<p id="demo"></p>` line. Try to load the file and see how it works it.

FirstJavaScript.html

If you want more things to happen, we might need to put the code in an external file instead. See the files: `FirstJavaScript.js` and `FirstJs.js` for how this is done. Essentially we have created a function in the JavaScript file and replace the inline code with a call to this function. The external JavaScript files are executed, as if they were residing within the HTML file where they are linked, which means that "`document`" now refers to the HTML-file from which the function is called.

SecindJavaScript.html

In the JavaScript file, we can define more functions to manage specialized functions. In the file SecondJavaScript.html we have created a function that takes the string used for the day in the date string and returns the full name of the day. So, "Sat" becomes "Saturday" instead.

As you can see in the .js file, JavaScript is very good at handling strings, cutting, appending and rearranging, something which is very useful in web programming contexts.

Now the date that is being output is not very well-formed. Try to edit the script so that:

1. The month names are written in full.
2. There is some spacing and commas to separate the parts of the date.
3. The day and month are written in the reversed order.

If you want a small challenge, try to remove any unnecessary zeroes in the day number.

Input.html

In the file Input.html we use an input field instead, and get the value, processes it, and replaces the inner part of the html with a new message. Essentially, you now know how to replace any readable text on the page with any other text, but there is much more that you can do. What happens if we replace the code below:

```
function feedBack(){
    name=document.getElementById("intxt").value;

    document.getElementById("namn").innerHTML=name + " is a beautiful
name";
}
```

with:

```
function feedBack(){
   name = document.getElementById("intxt").value;

   if (name != "") {
        document.getElementById("namn").innerHTML =
             name + " is a beautiful name";
   } else {
        document.getElementById("namn").innerHTML =
             '<img src="Bear.jpg"> ';
   }
}
```

and you forget to enter your name into the input box (using the lab folder)? What is the conclusion?

The combination of HTML, css and JavaScript is very powerful, and as you have seen you can even insert new HTML elements in the file on the go. But how would you go about to change a "`<div>`" with a certain content or ID with another content or to a div with a different ID?

ChangingIdentities.html

It is actually fairly simple to change the HTML, and using style sheets it is even possible to exchange div:s with different identities with each other. In the file ChangingIdentities.html this is done with JavaScript and it is easy to see since the two div identities used have been given two distinctly different looking styles.

Folder: Layout and positioning

As a final exercise, go back to the folder Layout and Positioning and open the JavaScript file canvasapp.js and go through the code. Try to see if you can understand what is happening there. With the help of the comments you should at least be able to get a grasp of what is happening. But don't worry if you cannot follow the drawing commands currently, we will get back to the Canvas a little bit later in the course. It will be used for some graphics and simple animation.

Conclusion

This lab has provides you with the basics for how to use JavaScript and HTML (+css) to create dynamic web sites, where there is a large room for fantasy and creativity. What has not been covered so far is the more normal constructs for programming, such as loops, mathematic calculations, and finally the use of the Canvas element for graphic manipulations. We will come back to the Canvas later in the course, but the more standard programming constructs will be up to you to keep learning. Despite what was said in the beginning, the standard programming concepts are relatively close to Java or C, so it is only a matter of smaller syntactic changes.