Week 7 – Presentation 4

Style Sheets

Skins

- Last JavaFx subject, superficial in all meanings of the word but important (looks sell): how to change the appearance of a JavaFx application. I have already briefly talked about it, the best way is to do it through an external style sheet (.css file). People will be able to change, often in a very impressive way, the looks of your application by changing this file and without any need to access the code (in fact, they just need the .css and the .class to run the "modified" application).
- If you want to see how far you go with "styling", you can visit http://csszengarden.com and click on designs on the right hand- side. The same page will look completely different.

CSS ZEN GARDEN

The Beauty of CSS Design

Select a Design:

- Mid Century Modern by Andrew Lohman
- Garments by Dan Mall
- Steel by Steffen Knoeller
- Apothecary by Trent Walton
- Screen Filler by Elliot Jay Stocks
- Fountain Kiss by Jeremy Carlson
- A Robot Named Jimmy by meltmedia

Cascading Style Sheet (CSS)

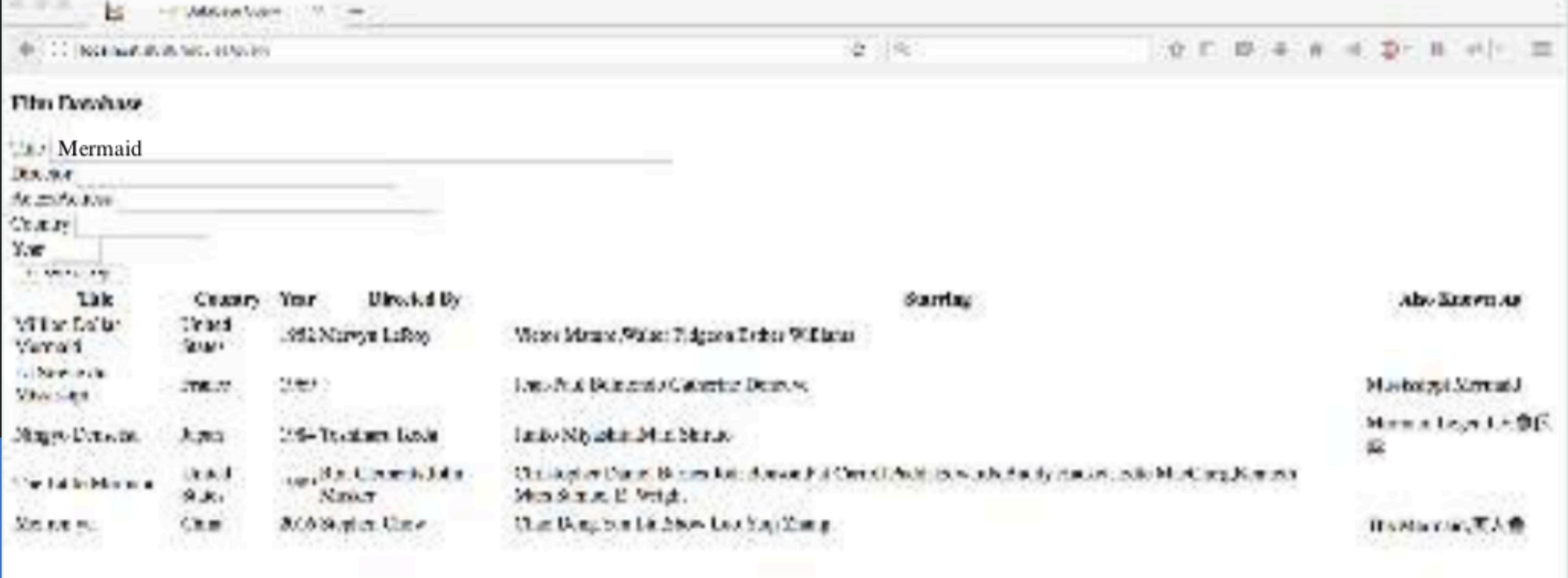
CSS stands for Cascading Style Sheet and Cascade is French for waterfall.
The name emphasizes that styles "drip down”, and are inherited from previous style sheets

It all starts with HTML…

The idea is stolen from web applications. Web applications just send HTML pages to browsers that decode and display these pages. An HTML page is organized by sections between pairs of tags (<tag> at the beginning, </tag> at the end) that structure the document and can contain in turn other pairs of tags, thus defining a kind of hierarchical structure (note that some tags, such as those for images, act both as opening and closing tags). Tags pretty often also contain attributes (such as the image file name for an image tag).

<tags>

In the very early days of the web, people were using tags to format their pages, for instance what they wanted in bold was between **<bold>** and **</bold>** (inspired by previous document generation systems that were sending special signals to printers), and you could change the fonts with **<font** *attributes specifying the font, size and everything***>** ... **</font>**. As websites were growing in size and number of pages, and as increasingly pages were generated by programs instead of being created by hand, it became unmanageable, especially when the marketing department was deciding on new corporate colors. So the idea was to associate formatting to tags in one or several separate text files.



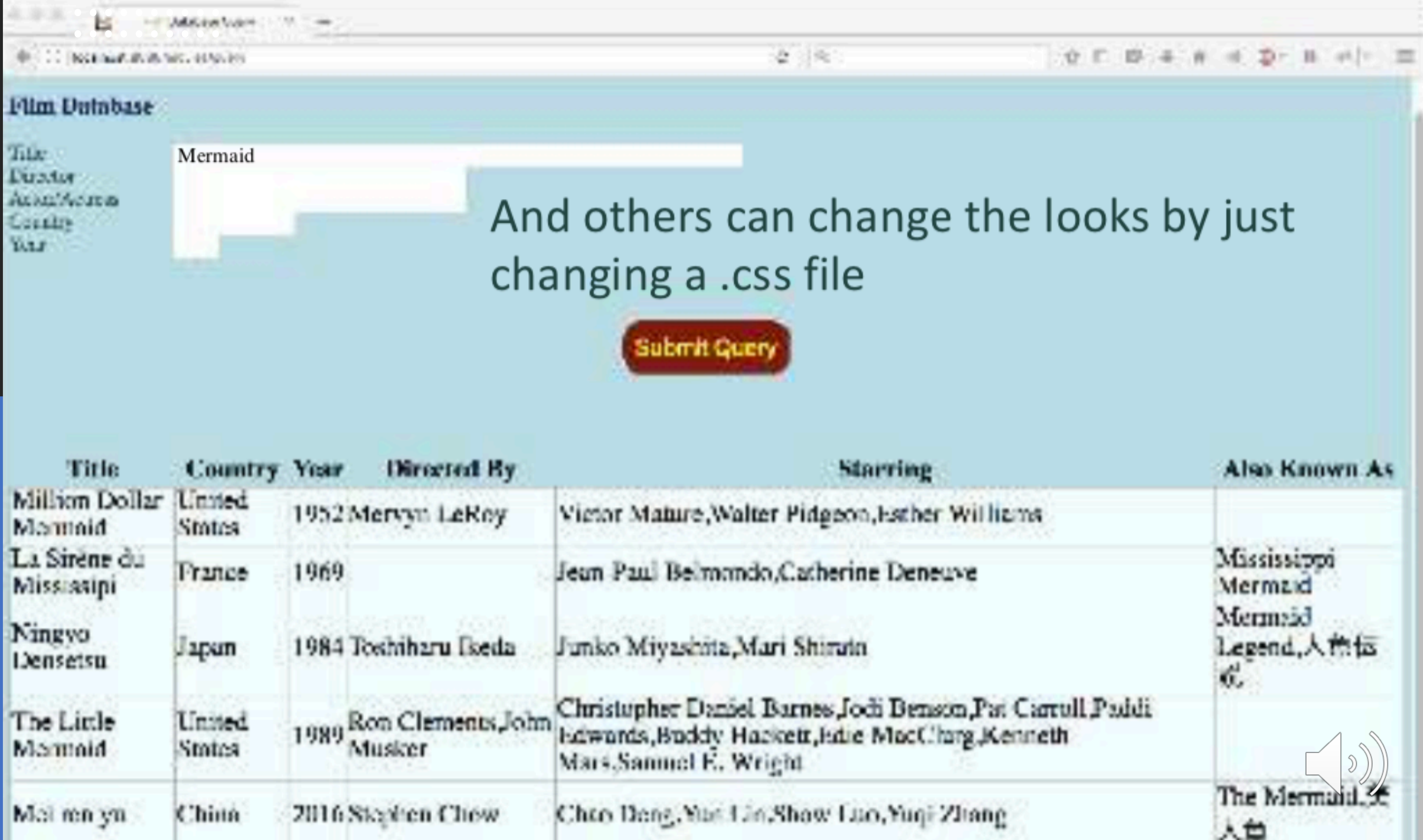
Film Database

Title: Mermaid
Director:
Actor/Actress:
Country:
Year:

Title	Country	Year	Directed By	Starring	Also Known As
Million Dollar Mermaid	United States	1952	Mervyn LeRoy	Victor Mature, Walter Pidgeon, Esther Williams	
La Sirène du Mississipi	France	[illegible]		Jean-Paul Belmondo, Catherine Deneuve	Mississippi Mermaid
Ningyo Densetsu	Japan	1984	Toshiharu Ikeda	[illegible]	Mermaid Legend, 人魚伝説
The Little Mermaid	United States	1989	Ron Clements, John Musker	Christopher Daniel Barnes, [illegible]	
Mei ren yu	China	2016	Stephen Chow	Chao Deng, Yun Lin, Show Luo, Yuqi Zhang	The Mermaid, 美人鱼

Some people could just focus on the "engine", such as here an application allowing to retrieve film information from a database ...

And others can change the looks by just changing a .css file


Film Database

Title	Mermaid
Director	
Actor/Actress	
Country	
Year	

Title	Country	Year	Directed By	Starring	Also Known As
Million Dollar Mermaid	United States	1952	Mervyn LeRoy	Victor Mature,Walter Pidgeon,Esther Williams	
La Sirène du Mississipi	France	1969		Jean Paul Belmondo,Catherine Deneuve	Mississippi Mermaid
Ningyo Densetsu	Japan	1984	Toshiharu Ikeda	Junko Miyashita,Mari Shirato	Mermaid Legend,人魚伝説
The Little Mermaid	United States	1989	Ron Clements,John Musker	Christopher Daniel Barnes,Jodi Benson,Pat Carroll,Paddi Edwards,Buddy Hackett,Edie McClurg,Kenneth Mars,Samuel E. Wright	
Mei ren yu	China	2016	Stephen Chow	Chao Deng,Yun Lin,Show Luo,Yuqi Zhang	The Mermaid,美人鱼

The way it works in web pages

- Before I discuss about CSS in JavaFx, I'm going to talk about CSS with HTML, because there is far more CSS written for HTML than for JavaFX.
- There are three main ways to specify how to display what is between tags.
- 1. You can specify for a given tag, associating a tagname with visual characteristics
  2. I have mentioned that tags can have attributes, one is "class" (unrelated to object-oriented programming) listing one or several categories. This allows for creating a subcategory, or to give some common visual characteristics across different tags that have the same class (for instance "inactive")
- 3. You can give another attribute "id" to a tag, an this allows you to make one particular tag look really special.

```
<tag>          tag {
                   attribute: value;
                         ...
               }
```

I focus on simple tags in the next example.

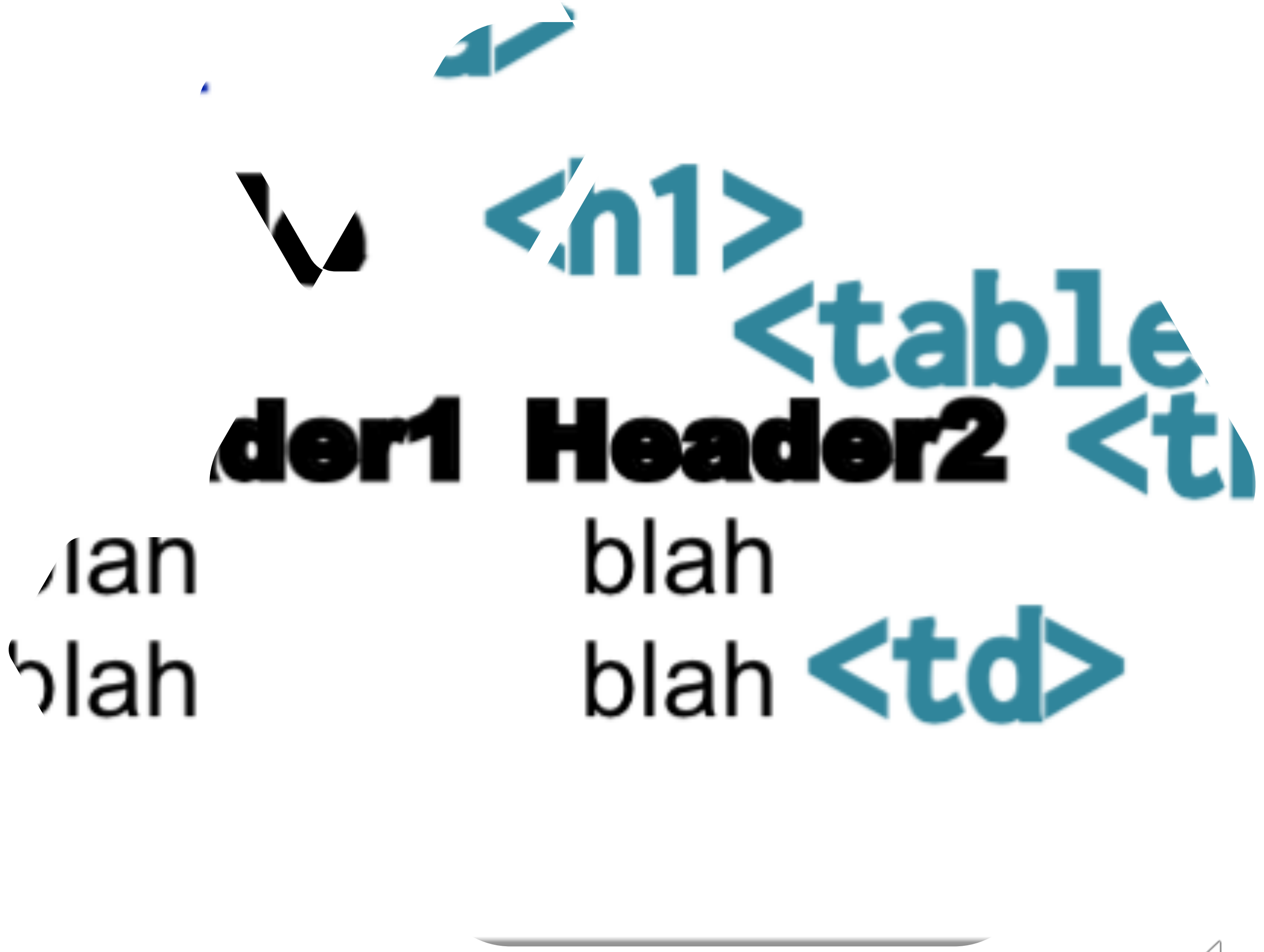
```
<tag class="category">
               .category {
                   attribute: value;
                         ...
               }
```

You have here how the tag looks in HTML and how styling looks in CSS.

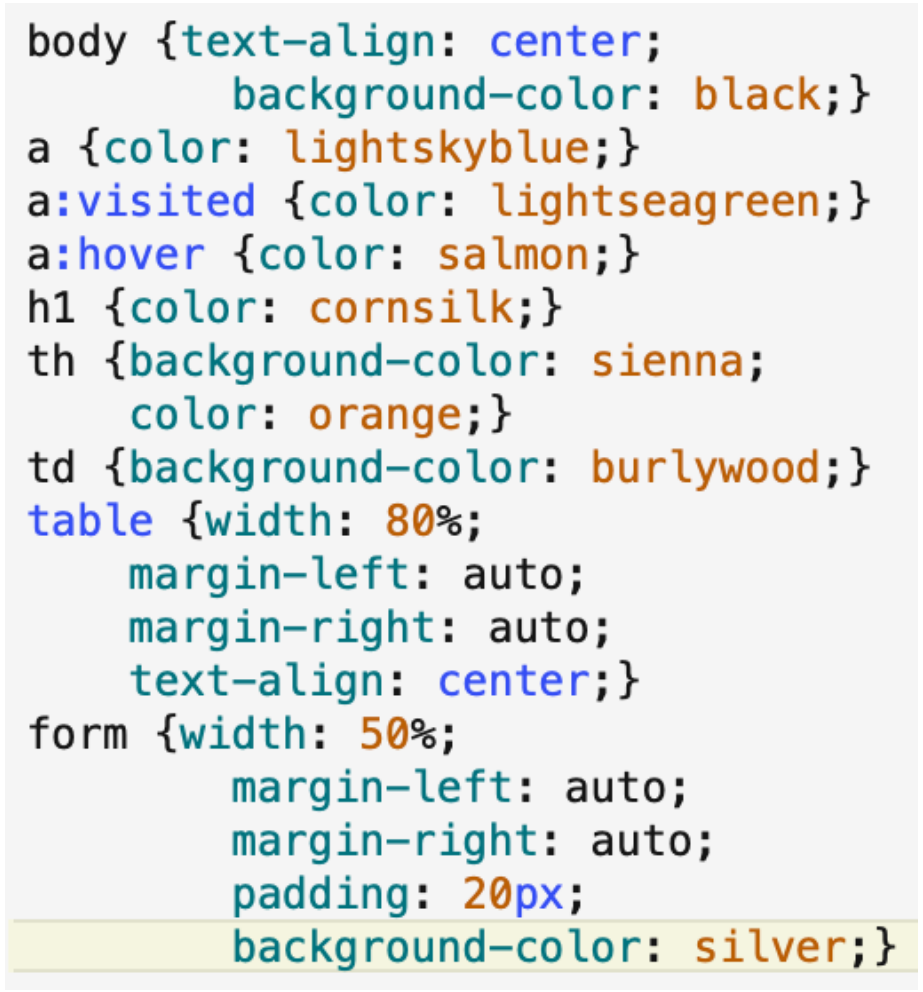
```
<tag id="name">
               #name {
                   attribute: value;
                         ...
               }
```

<body>

- In a HTML page, everything that is displayed is between <body> and </body> tags, so <body> really defines the global looks of the page.
- In my example, I'll use tags <a> associated with a link, <h1> with a (first level) header, <table>, <th> (table header) and <td> (table data). I have omitted <tr> (table row) which usually surrounds the columns of a same row. The tag will be there on the page, but I'm not associating any special style with it in my example.

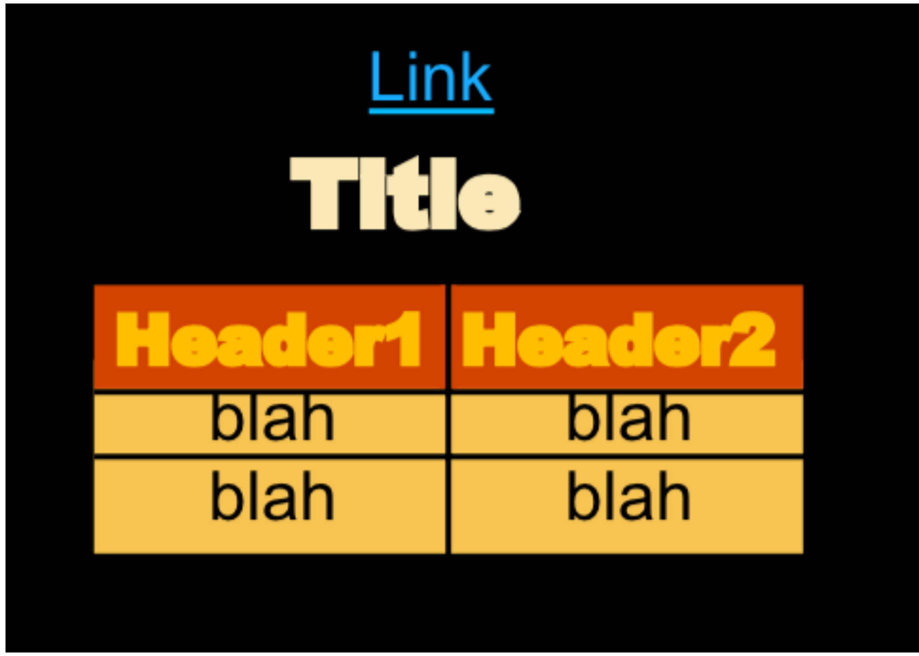


Notice the special a:visited (link you have already clicked on) and a:hover (when the cursor moves over the link)



```
body {text-align: center;
        background-color: black;}
a {color: lightskyblue;}
a:visited {color: lightseagreen;}
a:hover {color: salmon;}
h1 {color: cornsilk;}
th {background-color: sienna;
    color: orange;}
td {background-color: burlywood;}
table {width: 80%;
    margin-left: auto;
    margin-right: auto;
    text-align: center;}
form {width: 50%;
        margin-left: auto;
        margin-right: auto;
        padding: 20px;
        background-color: silver;}
```

styles.css



<link rel="stylesheet" href="styles.css"/>

Styling is written to a file that is included in the HTML page by a special tag in the <header>...</header> section that precedes the "body". Then you can change it when needed.

The way it works in JavaFX

Nodes are (almost) equivalent to tags

- .root plays the same role as body
- Otherwise use class names prefixed with a dot
- .button

You have of course no tags in a JavaFx application, but you have the same kind of hierarchy through nodes, containers and widgets. The JavaFx class names are used with the same syntax as the HTML classes in CSS, that means that they are prefixed by a dot.

The way it works in JavaFX

Nodes are (almost) equivalent to tags

- .root plays the same role as body
- Otherwise use class names prefixed with a dot
- .button

Attribute names are prefixed with –fx-

```
-fx-font-size: 150%;
```

CSS attributes also have a special name with JavaFx. The idea is to be able to have a single CSS files shared by a Web and a JavaFx application without having any conflict.

The way it works in JavaFX

Nodes are (almost) equivalent to tags

- .root plays the same role as body
- Otherwise use class names prefixed with a dot
- .button

Attribute names are prefixed with –fx-

-fx-font-size: 150%;

Node.setId("name")

Node.getStyleClass().add("css class")

Finally, node methods allow you to associate with a node the same kind of attributes as with a HTML tag. You can only have a single Id, but you can have several classes, and therefore getSyleClass() returns a list.

The way it works in JavaFX

Nodes are (almost) equivalent to tags

- .root plays the same role as body
- Otherwise use class names prefixed with a dot
- .button

Attribute names are prefixed with –fx-

-fx-font-size: 150%;

Node.setId("name")

Node.getStyleClass().add("css class")

Node.setStyle("-fx-attribute: value")

Scene scene = new Scene(new Group(), 500, 400);
scene.getStylesheets().add("path/styles.css");

(then load into the application)

Not Everything cannot be styled with CSS in JavaFX

CSS styling allows you to go rather far in JavaFx, but not as far as you could go in HTML. Some elements may prove hard to style with CSS. You may sometimes have to code some styling in the Java application. However, if you still want this styling to be "externalized", don't forget that properties files also provide a way to read attributes at run-time. It's of course better to have all styling at one place, but it's better to use a properties file than to hard-code.

A Properties file might sometimes be a workaround

The End